1999-11

Approximate declarative semantics for rule base anomalies

Zhang, Du; Luqi

Elsevier

Zhang, Du and Luqi. "Approximate declarative semantics for rule base anomalies." Knowledge-Based Systems 12.7 (1999): 341-353.
https://hdl.handle.net/10945/65274

ELSEVIER

Knowledge-Based Systems 12 (1999) 341–353

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

Approximate declarative semantics for rule base anomalies

Du Zhang[a,*], Luqi[b]

[a]*Department of Computer Science, California State University, Sacramento, CA 95819-6021, USA*
[b]*Department of Computer Science, Naval Postgraduate School, Monterey, CA 93943, USA*

Received 1 December 1998; received in revised form 9 June 1999; accepted 18 June 1999

Abstract

Despite the fact that there has been a surge of publications in verification and validation of knowledge-based systems and expert systems in the past decade, there are still gaps in the study of verification and validation (V&V) of expert systems, not the least of which is the lack of appropriate semantics for expert system programming languages. Without a semantics, it is hard to formally define and analyze knowledge base anomalies such as inconsistency and redundancy, and it is hard to assess the effectiveness of V&V tools, methods and techniques that have been developed or proposed. In this paper, we develop an approximate declarative semantics for rule-based knowledge bases and provide a formal definition and analysis of knowledge base inconsistency, redundancy, circularity and incompleteness in terms of theories in the first order predicate logic. In the paper, we offer classifications of commonly found cases of inconsistency, redundancy, circularity and incompleteness. Finally, general guidelines on how to remedy knowledge base anomalies are given.

Keywords: Knowledge base anomalies; Inconsistency; Redundancy; Circularity; Incompleteness; Knowledge base verification

1. Introduction

The last decade has witnessed a surge of publications in verification and validation (V&V) of expert systems and knowledge-based systems which resulted in several books [1,2], and special issues of several journals [3–6]. Major AI conferences have had workshops and special sessions that were devoted to the issue. A sample of additional publications can be found in Refs. [7–40]. Many V&V methods, techniques and tools have been proposed, developed or implemented for expert system applications. On the other hand, advances in knowledge engineering have resulted in better methodologies and practice that aim at reducing errors and faults during system development and maintenance [41–44]. Despite all these activities, there are still gaps in the study of V&V of expert systems, not the least of which is the lack of appropriate semantics for expert system programming languages. Without a semantics, it is hard to formally define and analyze knowledge base (KB) anomalies such as inconsistency and redundancy, and it is hard to assess the effectiveness of V&V tools, methods and techniques that have been developed or proposed.

V&V of expert systems in general and V&V of KB in particular need to be based on a sound theoretical foundation. However, the reality is that "the construction of either declarative or Hoare-style semantics for current rule-based languages is a hopeless task" [31]. In the long run, concern for verifiability and reliability should lead to the development of programming languages with tractable semantics for expert system applications. In the meantime, some approximate semantics (declarative or imperative) is needed to enable a formal analysis of properties of expert system components (such as a KB). For example, sketches of an approximate declarative semantics, which is based on a logical interpretation of a rule base, and an approximate imperative semantics, which is based on axiomatic logic and invariants, for the current rule-based programming languages were proposed in Ref. [31].

Adopting a declarative semantics for a rule-based language has some potential difficulties: (a) It is hard to provide a purely declarative interpretation of rules, because they often behave in an imperative manner with the intended side effects of updating a working memory. Simply treating a rule base as a logical theory may result in an excessively conservative semantics. (b) Due to the fact that consistency in the first order logic is semi-decidable, there does not exist an algorithm that can find all inconsistencies and redundancies in an arbitrary first order KB, thus, making it difficult to develop practical V&V tools.

* Corresponding author. Tel.: + 1-916-278-7952; fax: + 1-916-278-6774.

E-mail addresses: zhangd@ecs.csus.edu (D. Zhang), luqi@cs.nps.navy.mil (Luqi)

0950-7051/99/$ - see front matter
PII: S0950-7051(99)00032-5

Table 1
Typesetting conventions

Symbol	Meaning
Ð	A nonempty domain of elements
ζ	An interpretation
Boldface capital letter	Set of wff (literals), or set of rules
Ordinary capital letter	Individual wff (literal)
Lower-case ordinary letter	Constant
Lower-case italic letter(s)	Predicate
r_i	Rule label
f_j	Fact label
LHS (r_i)	Set of literals in the left-hand side of r_i
RHS (r_i)	Set of literals in the right-hand side of r_i
true, false	Logical values
x, y, z, x', y', z'	Variable

There have been several efforts toward providing a precise characterization of the logical nature of a rule-based KB [11,31,35]. An algorithm to detect all inconsistencies and redundancies in "a certain well-defined, reasonably expressive, subset of all quasi-first-order-logic KB" is presented in [11].[1] The results in [35] indicate that a rule-based language is still amenable to logical analysis.

The purposes of this paper are to (a) Provide an approximate declarative semantics for rule-based KB so that various KB anomalies can be formally defined and correctly understood. We go beyond the results of [11,31,35] by dealing with not only KB inconsistencies and redundancies, but also KB circularity and incompleteness. (b) Establish KB anomaly analysis procedures using theories in the first order predicate logic (such as the *model theory*, *satisfiability*, and *derivability* of certain tautologous well-formed formulas [45–47]). This may serve as the theoretical underpinnings of practical V&V tools. (c) Offer classifications for cases of inconsistency, redundancy, circularity and incompleteness commonly found in rule-based KB. (d) Propose guidelines on how to remedy the anomalies once they are identified.

The rest of the paper is organized as follows: Section 2 briefly reviews the terms and concepts to be used throughout the paper. Definitions, classifications and analyses of KB inconsistency, redundancy, circularity and incompleteness are provided in Sections 3–6, respectively. Some possible remedial measures for KB anomalies are discussed in Section 7. Section 8 concludes with remarks about future work.

[1] The key step in the algorithm is the subsumption tests which must be decidable for a given KB in order for the KB to be completely analyzed for inconsistency and redundancy. The subsumption tests will be decidable only when the expressions to be tested satisfy the *quantifier decoupled* (q-decoupled) property [11]. In general, one does not know in advance if a given KB will generate any non q-decoupled expressions because there does not exist a syntactic test for determining the q-decoupleability of the KB.

2. Preliminaries

We assume that the reader is familiar with the basic concepts and terminology in the first order predicate logic [45–47]. We use *wff* to denote the *well-formed formulas* in the predicate logic. An *atomic formula* (or *atom*) refers to an *n*-place predicate symbol and its *n* terms. A *ground* atom is one not containing any variables. A *literal* is an atom or its negation. To avoid confusion, we adopt the typesetting conventions as given in Table 1.

Definition 1. An *interpretation* of a wff consists of a nonempty domain Ð, and an assignment of "values" to each constant, function symbol and predicate symbol appearing in the wff according to the following: (a) assigning an element of Ð to each constant; (b) assigning a mapping from $Ð^n$ to Ð to each *n*-ary function symbol; and (c) assigning a mapping from $Ð^n$ to {*true*, *false*} to each *n*-ary predicate symbol.

Definition 2. A wff H (or a set **Ç** of wff) is *satisfiable (consistent)* if and only if there exists an interpretation ζ such that H (or every wff in **Ç**) is evaluated to *true* for all variable assignments[2] under ζ, which is denoted $\models_\zeta$ H ($\models_\zeta$ **Ç**). ζ is said to be a *model* of H (**Ç**) and ζ *satisfies* H (**Ç**). H (**Ç**) is *inconsistent* if and only if there exists no model for H (**Ç**). H is said to be *valid* (*tautologous*) if and only if every possible interpretation satisfies H. H is a *logical consequence* of **Ç** if and only if every model of **Ç** is also a model of H. This is denoted as **Ç** $\models$ H.

Theorem 1. Given a set of wff **Ç** = {P, …, Q} and a wff H, **Ç** $\models$ H if and only if P $\wedge$ … $\wedge$ Q $\rightarrow$ H is valid.

Definition 3. Let **Ç** and **Ç**′ be sets of wff. **Ç** $\approx$ **Ç**′ denotes that **Ç** is satisfiable if and only if **Ç**′ is satisfiable [45].

This paper focuses on rule-based knowledge bases. A rule-based KB can be divided into a set of *facts* which 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.

[2] A variable assignment is a mapping from variables in a wff to elements in Ð.

Table 2
Same, synonymous, complementary, mutual exclusive, incompatible, and conflict literals

Semantics	Syntax	
	Identical	Different
Equivalent	*Same:* denoted as $L_1 = L_2$[a]. L_1 and L_2 are syntactically identical (same predicate symbol, same arity, and same terms at corresponding positions)	*Synonymous:* denoted $L_1 \cong L_2$[b]. L_1 and L_2 are syntactically different, but logically equivalent
Conflict[c]	*Complementary*: denoted $L_1 \# L_2$. L_1 and L_2 are an atom and its negation	*Mutual exclusive*: denoted $L_1 \oplus L_2$. L_1 and L_2 are syntactically different and semantically have opposite truth values *Incompatible:* denoted $L_1 \not\approx L_2$. L_1 and L_2 are complementary pair of synonymous literals

[a] Given two rules r_i and r_k, if $LHS(r_i) = \{P1, \ldots, Pn\}$ and $LHS(r_k) = \{P1', \ldots, Pn'\}$, then $LHS(r_i) = LHS(r_k)$ iff $\forall i \in [1, n] Pi = Pi'$.
[b] Given two rules r_i and r_k, if $LHS(r_i) = \{P1, \ldots, Pn\}$ and $LHS(r_k) = \{P1', \ldots, Pn'\}$, then $LHS(r_i) \cong LHS(r_k)$ iff $\forall i \in [1,n]$ $Pi \cong Pi'$.
[c] L_1 and L_2 are conflict literals, denoted $L_1 \uparrow\downarrow L_2$, if $(L_1 \# L_2) \vee (L_1 \oplus L_2) \vee (L_1 \not\approx L_2)$.

Definition 4. Rules in a KB have the format: $P_1 \wedge \ldots \wedge P_n \rightarrow R$, where P_i's 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's and R are *literals*. If the conditions of a rule instance are satisfied by facts in WM, then its conclusion is deposited into WM.

Definition 5. 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 dynamic), or rule deduction.

Definition 6. 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.[3] Rule instances and negated literals can be utilized by the inference system, but are never deposited into WM [11].

Definition 7. Given two sets of literals $\mathbf{L}$ and $\mathbf{L}'$, $\mathbf{L}'$ is said to be a *specialization* of $\mathbf{L}$, denoted $\mathbf{L}' \trianglelefteq \mathbf{L}$, if there exists a nonempty set of *substitutions* θ, such that $\mathbf{L}' = (\mathbf{L})\theta$. In particular, a literal P' is a specialization of P, denoted as $P' \trianglelefteq P$ if there exists a nonempty set of substitution θ such that $P' = (P)\theta$.

Definition 8. Given a set $\mathbf{L}$ of n literals, $\rho(\mathbf{L})$ represents the set of all literal permutations in $\mathbf{L}$.

[3] There would be no effect on WM if R is not in WM when $\neg R$ is derived.

Definition 9. If r_i is a rule and P is a literal, the expression $r_i \vdash P$ is used to indicate arbitrary length derivation of P from r_i in terms of some inference methods.[4]

Using *logical equivalence*, we can always convert a logical implication into a disjunction of literals. We further simplify the notation by dropping the logical connective " $\vee$ " from such a disjunction. For instance, the set of wff $\{P \wedge Q \rightarrow R,\ U \wedge \neg V \rightarrow W\}$ has the following logically equivalent short representation: $\{\neg P\ \neg QR,\ \neg UVW\}$ where each element in the set is a disjunction of literals.

Definition 10. The concepts of the *same*, *synonymous*, *complementary*, *mutual exclusive*, *incompatible*, and *conflict* literals are defined in Table 2 in terms of syntax and semantics considerations.

Example 1. Given the following literals: *father*(*x*, john), *male_parent*(*x*, john), *animal*(sea_cucumber), *vegetable* (sea_cucumber), *bird*(fred), $\neg$*bird*(fred), *sent_to*(*x*, emergency_room), *sent_to*(*x*, waiting_room), *expensive*(*x*), *high_priced*(*x*), we have:

father(*x*, john) = *father*(*x*, john);
father(*x*, john) $\cong$ *male_parent*(*x*, john);
bird(fred)# $\neg$*bird*(fred);
animal(sea_cucumber) $\oplus$ *vegetable*(sea_cucumber);
sent_to(*x*, emergency_room) $\oplus$ *sent_to*(*x*, waiting_room);
expensive(*x*) $\not\approx$ $\neg$*high_priced*(*x*);
father(*x*, john) $\uparrow\downarrow$ $\neg$*male_parent*(*x*, john).

[4] Strictly speaking, the expression should be $\{r_i \cup WM\} \vdash P$ because facts in WM will be used during the derivation.

Table 3
Types of inconsistency

Type	Description	Pattern
I-1	Rules with the same LHS result in complementary conclusions	$LHS(r_i) = LHS(r_k)$ and $r_i \vdash P$ and $r_k \vdash Q$, where P#Q
I-2	Rules with shared condition(s) result in complementary conclusions	$LHS(r_i) \cap LHS(r_k) \neq \varnothing$ and $r_i \vdash P$ and $r_k \vdash Q$, where P#Q
I-3	Rules with the same LHS result in mutual exclusive conclusions	$LHS(r_i) = LHS(r_k)$ and $r_i \vdash P$ and $r_k \vdash Q$, where $P \oplus Q$
I-4	Rules with shared condition(s) result in mutual exclusive conclusions	$LHS(r_i) \cap LHS(r_k) \neq \varnothing$ and $r_i \vdash P$ and $r_k \vdash Q$, where $P \oplus Q$
I-5	Rules with the same LHS result in incompatible conclusions	$LHS(r_i) = LHS(r_k)$ and $r_i \vdash P$ and $r_k \vdash Q$, where $P \not\asymp Q$
I-6	Rules with shared condition(s) result in incompatible conclusions	$LHS(r_i) \cap LHS(r_k) \neq \varnothing$ and $r_i \vdash P$ and $r_k \vdash Q$, where $P \not\asymp Q$
I-7	Rules with synonymous LHS result in complementary conclusions	$LHS(r_i) \cong LHS(r_k)$ and $r_i \vdash P$ and $r_k \vdash Q$, where P#Q
I-8	Rules with shared synonymous conditions result in complementary conclusions	$\mathbf{L} \subset LHS(r_i)$ and $\mathbf{L}' \subset LHS(r_k)$ and $\mathbf{L} \cong \mathbf{L}'$ and $r_i \vdash P$ and $r_k \vdash Q$, where P#Q
I-9	Rules with synonymous LHS result in mutual exclusive conclusions	$LHS(r_i) \cong LHS(r_k)$ and $r_i \vdash P$ and $r_k \vdash Q$, where $P \oplus Q$
I-10	Rules with shared synonymous conditions result in mutual exclusive conclusions	$\mathbf{L} \subset LHS(r_i)$ and $\mathbf{L}' \subset LHS(r_k)$ and $\mathbf{L} \cong \mathbf{L}'$ and $r_i \vdash P$ and $r_k \vdash Q$, where $P \oplus Q$
I-11	Rules with synonymous LHS result in incompatible conclusions	$LHS(r_i) \cong LHS(r_k)$ and $r_i \vdash P$ and $r_k \vdash Q$, where $P \not\asymp Q$
I-12	Rules with shared synonymous conditions result in incompatible conclusions	$\mathbf{L} \subset LHS(r_i)$ and $\mathbf{L}' \subset LHS(r_k)$ and $\mathbf{L} \cong \mathbf{L}'$ and $r_i \vdash P$ and $r_k \vdash Q$, where $P \not\asymp Q$
I-13	Rules with consistent LHS result in complementary conclusions	$\models_\zeta \{LHS(r_i), LHS(r_k)\} \wedge LHS(r_i) \cap LHS(r_k) = \varnothing \wedge r_i \vdash P$ and $r_k \vdash Q$, where P#Q
I-14	Rules with consistent LHS result in mutual exclusive conclusions	$\models_\zeta \{LHS(r_i), LHS(r_k)\} \wedge LHS(r_i) \cap LHS(r_k) = \varnothing \wedge r_i \vdash P$ and $r_k \vdash Q$, where $P \oplus Q$
I-15	Rules with consistent LHS result in incompatible conclusions	$\models_\zeta \{LHS(r_i), LHS(r_k)\} \wedge LHS(r_i) \cap LHS(r_k) = \varnothing \wedge r_i \vdash P$ and $r_k \vdash Q$, where $P \not\asymp Q$
II-1	Rules with a condition result in complementary literal	$r_i \vdash Q$, where $P \in LHS(r_i) \wedge$ P#Q
II-2	Rules with a certain condition result in incompatible literal	$r_i \vdash Q$, where $P \in LHS(r_i) \wedge P \not\asymp Q$
II-3	Rules with a condition P result in mutual exclusive literal	$r_i \vdash Q$, where $P \in LHS(r_i) \wedge P \oplus Q$

In this paper, we do not consider the situation in which rules are augmented with *certainty factors*. Because of the way they are defined, rules and facts are subsets of wff. Therefore, the terms "rule" and "fact" can be freely replaced by the term "wff" throughout the rest of the paper.

3. KB inconsistency

3.1. Definition of inconsistency

The root cause of KB inconsistency is due to rules in RB, but its manifestation is through WM. For instance, the inconsistency of a RB containing a pair of rules $\{p(x) \rightarrow q(x),\ p(x) \rightarrow \neg q(x)\}$ is not apparent until a fact $p(a)$ is asserted into WM. In general, although the rules in a RB may be consistent on their own (because there exists a model for them), they can form an inconsistent theory when combined with certain facts in WM. In order for a KB to be consistent, there needs to be a model for both RB and WM.

On the other hand, facts in WM are changing over time due to dynamic assertions and retractions. If we use subscripts to denote states of WM at different times, RB may be consistent with WM_i, but inconsistent with WM_j where $i \neq j$. Thus, relying on a particular WM state in verifying the consistency of RB may not produce an accurate result.

Definition 11. Let WM_0 and $\mathbf{R}(WM_0)$ denote the initial state for WM and the reachability set of all possible WM states from WM_0, respectively. Let **WM** denote all legitimate facts[5] for an application. $\mathbf{WM} = \cup\{WM_i | WM_i \in \mathbf{R}(WM_0)\}$.

Definition 12. Given two interpretations ζ_i and ζ_j, ζ_j is an extension of ζ_i, denoted as $\zeta_i \sqsubset \zeta_j$, if the domain and assignments in ζ_i are retained in ζ_j.

Definition 13. Let ζ_0 be a model for **WM**.[6] A KB is *inconsistent* if and only if $\neg\exists\zeta\ [\zeta_0 \sqsubset \zeta \wedge \models_\zeta \mathrm{RB}]$.

During problem solving process, inconsistent rules in RB allow derivations of conflicting (complementary, mutual exclusive and incompatible) outcomes from the same, synonymous or consistent conditions, thus, seriously compromising the reliability and correctness of knowledge-based systems.

3.2. Classification of inconsistency

Two types of inconsistency are classified in Table 3. Each type consists of a set of patterns and each pattern encompasses different cases. Type I contains anomalous situations where rules with the same or synonymous conditions result in conflict (complementary, mutual exclusive and incompatible) conclusions. Type II captures the scenarios where a chain of deduction involves a condition and a conclusion (at two ends of the chain) which are either complementary, or mutual exclusive, or incompatible. It is very important to recognize the types of inconsistency for several reasons: (a)

[5] Facts that satisfy the validity constraints of the application domain.

[6] If there are validity constraints on facts in **WM**, then the models considered are restricted to those that satisfy the constraints.

so that effective detection algorithms can be developed; (b) the completeness of the V&V tools can be measured.

The exhaustive nature of the classification can be considered by enumerating all cases that result in an unsatisfiable RB (Definition 13). The clue is the derivation of conflict literals by a RB or a derived literal being in conflict with a fact in WM. Due to space limit, we will skip a formal proof.

3.3. Analysis

Given a RB and a WM containing a set of rules and a set of facts, respectively, we can show that the KB is consistent by trying to find a model for it. The way we try to find a model for the KB is through considering an arbitrary interpretation ζ. If ζ satisfies the KB (i.e. ζ satisfies RB and WM), then ζ is a model for it; otherwise, there is no model for the KB. If a model is found, then the KB is consistent; otherwise, it is inconsistent. We show the analysis through some examples.

Example 2. Given a KB consisting of a $\mathrm{RB} = \{r_1, r_2, r_3, r_4, r_5\}$ and a $\mathrm{WM} = \{f_1, f_2, f_3\}$ shown below

$r_1 : P \wedge Q \rightarrow A \quad f_1 : P$

$r_2 : R \wedge Q \rightarrow B \quad f_2 : Q$

$r_3 : A \wedge B \rightarrow W \quad f_3 : R$

$r_4 : A \rightarrow D$

$r_5 : B \rightarrow \neg D$

we can show that there is no model for the KB, thus, it is inconsistent.

Proof. We convert the KB into the set below

$$\Omega_1 = \{\neg P\ \neg QA, \neg R\ \neg QB, \neg A\ \neg BW, \neg AD, \neg B\ \neg D, P, Q, R\}$$

Let ζ be any interpretation for Ω_1.

- If ζ is a model for Ω_1, then $\models_\zeta P$, $\models_\zeta Q$, and $\models_\zeta R$;
- According to the first two elements in Ω_1, there must be $\models_\zeta A$ and $\models_\zeta B$;
- Since $\models_\zeta A$ and $\models_\zeta B$, there must be $\models_\zeta D$ and $\models_\zeta \neg D$ in order for $\neg AD$ and $\neg B\ \neg D$ to be *true*. But this is impossible. As a result, one of the rules of $\neg AD$ and $\neg B\ \neg D$ must be *false* under ζ.
- Since ζ cannot satisfy all rules in Ω_1, it is not a model for Ω_1. Because ζ is an arbitrary interpretation, there is no model for Ω_1. Thus, the given KB is inconsistent. □

The inconsistency in Example 2 is of type I-13 because r_4 and r_5 have different but consistent LHS and result in conflicting conclusions D and $\neg D$. The proof procedure can be automated using the *resolution principle* where the derivation of an empty clause amounts to the failure of finding a model (or the presence of inconsistency in the KB). In practice, we can use the structure of the derivation generated by the resolution principle to extract a set of inconsistent rules.

The above example demonstrates an inconsistency in the current state of a KB. There is, however, another scenario in which the proof procedure yields a model for a KB, but there exists the potential of inconsistency in a possible future state of the KB. Consider the situation where fact f_3 is a legitimate input but is not present in the WM at the time of checking, the proof procedure will find a model for $(\mathrm{KB} - f_3)$ and conclude that it is consistent. (This coincides with the intuitive explanation that the conflicting conclusion $\neg D$ is not deducible because the LHS of r_2 cannot be satisfied[7]). However, inconsistency arises when fact f_3 is asserted into WM. This phenomenon confirms our early arguments that:

- The cause of inconsistency stems from rules, but facts will help expose the inconsistency. Thus the inconsistency checking should involve both RB and WM.
- KB consistency can be either temporary or persistent. For instance, $\mathrm{KB} - f_3$ is temporarily consistent until f_3 is asserted. Such a transient consistency is not a reliable indicator. What is needed is an ultimate consistency that guarantees that a KB will be consistent for all possible states.
- The set of all legitimate facts in an application domain usually changes with time. Given a time period, it is important to identify the set of all legitimate facts during the period in order to conclude whether a KB will be persistently consistent during the period.

Operationally, when a pair of conflicting conclusions is derived, it amounts to a fact retraction in WM. In a rule-based programming language, there are two types of fact retraction: *explicit* one through a language construct such as retract and *implicit* one through derivation of a negated fact and negation as absence rule for WM. The implicit fact retraction would be an indicator for RB inconsistency, but it is not a necessary condition for RB inconsistency. The reason is that in general, a rule-based system may not have the *Church–Rosser* property,[8] therefore the derived facts by RB for the same initial facts in WM may not be unique. For instance, when both r_4 and r_5 are enabled, depending on the conflict resolution strategy used by the control component of the system, r_4 and r_5 can be fired in different order. As a result, different sets of output (derived facts) will be produced.

[7] If f_3 is not a legal input, then rule r_2 can never be enabled because of the unsatisfiability of its LHS. As a result, the rule will be picked up by the incompleteness checking and classified as an incomplete case.

[8] The Church–Rosser property of a rule-based system refers to the fact that the order in which rules are fired does not affect the final values produced [31].

Example 3. Given a KB containing the following rules and facts

$r_1 : P \wedge Q \rightarrow R \quad f_1 : P$

$r_2 : R \rightarrow W \quad f_2 : Q$

$r_3 : W \rightarrow A$

$r_4 : A \rightarrow \neg P$

we can show that there is no model for the KB, thus, the KB is inconsistent.

Proof. We convert the KB into the set $\Omega_2 = \{ \neg P\ \neg QR, \neg RW, \neg WA, \neg A\ \neg P, P, Q\}$

Let ζ be any interpretation for Ω_2.

- If ζ is a model for Ω_2, then $\models_\zeta P$ and $\models_\zeta Q$;
- There must be $\models_\zeta \neg A$, $\models_\zeta \neg W$ and $\models_\zeta \neg R$, respectively, in order for $\neg A\ \neg P$, $\neg WA$, and $\neg RW$ to be *true* under ζ;
- However, there must be $\models_\zeta R$ according to the first element in Ω_2. R and $\neg R$ cannot be both *true* under ζ. As a result, one of the clauses of $\neg P\ \neg QR$ and $\neg RW$ must be *false* under ζ.
- Since ζ cannot satisfy all rules in Ω_2, it is not a model for Ω_2. Because ζ is an arbitrary interpretation, there is no model for Ω_2. Thus, the given KB is inconsistent. □

The inconsistency in Example 3 is of type II-1 because r_1 has a condition P and results in the derivation of $\neg P$. Type II inconsistency not only introduces the logical contradiction into the inference process, it also has other pragmatic ramifications:

- In Example 3, the inconsistency involves a pair of complementary literals. When r_4 is fired, it causes P to be removed from WM, thus either preventing those rules that rely on P as input from being enabled or deactivating those rules that are enabled as a result of P.
- A list of synonymous literals and a list of mutual exclusive literals must be declared and maintained as a KB is being built and modified. In addition to Definition 6, the following should be used to maintain the validity of WM:

  If $(P \oplus Q) \wedge (Q \in WM)$, then $KB \vdash P$ would result in $(WM - \{Q\}) \cup \{P\}$.
  If $(P \not\approx Q) \wedge (Q \in WM)$, then $KB \vdash P$ would result in $(WM - \{Q\})$.

- Computationally, when $\neg P$ is a derived fact, the inference engine will check not only for the presence of P in WM, but also the presence of some literal synonymous to P.[9] Alternatively, before a derived fact P gets deposited into WM, the inference system also need to check for the presence of Q in WM that is mutually exclusive to P.

[9] Definition 6 now needs to be modified to reflect the impact of synonymous literals on the occurrence of $\neg P$ in LHS or RHS of a rule.

Though the use of synonymous and mutual exclusive literals may aid the expressive power of the language, their potential complications in system correctness should never be underestimated and their computational cost should not be ignored. Therefore, the use of those literals, especially synonymous literals, should be judicious.

4. KB redundancy

4.1. Definition of redundancy

Though redundancy may not cause logical problems (i.e. with no effect on the set of deducible literals), it may lead to following situations where potential problems may arise:

- During KB maintenance or evolution, if one of the redundant rules is modified and the others remain unchanged, then the updated KB will not correspond to the intended change, and inconsistencies can be introduced as well;
- For a KB where no certainty factors are utilized, redundant rules may be enabled under a given state, thus resulting in performance slow down because all the enabled redundant rules may be fired, even though the firings of those redundant rules will yield the same set of literals (conclusions);
- For a KB containing certainty factors, redundancy will become a serious problem, the reason being that each redundant rule may be fired, resulting in multiple countings of the same information, which, in turn, erroneously increases the level of confidence assigned to the derived literals (conclusions). This may ultimately impact the set of deducible literals.

If redundancy is introduced by design to speed up some classes of frequent deductions, then it is usually confined to a subset of the cases (e.g. types I-2, I-3, I-5 in Table 4). We can always isolate those "useful" redundant rules, and weed out redundancy from the KB where there is supposed to be none.

Definition 14. For a set **S** of rules, we define a function ψ which returns the number of distinct literals in **S**. If both L and $\neg L$ are in **S**, they will be counted as two different literals.

Definition 15. Given a set **S** of rules, if we can construct a set $\mathbf{S}'$ of rules such that $\mathbf{S} \approx \mathbf{S}'$ and

(a) either $\mathbf{S}' = \mathbf{S} - \Delta$, where $\Delta \neq \varnothing$ and $\Delta \subset \mathbf{S}$;
(b) or $\mathbf{S}' = \phi(\mathbf{S})$, where ϕ is a transformation on **S** such that $|\mathbf{S}'| = |\mathbf{S}|$ and $\psi(\mathbf{S}') < \psi(\mathbf{S})$; then there is redundancy in **S**.

Table 4
Types of redundancy

Type	Description	Pattern
I-1	Rules having the same conclusion but different permutations of the same set of conditions	$(RHS(r_i) = RHS(r_k)) \wedge (LHS(r_i) \in \rho(\mathbf{L})) \wedge (LHS(r_k) \in \rho(\mathbf{L}))$, where $\mathbf{L}$ is a set of literals
I-2	A rule r_k which can be deduced from a set of rules	$\{r_i,\ldots, r_j\} \vdash r_k$, where $(RHS(r_i) = LHS(\ldots)) \wedge \ldots \wedge (RHS(\ldots) = LHS(r_j)) \wedge (LHS(r_i) = LHS(r_k)) \wedge (RHS(r_j) = RHS(r_k))$
I-3	A rule r_i which is a specialization of another rule r_k	$(LHS(r_i) \trianglelefteq LHS(r_k)) \wedge (RHS(r_i) \trianglelefteq RHS(r_k))$, where $LHS(r_i)$ and $RHS(r_i)$ are specializations based on the same set of substitutions
I-4	A rule r_k which is subsumed by another rule	$(LHS(r_i) \subset LHS(r_k)) \wedge (RHS(r_i) = RHS(r_k))$
I-5	Generalized subsumed rule (r_k is subsumed by r_i and r_j)	$(RHS(r_i) \subset LHS(r_j)) \wedge (RHS(r_j) = RHS(r_k)) \wedge (LHS(r_k) = (LHS(r_i) \cup LHS(r_j) - RHS(r_i)))$
I-6	Rules with same condition(s) and synonymous conclusions	$(RHS(r_i) \cong RHS(r_k)) \wedge (LHS(r_i) = LHS(r_k))$
I-7	Rules with synonymous conditions and same conclusion	$(RHS(r_i) = RHS(r_k)) \wedge (LHS(r_i) \cong LHS(r_k))$
I-8	Rules with synonymous conditions and synonymous conclusion	$(RHS(r_i) \cong RHS(r_k)) \wedge (LHS(r_i) \cong LHS(r_k))$
II-1	Two rules which have the same or synonymous conclusion but contain pair(s) of conflict literals in their conditions	$((RHS(r_i) = RHS(r_j)) \vee (RHS(r_i) \cong RHS(r_j))) \wedge (LHS(r_i) = \mathbf{L} \cup \{P\}) \wedge (LHS(r_j) = \mathbf{L} \cup \{Q\})$, where $\mathbf{L}$ is set of literals and $P \Uparrow\Downarrow Q$
II-2	A rule with redundant condition(s)	$(P \in LHS(r_i)) \wedge (P' \in LHS(r_i)) \wedge ((P = P') \vee (P \cong P') \vee (P \trianglelefteq P'))$
II-3	Two rules sharing the same conclusion, and one rule having a singleton condition that is in conflict with a condition of another rule	$(RHS(r_i) = RHS(r_j)) \wedge (LHS(r_i) = \mathbf{L} \cup \{P\}) \wedge (LHS(r_j) = \{Q\})$, where $\mathbf{L}$ is set of literals and $P \Uparrow\Downarrow Q$

4.2. Commonly found types of redundancy

If either of the conditions in Definition 15 holds for a given RB, then the RB is said to contain redundancy. Thus, in essence, all types of redundancy are captured by Definition 15. However, in practice, there are sets of commonly found types of redundancy. What are included in Table 4 are the frequently encountered types of redundancy. Type I redundancy in Table 4 involves redundant rule(s) and Type II involves redundant (or unnecessary) literal(s). Each type encompasses a set of specific cases.

4.3. Analysis

Given a set $\mathbf{S}$ of rules, $\mathbf{S} \vdash \mathbf{C}$ indicates the set $\mathbf{C}$ of conclusions derivable from $\mathbf{S}$. If we can construct a set $\mathbf{S}'$ of rules from $\mathbf{S}$ such that Property (a) in Definition 15 is satisfied, we further divide $\mathbf{C}$ into $\mathbf{C}'$ and $\mathbf{C}''$ where $\mathbf{S}' \vdash \mathbf{C}'$ and $\Delta \vdash \mathbf{C}''$. We can prove that if $\mathbf{S}' \approx \mathbf{S}$, then $\mathbf{S}' \models \Delta$. According to Theorem 1, for every rule $P \in \Delta$, $\mathbf{S}' \rightarrow P$ is valid, thus $\mathbf{C}'' \subset \mathbf{C}'$ and $\mathbf{C} = \mathbf{C}'$. Therefore, rules in Δ are redundant. During the analysis process we can select a model ζ for $\mathbf{S}'$ with regard to the enabling facts and obtain $\mathbf{C}'$ from $\mathbf{S}'$, and then obtain $\mathbf{C}''$ from Δ to show $\mathbf{C}'' \subset \mathbf{C}'$.

When $\mathbf{S}'$ is constructed with Property (b) of Definition 15, the number of literals in $\mathbf{S}'$ is reduced, even though the number of rules remain the same. Similar analysis can be carried out to prove that $\mathbf{C} = \mathbf{C}'$. Since $\mathbf{S}'$ either contains fewer rules or has fewer literals, we can use $\mathbf{S}'$ to replace $\mathbf{S}$. Examples 4 and 5 are used to demonstrate the analysis process.

Example 4. Given the following set $\mathbf{S}$ of rules

r_1: $P \wedge Q \rightarrow R$
r_2: $A \wedge B \rightarrow U$
r_3: $U \wedge V \rightarrow W$
r_4: $R \wedge W \rightarrow D$
r_5: $P \wedge Q \wedge A \wedge B \wedge V \rightarrow D$

Let $\mathbf{S}' = \mathbf{S} - \{r_5\}$. We can show that $\mathbf{S}' \approx \mathbf{S}$ and r_5 is redundant.

Proof. We first convert $\mathbf{S}$ and $\mathbf{S}'$ into the abbreviated format:

$$\mathbf{S} = \{\neg P\ \neg QR, \neg A\ \neg BU, \neg U\ \neg VW, \neg R\ \neg WD, \neg P\ \neg Q\ \neg A\ \neg B\ \neg VD\}$$

$$\mathbf{S}' = \{\neg P\ \neg QR, \neg A\ \neg BU, \neg U\ \neg VW, \neg R\ \neg WD\}$$

Let ζ be an interpretation. Two situations need to be considered:

1. If $\models_\zeta \mathbf{S}$, then $\models_\zeta \mathbf{S}'$ is obvious. This is a trivial case.
2. If $\models_\zeta \mathbf{S}'$, we need to show that $\models_\zeta \mathbf{S}$ also holds. This boils down to proving that $\models_\zeta r_5$. Since $\models_\zeta r_4$ in $\mathbf{S}'$, we must have $\models_\zeta D$ or $\models_\zeta \neg R$ or $\models_\zeta \neg W$

   Case 1: If $\models_\zeta D$, then $\models_\zeta r_5$;
   Case 2: If $\models_\zeta \neg R$, then $\models_\zeta \neg P$ or $\models_\zeta \neg Q$ because $\models_\zeta r_1$. Hence, $\models_\zeta r_5$;

Case 3: If $\models_\zeta \neg W$, then $\models_\zeta \neg U$ or $\models_\zeta \neg V$ because $\models_\zeta r_3$

if $\models_\zeta \neg V$, then $\models_\zeta r_5$.
if $\models_\zeta \neg U$, then either $\models_\zeta \neg A$ or $\models_\zeta \neg B$ because $\models_\zeta r_1$.
Thus, $\models_\zeta r_5$.

Therefore, if S' is satisfied under ζ, so is **S**. $\mathbf{S}' \approx \mathbf{S}$.

If we choose a model ζ_0 for $\mathbf{S}'$ in which $\models_{\zeta 0}$ {P, Q, A, B, V}, ζ_0 is also a model for r_5. The set of derivable facts from $\mathbf{S}'$ and r_5 are $\mathbf{C}' = \{R, U, W, D\}$ and $\mathbf{C}'' = \{D\}$, respectively. Obviously, $\mathbf{C}'' \subseteq \mathbf{C}'$, therefore r_5 is redundant. □

S is of redundancy type of I-5. Removing r_5 will eliminate the redundancy.

Example 5. Given the following set **S** of rules

r_1: $P \wedge Q \wedge W \rightarrow R$
r_2: $\neg Q \rightarrow R$

Let ϕ_1 be a transformation that results in a rule r_1' by eliminating the literal Q from r_1, and let $\mathbf{S}' = \{r_1', r_2\}$. We can show that $\mathbf{S}' \approx \mathbf{S}$ and the literal Q is redundant (or unnecessary).

Proof. We first convert **S** and $\mathbf{S}'$ into the format below:

$$\mathbf{S} = \{\neg P\ \neg Q\ \neg W R, QR\}, \qquad \mathbf{S}' = \{\neg P\ \neg W R, QR\}$$

Let ζ be an interpretation. Two cases need to be considered:

1. If $\models_\zeta \mathbf{S}'$, then $\models_\zeta \mathbf{S}$ is trivial.
2. If $\models_\zeta \mathbf{S}$, we need to show that $\models_\zeta \mathbf{S}'$ also holds. This boils down to proving that whenever **S** is satisfied by ζ, $\models_\zeta r_1'$. Since $\models_\zeta r_2$ in **S**, we must have $\models_\zeta Q$ or $\models_\zeta R$

   Case 1: If $\models_\zeta R$, then $\models_\zeta r_1'$;
   Case 2: If $\models_\zeta Q$ and $\not\models_\zeta R$,[10] then $\models_\zeta \neg P$ or $\models_\zeta \neg W$ must be *true* because $\models_\zeta r_1$. Hence, $\models_\zeta r_1'$.

Therefore, $\mathbf{S}' \approx \mathbf{S}$, the literal Q in r_1 is redundant. □

S is of redundancy type of II-3. Correcting Type II redundancy involves removing the literal(s) in question. For instance, for Type II-3, when RB contains a rule set **S** matching the pattern, it can be replaced by the corresponding rule set $\mathbf{S}'$ as shown below:

S : r_1: $P1 \wedge \cdots \wedge Pk \wedge Q \rightarrow R \quad k \geq 1$
r_2: $\neg Q \rightarrow R$

S′ : r_1': $P1 \wedge \cdots \wedge Pk \rightarrow R$
r_2: $\neg Q \rightarrow R$

5. KB circularity

5.1. Definition of circularity

Circularity in a KB has been informally defined as a set of rules forming a cycle [7,24,30]. What exactly a circularity entails semantically is not that clear in the literature. In this section, we provide a definition of the KB circularity in terms of the derivation of tautologous rules and argue that the phenomenon reflects an anomalous situation in a KB and has both operational and semantic ramifications.

Definition 16. A rule E is *tautologous*, denoted as Ë, if it contains a complementary or an incompatible pair of literals.

Example 6. Following are two tautologous rules:

- $P \wedge Q \rightarrow P$, where $\neg P$ and P are a complementary pair (in $\neg P \vee \neg Q \vee P$)
- *high_priced*(*x*) ∧ *spacious*(*x*) → *expensive*(*x*), where ¬ *high_priced*(*x*) and *expensive*(*x*) are an incompatible pair (in ¬*high_priced*(*x*) ∨¬*spacious*(*x*) ∨ *expensive*(*x*)).

Definition 17. A nonempty set **S** of rules is *circular* if we can deduce a tautologous rule from **S**.

Definition 18. A nonempty set **S** of rules is *minimally circular*, denoted as $\check{\mathbf{S}}$, if **S** is circular and no proper subset of **S** is circular.

Given $\check{\mathbf{S}}$, rules in $\check{\mathbf{S}}$ are said to be forming a cycle. The deduction of a tautologous rule is trivial if $\check{\mathbf{S}}$ is a singleton set satisfying the aforementioned condition. In a given $\check{\mathbf{S}}$, there may be more than one tautologous rule deducible from it that involves different pairs of (complementary or incompatible) literals.

Operationally speaking, circular rules may result in infinite loops (if an exiting condition is not properly defined) during inference, thus hampering the problem solving process. Semantically speaking, the fact that a tautologous

[10] $\not\models_\zeta R$ indicates that R evaluates to false under ζ.

wff is derivable indicates that the circular rule set encompasses knowledge that is always true regardless of any problem specific information. In general, tautologous wffs are those that are true by virtue of their logical form and thus provide no useful information about the domain being described [47]. Therefore, circular rules prove to be less useful in the problem solving process. What is needed, as evidenced in many real KB systems, are consistent rules that are triggered by problem specific information (facts) rather than tautologous rules that are true regardless of the problem to be solved.

5.2. Types of circularity

Circularity primarily stems from the definitions of rules in RB. However, control strategies deployed (in places such as the mechanisms of agendas, rule salience or priority level definitions and module selections) in the inference system may also be cause for the infinite looping of certain rules. In this paper, we focus on the types of circularity that are confined in the RB.

Definition 19. Given a minimally circular rule set $\check{\mathbf{S}}$, we define two sets of literals $\check{\mathbf{S}}_L$ and $\check{\mathbf{S}}_R$ as follows:

$$\check{\mathbf{S}}_L = \{L | L \in \mathrm{LHS}(r) \wedge r \in \check{\mathbf{S}}\}$$

$$\check{\mathbf{S}}_R = \{L | L \in \mathrm{RHS}(r) \wedge r \in \check{\mathbf{S}}\}.$$

The types of circularity in a rule base, as summarized in Table 5, are classified based on enumerating possible relationships between $\check{\mathbf{S}}_L$ and $\check{\mathbf{S}}_R$ and the nature of the tautology. Type I circularity indicates cycles in which $\check{\mathbf{S}}_L = \check{\mathbf{S}}_R$. Type II describes cycles with additional conditions involved in the rules, therefore, $\check{\mathbf{S}}_R$ is a proper subset of $\check{\mathbf{S}}_L$. If $\mathbf{C}_{\check{S}}$ is a cycle formed out of a minimally circular rule set $\check{\mathbf{S}}$, the *girth g* of $\mathbf{C}_{\check{S}}$ can be defined as $g(\mathbf{C}_{\check{S}}) = |\check{\mathbf{S}}|$. Cycles in these types can have a girth ranging from one to some integer MAX where MAX is bounded by the cardinality of the rule base $|\mathrm{RB}|$ of a given KB.

5.3. Analysis

The analysis of KB circularity amounts to deriving from a given rule base a tautologous rule r that satisfies the conditions in Definition 16, using some inference method.

Example 7. Below is a rule base **S** containing five rules

r_1: $W \rightarrow U$
r_2: $P \wedge A \rightarrow R$
r_3: $Q \wedge C \rightarrow W$
r_4: $R \wedge B \rightarrow Q$
r_5: $U \wedge D \wedge E \wedge G \rightarrow P$

Using the *resolution* method, we can derive a tautologous rule from **S**. Since **S** is the smallest set that yields such a tautologous rule, it is thus minimally circular.

Proof. We convert **S** into the following format

$$\mathbf{S} = \{\neg W U, \neg P \neg A R, \neg Q \neg C W, \neg R \neg B Q, \neg U \neg D \neg E \neg G P\}.$$

It is not difficult to see that the following rule is derivable from **S** by using the resolution method

$$\neg W \neg D \neg E \neg G \neg A \neg B \neg C W.$$

Since $\neg W$ and W are a pair of complementary literals, the derived rule is tautologous. Therefore, **S** is minimally circular. □

Incidentally, there are four other tautologous rules involving $\neg P$ and P, $\neg Q$ and Q, $\neg R$ and R, and $\neg U$ and U, respectively. This example exhibits Type II-1 circularity.

Once a circularity is detected, the circular rule set needs to be syntactically redefined to break up the circularity. Semantically, information about a problem domain needs to be reorganized so that it will contribute to the problem solving process. Some of the possible remedial measures for circularity can be found in Section 7.

6. KB incompleteness

Informally speaking, a KB is incomplete when it does not have all the necessary information to answer a question of interest in an intended application [16,31]. Thus, completeness represents a query-centric measure for the quality of a KB. KB incompleteness is a real issue to be reckoned with for at least the following reasons: (a) In many applications, the KB is built in an incremental and piecemeal fashion and it undergoes a continual evolution. The information acquired at each stage of the evolution may be vague or indefinite in nature. (b) The deployment of a KB system cannot just wait for the KB to be stabilized in some final and complete form since this may never happen.

Despite the fact that a practical KB can never completely capture all aspects of a real problem domain, it is still possible for a KB to be complete for a specific area in the domain. The boundaries of this specific area may be defined in terms of all *relevant queries* to be asked during problem solving process. If a KB has all the information to answer those relevant queries *definitely*, then the KB is complete with regard

Table 5
Types of circularity in a rule base

Type	Description	Pattern
I-1	$\check{\mathbf{S}}_L = \check{\mathbf{S}}_R$ for $\check{\mathbf{S}}$ and tautologous rule involves complementary pair of literals	$(\check{\mathbf{S}}_L = \check{\mathbf{S}}_R) \wedge (\check{\mathbf{S}} \vdash \ddot{\mathbf{E}}) \wedge (L, \neg L \in \ddot{\mathbf{E}}) \wedge (L \# \neg L)$
I-2	$\check{\mathbf{S}}_L = \check{\mathbf{S}}_R$ for $\check{\mathbf{S}}$ and tautologous rule involves pair of incompatible literals	$(\check{\mathbf{S}}_L = \check{\mathbf{S}}_R) \wedge (\check{\mathbf{S}} \vdash \ddot{\mathbf{E}}) \wedge (L, \neg L \in \ddot{\mathbf{E}}) \wedge (L \cong \neg L)$
II-1	$\check{\mathbf{S}}_R \subset \check{\mathbf{S}}_L$ for $\check{\mathbf{S}}$ and tautologous rule involves complementary pair of literals	$(\check{\mathbf{S}}_R \subset \check{\mathbf{S}}_L) \wedge (\check{\mathbf{S}} \vdash \ddot{\mathbf{E}}) \wedge (L, \neg L \in \ddot{\mathbf{E}}) \wedge (L \# \neg L)$
II-2	$\check{\mathbf{S}}_R \subset \check{\mathbf{S}}_L$ for $\check{\mathbf{S}}$ and tautologous rule involves pair of incompatible literals	$(\check{\mathbf{S}}_R \subset \check{\mathbf{S}}_L) \wedge (\check{\mathbf{S}} \vdash \ddot{\mathbf{E}}) \wedge (L, \neg L \in \ddot{\mathbf{E}}) \wedge (L \cong \neg L)$

to those queries. In what follows, we base our discussions of completeness on the concepts of relevant queries and the ability of a KB to answer those queries.

6.1. Definition of query-based incompleteness

Definition 20. Given a KB, we define $\mathbb{P}_{KB}$ and $\mathbb{P}_A$ as sets of all predicate symbols and *askable* predicate symbols in the KB, respectively. An askable predicate symbol is one that can appear in a query. Usually it is the case that $\mathbb{P}_{KB} \supseteq \mathbb{P}_A$.[11] A query $\bar{Q}$ containing predicate symbols $p_i, \ldots, p_j \in \mathbb{P}_A$ is denoted as

$$\bar{Q} \simeq Q(p_i, \ldots, p_j)$$ [12]

Definition 21. A set $\mathbb{Q}$ of *relevant queries* is now defined as follows:

$$\mathbb{Q} = \{\bar{Q} | \bar{Q} \text{ appears in some query session } \wedge \; \bar{Q} \simeq Q(p_i, \ldots, p_j) \wedge p_i, \ldots, p_j \in \mathbb{P}_A\}.$$

Definition 22. Given a query $\bar{Q} \in \mathbb{Q}$, the answer to $\bar{Q}$, denoted as $\alpha(\bar{Q})$, can be either *definite* or *unknown*. $\alpha(\bar{Q})$ is definite if either $KB \vdash \bar{Q}$ or $KB \vdash \neg\bar{Q}$; $\alpha(\bar{Q})$ is unknown if neither $KB \vdash \bar{Q}$ nor $KB \vdash \neg\bar{Q}$.

Definition 23. A KB is *complete* with regard to a relevant query set $\mathbb{Q}$ if $\forall \bar{Q} \in \mathbb{Q}$ [$\alpha(\bar{Q})$ is definite].

6.2. Types of incompleteness

Let $\mathbb{P} = \mathbb{P}_{KB} \cup \mathbb{P}_A$. For a predicate symbol $p \in \mathbb{P}$, we introduce a set of predicate symbols $\Re(p)$ on which p directly or indirectly depends. $\Re(p)$ can be obtained using the following procedure.

INPUT: $p \in \mathbb{P}$
OUTPUT: $\Re(p)$
$\Re(p) := \varnothing$;
while $\exists r \in KB \quad [p \in RHS(r)]$ ***do*** $\Re(p) := \Re(p) \cup LHS(r)$;
while $\exists r \in KB \quad \exists q \in \mathbb{P}_{KB} \quad [q \in RHS(r) \wedge q \in \Re(p) \wedge LHS(r) \not\subseteq \Re(p)]$ ***do*** $\Re(p) := \Re(p) \cup LHS(r)$;

If a literal containing a predicate symbol p cannot be satisfied by either a given fact or a derived fact, then it is denoted as $\nvdash p$. Three types of incompleteness are defined in Table 6. Types I and II reveal KB incompleteness from the perspective of relevant queries, i.e., lack of necessary information to answer queries, and Type III indicates the potential incompleteness of the relevant query set $\mathbb{Q}$ from the perspective of known information (rules/facts).

Though the classification in Table 6 is exhaustive with regard to Definition 23, there are pragmatic and application specific considerations that will help determine the validity of incompleteness cases.

6.3. Analysis

The analysis of KB incompleteness depends critically on the availability of information regarding the relevant query set in a problem domain. Prototyping often serves as a means to ascertain the relevant query set. If the relevant query set is available, the analysis amounts to finding out if all queries can be answered definitely. Checking for the presence or absence of the aforementioned syntactic symptoms is an integral and necessary part of the analysis process. However, there are other considerations in the analysis process that are semantic, pragmatic, or problem specific. The analysis process is really an iterative one, because as KB continually evolves, so will the relevant query set.

[11] When there is incompleteness in a KB, this may not be true, as evidenced in Table 6.

[12] We assume that the query $\bar{Q}$ is a conjunction of the literals containing predicate symbols $P_i, \ldots, P_j$.

Table 6
Types of incompleteness

Type	Descriptions [7,24,37]	Pattern
I	Dangling conditions, unreachable conclusions	$\exists q \in \mathbb{P}\ \exists p \in \mathbb{P}_A[q \in \Re(p) \wedge \nvdash q]$[a]
II	Missing initial facts, missing rules	$\exists p \in \mathbb{P}_A\ [\Re(p) = \varnothing \wedge p \notin \mathbb{P}_{KB}]$
III	Useless conclusions, unused initial facts, isolated rules	$\exists q \in \mathbb{P}_{KB}\ \forall p \in \mathbb{P}_A\ [q \notin \Re(p)]$

[a] Because the criterion for the completeness issue is domain-specific, it is possible that q in $[q \in \Re(p) \wedge \nvdash q]$ may be useless structure in the KB. Ultimately, the domain expert or knowledge engineer has to determine the nature of the anomaly.

Example 8. For the following KB,

$r_1 : h(x,y) \wedge r(y,z) \rightarrow p_1(x,z)$ $\quad f_1 : m(d)$

$r_2 : w(y) \wedge u(x) \rightarrow r(x,y)$ $\quad f_2 : v(a)$

$r_3 : v(x) \rightarrow w(x)$ $\quad f_3 : u(b)$

$r_4 : m(x) \rightarrow p_3(x)$ $\quad f_4 : u(c)$

we have

$\mathbb{P}_A = \{p_1, p_2\}$
$\mathbb{P}_{KB} = \{p_1, p_3, r, u, v, w, h, m\}$
$\Re(p_1) = \{h, r, u, v, w\}$
$\Re(p_2) = \varnothing$.

Since $p_2 \in \mathbb{P}_A$ and $[\Re(p_2) = \varnothing \wedge p_2 \notin \mathbb{P}_{KB}]$, there exists Type II incompleteness. No rules and facts could be used to answer queries involving p_2. In addition, $h \in \Re(p_1)$ and $\nvdash h$. So Type I incompleteness also exists. Finally, the presence of the rule r_4 and the fact f_1 may indicate that p_3 should have been an askable predicate. In other words, $\mathbb{P}_A$ is incomplete, and there is reason to believe that the relevant query set is incomplete also. □

7. Remedial measures

Once KB anomalies are identified, the next issue is how to correct the situations in which the quality of a KB has been compromised. Though it is of pivotal importance, the issue has not been adequately addressed in the literature. To a certain extent, this is due to the fact that the issue of how to mend a KB relies on a whole host of considerations, many of which are problem or application specific. In the rest of this section, we would like to address the issue in terms of some general principles and provide some example remedial measures for the cases dealt with in the previous four sections.

For correcting inconsistency, we suggest the following actions:

- Avoid using synonymous literals if possible.
- Delete one of the offending rules that derives the conflict conclusion.
- Modify the conditions (e.g. predicate symbols) of the rules involved such that they no longer have or share the same or synonymous conditions.
- Modify the conclusions (e.g. predicate symbols) of the rules involved such that they are no longer in conflict.
- Move one of the offending rules to a different rule module such that the derivation of conflict conclusions cannot take place in the same problem-solving session or at the same time.

Actions to eliminate redundancy may include:

- Delete redundant rule(s).
- Merge or collapse rules into one.

  For example, $P \wedge Q \rightarrow R$, $\neg P \wedge Q \rightarrow R \Rightarrow Q \rightarrow R$

- Delete condition(s) of certain rule(s).

  For example, $P \wedge Q \rightarrow R$, $\neg Q \rightarrow R \Rightarrow P \rightarrow R$, $\neg Q \rightarrow R$

- Modify the conditions or conclusions of the redundant rules such that they no longer are the same or synonymous.

To resolve circularity, the following remedial measures may be taken:

- Remove a rule from a circular rule set.

  For example, $P \rightarrow Q$, $Q \rightarrow R$, $R \rightarrow P \Rightarrow P \rightarrow Q$, $Q \rightarrow R$

- Redefine a conclusion of a rule in the set such that it no longer serves as a condition of another rule in the set.

  For example, $P \rightarrow Q$, $Q \rightarrow R$, $R \rightarrow P \Rightarrow P \rightarrow Q$, $Q \rightarrow R'$, $R \rightarrow P$ where R' and R are no longer unifiable.

- Redefine a condition of a rule in the set such that it no longer matches a conclusion of another rule in the set.

To plug holes in an incomplete KB, we could

- Add new rules and/or facts to make all relevant queries definite.

  For example, new rules and facts can be added to make $h(x, y)$ satisfiable in Example 8.

- Modify the initial facts to patch up holes.
- Modify the conditions and/or conclusions of rules involved in an incompleteness case so that they will be "connected" with the rest of RB.

Though it is beyond the scope of this paper, we would like to point out that in a KB where certainty factors (CF) are used, there are additional actions to be considered. For instance, add or modify CF values for rules or facts, or

modify the threshold value(s) for the CF-value propagation during inference process.

8. Concluding remarks

As more and more expert systems and knowledge-based systems are deployed in settings where failures may result in loss of productivity, decision-making quality, property, business, services, investment, or even life, ways to detect and resolve potential anomalies in a KB become critical issues in developing correct, accurate and reliable systems. In order for the results to be credible, V&V techniques must be built on a solid theoretical foundation.

It is difficult to assess many of the V&V tools, methods and techniques that have been developed or proposed because there is no accepted standard against which to measure the reliability or correctness of an expert system. Indeed there is lack of definite semantics for expert systems in general and KB in particular. This prevents any definite conclusions about reliability and hinders the use of expert systems in safety-critical applications. The field of V&V for expert systems is far from having tractable formal models that can cover all of the features of real expert systems, which often rely on imperative state changes and other non-logical features. Our simplified model, though a preliminary one, does provide a basis for reaching definite conclusions about the reliability of those aspects in expert systems that can be expressed in logical terms. It is our hope that the logical formulation presented in this paper makes a step in the right direction.

Future work can continue in several directions. One is concerned with how to establish an assessment standard, based on logical instruments similar to those discussed in this paper, for the V&V tools and methodologies. For instance, given a KB and its semantics Γ, we use $\tilde{A}_\Gamma$ to indicate the set of anomalies defined under Γ. For a V&V method M, we use $\tilde{A}_M$ to denote the set of anomalies M is capable of discovering. M is *sound* if $\forall \tilde{a} \in \tilde{A}_M\ [\tilde{a} \in \tilde{A}_\Gamma]$; M is *complete* if $\forall \tilde{a} \in \tilde{A}_\Gamma\ [\tilde{a} \in \tilde{A}_M]$.

Another direction is to study the KB anomalies in an object-oriented (OO) paradigm. Recent developments in knowledge representation formalisms include: (a) extending the OO paradigm to include rules (i.e. rules can be considered as a specific type of behavior for objects); (b) bringing objects into the rule-based paradigm (i.e. rules are specified about objects); (c) hybrid representation formalism that blends frames, objects, cases and rules together [48]. The next challenge to us is the issue of how to (re)define the concepts and meanings of KB anomalies in the context of those formalisms.

A KB should be developed based on its underlying ontology [49,50]. It is not clear what relationship there is between the anonymous situations that are manifested at a KB level and the root causes at its ontology. This is yet another direction worth exploring.

Acknowledgements

The authors would like to express their sincere appreciation to Prof. V. Berzins, and the anonymous referees for their helpful comments and suggestions on the earlier drafts of the paper.

References

[1] M. Ayel, J.P. Laurent (Eds.), Validation, Verification and Test of Knowledge-Based Systems Wiley, Chichester, UK, 1991.
[2] U.G. Gupta (Ed.), Validating and Verifying Knowledge-Based Systems IEEE Computer Society Press, Los Alamitos, CA, 1991.
[3] Verification and validation of knowledge-based systems [Special issue], C. Culbert (Ed.), Expert Systems with Applications 1 (3) (1990).
[4] Verification and validation of intelligent systems: five years of AAAI workshops [Special issue], D.E. O'Leary (Ed.), International Journal of Intelligent Systems 9 (8/9) (1994).
[5] E. Plaza, Validation and verification of knowledge-based systems, IEEE Expert 8 (3) (1993) 45–81.
[6] Verification and validation of knowledge-based systems, A. Preece, C. Suen (Eds.), International Journal of Expert Systems 6 (2/3) (1993) (special issue).
[7] C.L. Chang, J.B. Combs, R.A. Stachowitz, A report on the expert systems validation associate (EVA), Expert Systems with Applications 1 (1990) 217–230.
[8] B.J. Cragun, H.J. Steudel, A decision-table-based processor for checking completeness and consistency in rule-based expert systems, International Journal of Man–Machine Studies 26 (1987) 633–648.
[9] R.F. Gamble, G.C. Roman, W.E. Ball, Formal verification of pure production system programs, Proceedings of Ninth National Conference on AI, 1991, pp. 329–334.
[10] A. Ginsberg, Knowledge-base reduction: a new approach to checking knowledge bases for inconsistency and redundancy, Proceedings of Seventh National Conference on AI, 1998, pp. 585–589.
[11] A. Ginsberg, K. Williamson, Inconsistency and rdundancy checking for quasi-first-order-logic knowledge bases, International Journal of Expert Systems 6 (3) (1993) 321–340.
[12] D. Hamilton, K. Kelley, C. Culbert, State-of-the-practice in knowledge-based system verification and validation, Expert Systems with Applications 3 (1991) 403–410.
[13] M. Jafar, A.T. Bahill, Interactive verification of knowledge-based systems, IEEE Expert 8 (1) (1993) 25–32.
[14] J.D. Kiper, Structural testing of rule-based expert systems, ACM Transactions on Software Engineering and Methodology 1 (2) (1992) 168–187.
[15] S. Kirani, I.A. Zaulkernan, W.T. Tsai, Evaluation of expert system testing methods, Communications of ACM 37 (11) (1994) 71–81.
[16] H.J. Levesque, The logic of incomplete knowledge bases, in: M.L. Brodie, J. Mylopoulos, J.W. Schmidt (Eds.), On Conceptual Modeling, Springer, New York, 1984.
[17] N.K. Liu, T. Dillon, An approach towards the verification of expert systems using numerical petri nets, International Journal of Intelligent Systems 6 (1991) 255–276.
[18] Luqi, D. Cooke, How to combine nonmonotonic logic and rapid prototyping to help maintain software, International Journal of Software Engineering and Knowledge Engineering (1999).
[19] Luqi, Knowledge-based support for rapid software prototyping, IEEE Expert 3 (4) (1988) 9–18.
[20] Luqi, Rapid prototyping languages and expert systems, IEEE Expert 4 (2) (1989) 2–5.
[21] L.A. Miller, Recommended guidelines for V&V of cases, AAAI-94

Workshop on Validation and Verification of Knowledge-Based Systems, 1994, pp. 1–9.
[22] T.J. Murray, M.R. Tanniru, Control of inconsistency and redundancy in prolog-type knowledge bases, Expert Systems with Applications 2 (1991) 321–331.
[23] D.L. Nazareth, Investigating the applicability of petri nets for rule-based system verification, IEEE Transactions on Knowledge and Data Engineering 5 (3) (1993) 402–415.
[24] T.A. Nguyen, W.A. Perkins, T.J. Laffey, D. Pecora, Knowledge base verification, AI Magazine 8 (1987) 69–75.
[25] H. Nonfjall, H.L. Larsen, Detection of potential inconsistencies in knowledge bases, International Journal of Intelligent Systems 7 (1992) 81–96.
[26] R.M. O'Keefe, D.E. O'Leary, Expert system verification and validation: a survey and tutorial, Artificial Intelligence Review 7 (1993) 3–42.
[27] R. Plant, S. Murrell, On the validation and verification of production systems: a graph reduction approach, AAAI-94 Workshop on Validation and Verification of Knowledge-Based Systems, 1994, pp. 56–63.
[28] A. Preece, R. Shinghal, A. Batarekh, Verifying expert systems: a logical framework and a practical tool, Expert Systems with Applications 5 (2/3) (1992) 421–436.
[29] M.C. Rousset, On the consistency of knowledge bases: the COVADIS system, Proceedings of Eighth European Conference on AI, 1988, pp. 79–84.
[30] J. Rushby, Quality Measures and Assurance for AI Software, NASA Contractor Report 4187, October 1988.
[31] J. Rushby, R.A. Whitehurst, Formal Verification of AI Software, NASA Contractor Report 181827, February 1989.
[32] M. Suwa, A.C. Scott, E.H. Shortliffe, An approach to verifying completeness and consistency in a rule-based expert system, AI Magazine 3 (1982) 16–21.
[33] W.T. Tsai, I.A. Zualkernan, S. Kirani, Pragmatic testing methods for expert systems, International Journal of AI Tools 2 (2) (1993) 181–217.
[34] Verifying Knowledge Bases: A Bibliography, Knowledge Engineering Review 7 (2) (1992) 143–146.
[35] R.J. Waldinger, M. Stickel, Proving properties of rule-based systems, International Journal of Software Engineering and Knowledge Engineering 2 (1) (1992) 121–144.
[36] D. Zhang, Luqi, Formal analysis of inconsistency and redundancy in knowledge bases, IJCAI-95 Workshop on Verification and Validation of Knowledge-Based Systems, pp. 110–116.
[37] D. Zhang, D. Nguyen, PREPARE: a tool for knowledge base verification, IEEE Transactions on Knowledge and Data Engineering 6 (6) (1994) 983–989.
[38] D. Zhang, Perspectives in knowledge base verification, Proceedings of Fifth International Conference on Software Engineering and Knowledge Engineering, 1993, pp. 123–456.
[39] N. Zlatareva, A framework for verification, validation, and refinement of knowledge bases: the VVR system, International Journal of Intelligent Systems 9 (1994) 703–737.
[40] N. Zlatareva, A. Preece, State of the art in automated validation of knowledge-based systems, Expert Systems with Applications 7 (2) (1994) 151–167.
[41] R. de Hoog, et al., The CommonKADS organization model: content, usage, and computer support, Expert Systems with Applications 11 (1) (1996) 247–260.
[42] G. Guida, C. Tasso, Design and Development of Knowledge-Based Systems: From Life Cycle to Methodology, Wiley, Chichester, UK, 1994.
[43] G. Schreiber, et al., CommonKADS: a comprehensive methodology for KBS development, IEEE Expert 9 (6) (1994) 28–37.
[44] D.S.W. Tansley, C.C. Hayball, Knowledge-Based Systems Analysis and Design: A KADS Developer's Handbook, Prentice Hall, UK, 1993.
[45] M. Ben-Ari, Mathematical Logic for Computer Science, Prentice Hall, UK, 1993.
[46] C.L. Chang, R.C.T. Lee, Symbolic Logic and Mechanical Theorem Proving, Academic Press, New York, 1973.
[47] M.R. Genesereth, N.J. Nilsson, Logical Foundations of Artificial Intelligence, Morgan Kaufmann Publishers, Los Altos, CA, 1987.
[48] K.W. Tracy, P. Bouthhoorn, Object-Oriented Artificial Intelligence Using C++, Computer Science Press, New York, 1997.
[49] B. Chandrasekaran, J.R. Josephson, V.R. Benjamins, What are ontologies, and why do we need them? IEEE Intelligent Systems 14 (1) (1999) 20–26.
[50] D.E. O'Leary, Using AI in knowledge management: knowledge bases and ontologies, IEEE Intelligent Systems 13 (3) (1998) 34–39.