A State-based Knowledge Representation Approach for Information Logical Inconsistency Detection in Warning Systems

Jun Ma, Guangquan Zhang, Jie Lu

Faculty of Engineering and Information Technology, University of Technology, Sydney (UTS)
P.O. Box 123, Broadway, NSW 2007, Australia
{*junm, zhangg, jielu*}*@it.uts.edu.au*

Abstract

Detecting logical inconsistency in collected information is a vital function when deploying a knowledge-based warning system to monitor a specific application domain for the reason that logical inconsistency is often hidden from seemingly consistent information and may lead to unexpected results. Existing logical inconsistency detection methods usually focus on information stored in a knowledge base by using a well-defined general purpose knowledge representation approach, and therefore cannot fulfill the demands of a domain-specific situation. This paper first proposes a state-based knowledge representation approach, in which domain-specific knowledge is expressed by combinations of the relevant objects' states. Based on this approach, a method for information logical inconsistency detection (ILID) is developed which can flexibly handle the demands of various domain-specific situations through reducing some restrictions in existing methods. Finally, two real-case based examples are presented to illustrate the ILID method and its advantages.

Key words: Knowledge representation, Knowledge base verification, Decision support systems, Warning systems

1. Introduction

Emergency management and warning systems historically have focused on the immediate and urgent aspects of disasters such as prediction, response and post-disaster recovery. Currently, practices have brought growing awareness of people-centered warning system frameworks for information integration and emergency response [2]. Developing a decision model for a people-centered warning system requires effectively analysis, integration, and utilization of information collected from various sources, because the cost of decision-making errors in a warning system can be very large. Due to the continuous changes in the application environment and higher uncertainty in information sources and information itself, keeping information consistency is an essential and challenged issue for deploying a warning system.

Information inconsistency is ubiquitous, which may lead to conflicts and cause difficulty in decision making [14]. Two fundamental forms of information inconsistency are generally encountered, i.e., data inconsistency and logical inconsistency. Data inconsistency is usually manifested in errors or incorrectness of certain of facts, such as the assertion of "Sydney, the capital of Australia," which often exists in a single assertion or statement. Logical inconsistency is not easily recognized in an isolated fact or assertion; however, it can be disclosed through paradoxes resulted from several seemingly correct facts. For instance, the two pieces of information that "a veteran of World War I died in 2006" and "the veteran was aged 95 when he died" will deduce an absurd conclusion that "the veteran took part in World War I when he was an infant." This is a typical case of logical inconsistency in information in a real application. In this paper, we mainly focus

on the detection of logical inconsistency.

Logical inconsistency in information arises for various reasons [14] such as the topicality of information gathering, the technique of information collecting, and the distribution of information sources. Studies of logical inconsistency are often conducted on the syntactic level and the semantic level [14]. On the syntactic level, each piece of information is treated as a logical formula, and thus logical inconsistency is described as there being no interpretation model for a set of formulae. On the semantic level, each piece of information is linked to a concrete context and is embedded with some facts; therefore, the logical inconsistency in a set of information is recognized when paradoxes are inferred from given facts. Corresponding to these two study levels, detection methods such as using fuzzy sets, matrix, binary diagrams, as well as unification are presented [4, 10, 12, 15, 23, 24, 25].

Information logical inconsistency deduction methods are widely used to deal with logical inconsistency at the syntactic level [1, 7, 10, 19, 23, 25]. These methods are mainly based on designed logic systems and their reasoning mechanisms. The primary procedure of logical inconsistency detection is implemented through logical reasoning. Hence, they are reasoning-based. For example, Hunter [8, 9] used weakly-negative logic, four-valued logic, quasi-logical logic, to implement logical inconsistency detection. Since a logical inconsistency at the syntactic level is derived when there is no model for a set of formulae which are used to represent information or knowledge, resolution strategies for the satisfiability problem [6, 21] are introduced to check existence of possible model. For instance, Polat [19] applied a unification strategy for logical inconsistency; and Mazure et al. [10] used the bounded resolution technique and the local searching method for inconsistency in non-monotonic knowledge bases. Zhang et al. [25] presented a set of analysis models for describing and detecting inconsistency, redundancy, circularity, and incompleteness by using the first-order predicate logic.

Information logical inconsistency detection methods at the semantic level are mainly developed on the basis of well-defined graphs, such as Petri nets [24], binary directed graph [12], and their extensions. These methods take the objects and their relationships in a real world into account and disclose inconsistency through searching conflict cases along possible paths in a graph. Hence these methods are graph-based. For example, Park and Seong [15] reported a knowledge base detecting method based on extended colored Petri nets and used this method in nuclear power plant dynamic alarms analysis. Yang et al. [24] also proposed a high-level Petri nets formalization model for detecting inconsistency in rule bases, in which each rule is represented by a Horn clause. Furthermore, Botten [4] used a matrix to describe the rules in a knowledge base, which is similar to the incident matrix in Petri nets theory. Similarly, Mues et al. [12, 13] developed a logical consistency detection method by applying binary decision diagrams.

However, both the reasoning-based and graph-based methods have drawbacks when applied to logical inconsistency detection in real applications such as warning systems. Firstly, these methods mainly focus on logical inconsistency in knowledge [4, 10, 12, 22, 23] stored in a knowledge-base and very few of them are able to apply for real-time information which is most concerned in real applications. Secondly, these methods lack the ability to identify and classify models which may never occur from those exising in theory. Therefore, they deal with all possible cases no matter whether they are meaningful or not in a given context. For example, suppose $\{A \to B,\ B \to C,\ C \to \neg A\}$ is a set of information. Obviously, $A = 0$, $B = 1$ and $C = 1$ is a model for these three rules in the conventional two-valued first-order logic. However, $A \to \neg A$ as a logical consequence of $A \to B$, $B \to C$, and $C \to \neg A$, can be deduced without difficulties. Unfortunately, this conclusion cannot be accepted in a real situation. Hence, the cost to check all possible models is time-consuming and

uneffective.

The main reasons for these drawbacks are: (1) current methods are usually based on well-defined knowledge representation techniques for general purpose knowledge rather than for domain-specific knowledge; (2) they take two-valued first-order logic as the main logic basis of them which is unsuitable for a domain-specific situation where the underlying logic is often of multiple-valued features; (3) these methods mainly focus on stored knowledge rather than real-time information which is the case for warning systems. Stored knowledge is static information, while real-time information is dynamic information. In real applications, both static and dynamic information are needed [19] when applying an application system in an uncertain and changeable environment where the processing for dynamic information is more crucial than that for static information.

Literature has shown that domain-specific knowledge and information can be represented by a set of objects, their states, and their classifications [11, 16, 17, 18]. Therefore, with regard to the characteristic of knowledge processed in a warning system, this paper proposes a state-based domain knowledge representation approach and then applies it to detect logical inconsistency for real-time information. A method for information logical inconsistency detection (ILID) is then proposed, which can efficiently deal with domain-specific knowledge and information in warning or other information systems. The rest of the paper is organized as follows. The approach for representing domain knowledge by the states of objects is proposed in Section 2. Section 3 presents the ILID method which uses the proposed knowledge representation approach to detect information logical inconsistency in the real-time information. Two case-based examples illustrate the application of the ILID method in Section 4. Finally, our future study is discussed in Section 5.

2. State-based knowledge representation and consistency

This section will present a state-based knowledge representation approach.

2.1. State-based knowledge representation

Usually, domain-specific knowledge has close relation with pertinent objects. First of all, a piece of domain-specific knowledge can be recognized through the special states of relevant objects. Secondly, domain-specific knowledge may vary when states of those objects change. Hence, states of objects are used to establish knowledge representation approach in this section.

An object may have lots of states in different domains. However, it is only in a finite normal states in a specific domain. These normal states of different objects often form some regular combinations in a stable condition. These state combinations are formed at two levels. At the first level, a combination includes the states of different objects, which reflects the interaction between different objects. At the second level, a combination includes the multiple states of the same object which reflects the linkage between a piece of knowledge and an object's states. People's domain-specific knowledge may be established on those regular combinations. Based on this consideration, we treat domain-specific knowledge as combinations of states of related objects.

Suppose D is a domain, the knowledge in D is related to a set of objects denoted by $O_1, O_2, \ldots, O_n$. Each object O_i has several possible states $s_1^{(i)}$, $s_2^{(i)}$, ..., $s_{i_m}^{(i)}$ (denoted by S_i) in domain D. At a given time t ($t \in T$), the normal states of the object O_i are denoted by $S_i(t)$ ($S_i(t) \subseteq S_i$), $i = 1, 2, \ldots, n$. These normal states can be obtained from many sources such as historical records and relevant theories. In the following, each $S_i(t)$ is supposed to be a non-empty set.

Remark 2.1. *It is difficult to clearly assert which state of an object is in some situations although an object should*

be in a unique state in a given time t by intuition. In those situations, people always use uncertain expression to depict knowledge. For instance, people often say "a young person is more energetic than an old person" where both "young" and "old" are often used to express possible ages of a person but they are not corresponding to any particular value, e.g. 25 or 55. Without other specification, the states of objects in this paper are assumed to be distinguishable. Thus an object taking state s_1 is definitely different from that it taking state s_2.

Before given the definition of domain-specific knowledge, we first introduce the notion of unordered n-tuples. An unordered n-tuples, here, means a set of states of objects such that each element belongs to an individual object's state set. Formally, let X_1, X_2, ..., X_n be n non-empty sets. That $(x_1, x_2, \ldots, x_n)$ is an unordered n-tuples means $x_i \in X_i$ and $x_i \notin X_j$ if $i \neq j$, $i, j = 1, 2, \ldots, n$. For convenience, we use $X_1 \otimes X_2 \otimes \cdots \otimes X_n$ to denote the set of all unordered n-tuples obtained from X_1, ..., X_n.

Definition 2.1. *A piece of domain-specific knowledge $\omega(t)$ of D at time t is composed of a set of unordered p-tuples, i.e.,*

$$\omega(t) \subseteq \bigotimes_{j \in J(\omega(t))} S_j, \tag{1}$$

where $J(\omega(t))$ $(\subseteq \{1, 2, \ldots, n\})$ is the index set of relevant objects of the knowledge $\omega(t)$, and $p = |J(\omega(t))|$.

In the following, we use "knowledge" representing "domain-specific knowledge" without other specification and use $S_j^{(\omega(t))}$ to denote the states of object O_i with respect to the knowledge $\omega(t)$. A piece of knowledge $\omega(t)$ is called empty knowledge if for some $j \in J(\omega(t))$, $S_j^{(\omega(t))}$ is an empty set. Empty knowledge indicates a kind of impossible state combination. Thus empty knowledge is not unique.

Definition 2.1 can be used to explain some typical relationships between objects. In general, there are three kinds of relationships between two objects, i.e., $A \Rightarrow B$, $B \Rightarrow A$, and $A \Leftrightarrow B$. When $A \Rightarrow B$, this means from a given state of A, some states of B can be obtained. This relationship can be expressed by a piece of knowledge composed 2-tuples

$$\omega = \{(a, b) | b \in S_B\}. \tag{2}$$

Similarly, relation $B \Rightarrow A$ can be expressed by a piece of knowledge

$$\omega = \{(a, b) | a \in S_A\}. \tag{3}$$

As relation $A \Leftrightarrow B$ often indicates the corresponding between particular states of A and B, such as

$$\omega = \{(a_i, b_i) | a_i \in S_A, b_i \in S_B, i = 1, \ldots, q\}, \tag{4}$$

each pair of those states forms a combination between states of A and B. Hence, all these pairs (combinations) form a piece of knowledge given by Definition 2.1.

Definition 2.2. *The knowledge base (Ω) of the domain D before time t_c is a non-empty set of domain-specific knowledge $\omega(t)$ such that*

$$\Omega = \bigcup_{t < t_c, t \in T} \Omega(t), \tag{5}$$

and $\Omega(t)$ is the set of knowledge at the time t.

Definition 2.2 indicates that the knowledge base of domain D is composed of a set of state combinations of relevant objects. Moreover, it is predictable that the knowledge base is incomplete in most situations because it consists of knowledge known to the end of a particular time slot. This feature is in accordance with people's cognitive experience. Secondly, the knowledge base may includes duplicate knowledge because some knowledge is correct in multiple times. Furthermore, the knowledge base may include both consistent and inconsistent knowledge because some knowledge is only applicable to some special circumstances.

By Definitions 2.1 and 2.2, it is known that there are three kinds of relationship between a state combination c of objects and a knowledge base, i.e.,

- c is an existed combination. In this case, c occurs in some pieces of knowledge in Ω.
- c can be a potential combination. In this case, c does not occur in any piece of knowledge in Ω but any of its component (i.e., the state of an particular object) is a normal state.
- c can never be a potential combination. In this case, c does not occur in any piece of knowledge in Ω and at least one of its components is an abnormal state.

Since a piece of real-time information obtained through observation can be expressed by a state combination of related objects, the above three relations indicate possible process strategy for detecting the information logical consistency of real-time information. The first case means that the obtained state combination has been recognized. It is not needed to check the consistency of such information provided that the knowledge base is consistent. The second case shows that the obtained state combination has not been observed previously but it may exist. Hence, it is needed to check the consistency of such information on the basis of the domain-specific knowledge base. Once it is known that the new state combination is consistent with the knowledge base, the state combination can be added to the knowledge base as a piece of new knowledge. As for the third case, the obtained state combination is bound to be inconsistent with the knowledge base. Therefore, it is not needed to check the consistency of such information.

2.2. State-based knowledge consistency

State-based knowledge consistency refers to two levels of meanings. On the first level, knowledge consistency means the consistency in a knowledge base at a given time, i.e., no paradox can be derived from each $\Omega(t)$. On the second level, knowledge consistency refers to the consistency of the whole knowledge base, i.e., no contradiction can be obtained from Ω. Existing information detection methods mainly focus on the consistency of the whole knowledge base and the consistency of knowledge at each time is presumed implicitly. Because those techniques do not consider the influence of time change, the knowledge set at each time is the same. Obviously, such a knowledge base is smaller than the one given in Definition 2.2 on the one hand. On the other hand, the smaller knowledge base may exclude some consistent information by mistake.

In this paper, we mainly focus on the consistency at a given time, i.e., the consistency between real-time information and the knowledge base $\Omega(t)$, where $t \in T$. To do this, we suppose Ω is consistent.

Definition 2.3. *Let $\omega(t)$ be a piece of knowledge, $J = \{i_1, \ldots, i_q\} \subseteq J(\omega(t))$ a non-empty set. Then the J-part of $\omega(t)$ is denoted by $\omega(t)|_J$ such that:*

$$\omega(t)|_J = \{(s_{k_1}^{(i_1)}, \ldots, s_{k_q}^{(i_q)}) | \exists (s_{k_1}^{(i_1)}, \ldots, s_{k_q}^{(i_q)}, s_{k_{l_1}}^{(l_1)}, \ldots, s_{k_{l_p}}^{(l_p)}) \in \omega(t)\}. \quad (6)$$

Definition 2.3 indicates a J-part of a piece of knowledge is the set of J-part of each state combination in the knowledge.

By Definition 2.1, a J-part of a piece of knowledge $\omega(t)$ is also a piece of knowledge. This knowledge can be seen as a logical consequence of knowledge $\omega(t)$. Therefore, we can use this property to define the consistency between two pieces of knowledge.

Suppose $\omega(t)$ and $\phi(t)$ are two pieces of knowledge, and J is the intersection of $J(\omega(t))$ and $J(\phi(t))$, and J is not empty set. Then $\omega(t)|_J$ and $\phi(t)|_J$ have three possible relationships, i.e.,

(1) $\omega(t)|_J = \phi(t)|_J$. In this case, the same conclusion is derived from two different pieces of knowledge. This means these two pieces of knowledge are consistent.

(2) $\omega(t)|_J \cap \phi(t)|_J \neq \emptyset$ but $\omega(t)|_J \neq \phi(t)|_J$. In this case, there is a common part between the logical consequences from two different pieces of knowledge. This means these two pieces of knowledge are partly consistent although the consequences from them may not completely coincide.

(3) $\omega(t)|_J \cap \phi(t)|_J = \emptyset$. In this case, no common part between logical consequences of two different pieces of knowledge exists. This indicates potential inconsistency between those knowledge.

However, sometimes the intersection of $J(\omega(t))$ and $J(\phi(t))$ is an empty set. In this case, to judge the consistency between $\omega(t)$ and $\phi(t)$, we need to find a possible combination which links $\omega(t)$ and $\phi(t)$ through some intermediate knowledge. Suppose $\psi(t)$ is a piece of knowledge and both $J(\omega(t))\cap J(\psi(t))$ and $J(\phi(t))\cap J(\psi(t))$ are not empty sets. We hope to find such a state combination c

$$\begin{aligned}(s_1^{\omega(t)},\cdots,s_{i_1}^{\omega(t)},\cdots,s_{i_m}^{\omega(t)},s_{i_m+1}^{\psi(t)},\cdots,s_{j_1-1}^{\psi(t)},\\ s_{j_1}^{\psi(t)},\cdots,s_{j_m}^{\phi(t)},\cdots,s_{j_n}^{\phi(t)})\end{aligned} \tag{7}$$

such that

$$\begin{aligned}&(s_1^{\omega(t)},\cdots,s_{i_1}^{\omega(t)},\cdots,s_{i_m}^{\omega(t)})\in\omega(t)\\ &(s_{i_1}^{\omega(t)},\cdots,s_{i_m}^{\omega(t)},s_{i_m+1}^{\psi(t)},\cdots,s_{j_1-1}^{\psi(t)},s_{j_1}^{\psi(t)},\cdots,s_{j_m}^{\phi(t)})\in\psi(t)\\ &(s_{j_1}^{\psi(t)},\cdots,s_{j_m}^{\phi(t)},\cdots,s_{j_n}^{\phi(t)})\in\phi(t).\end{aligned}$$

If such a combination exists, a potential consistent observation can be obtained from $\omega(t)$ and $\phi(t)$. Hence, they are consistent to some extent. (In the following, we say $\psi(t)$ connects between $\omega(t)$ and $\phi(t)$ and such a combination c is called a string linking $\omega(t)$ and $\phi(t)$.)

Based on the above analysis, the following definitions about consistency between two pieces of knowledge are given.

In the following, we use J^* to denote the intersection of $J_{\omega(t)}$ and $J_{\phi(t)}$.

Definition 2.4. *Suppose $\omega(t)$, $\phi(t)\in\Omega(t)$ are two pieces of knowledge.*

When J^* is a non-empty set, $\omega(t)$ and $\phi(t)$ are said to be strict consistent if $S_j^{(\omega(t))}=S_j^{(\phi(t))}$ for any $j\in J^*$; $\omega(t)$ and $\phi(t)$ are said to be partial consistent if $S_j^{(\omega(t))}\neq S_j^{(\phi(t))}$ for some $j\in J^*$ and $S_j^{(\omega(t))}\cap S_j^{(\phi(t))}\neq\emptyset$ for any $j\in J^*$; and $\omega(t)$ and $\phi(t)$ are said to be inconsistent if for some $j\in J^*$, $S_j^{(\omega(t))}\cap S_j^{(\phi(t))}=\emptyset$.

When J^* is an empty set, $\omega(t)$ and $\phi(t)$ are said to be strict consistent if there is a sequence of knowledge $\psi_1(t)$, $\psi_2(t)$, ..., $\psi_n(t)$ which connect between $\omega(t)$ and $\phi(t)$ such that any two consequent pieces of knowledge are strict consistent; $\omega(t)$ and $\phi(t)$ are said to be partial consistent if there is a sequence of knowledge $\psi_1(t)$, $\psi_2(t)$, ..., $\psi_n(t)$ which connect between $\omega(t)$ and $\phi(t)$ and there is at lest one string linking $\omega(t)$ and $\phi(t)$; $\omega(t)$ and $\phi(t)$ are said to be inconsistent if no string linking $\omega(t)$ and $\phi(t)$ can be found.

By Definition 2.4, it is known that the operations of extracting the J^*-part of two pieces of knowledge and finding a string linking them are very important for detecting the consistency of two pieces of knowledge. Here, we formally define the first operation by $E(\omega,\phi)$ and the second operation by $C(\omega,\phi)$. These two operations serve as knowledge generalization and knowledge specification.

Definition 2.5 (extracting). *Let $\omega(t)$, $\phi(t)\in\Omega(t)$ be two pieces of knowledge and $J(\omega(t))\cap J(\phi(t))\neq\emptyset$. Then $E(\omega,\phi)$ is obtained by:*

$$E(\omega(t),\phi(t))=\omega(t)|_{J^*}\cap\phi(t)|_{J^*}, \tag{8}$$

where $J^*=J(\omega(t))\cap J(\phi(t))\neq\emptyset$.

Definition 2.6 (coupling). *Let $\omega(t)$, $\phi(t)\in\Omega(t)$ be two pieces of knowledge and $J^*\neq\emptyset$. Then $C(\omega,\phi)$ is obtained by:*

$$C(\omega(t),\phi(t))=\{c|c \text{ is a string linking } \omega(t) \text{ and } \phi(t)\}. \tag{9}$$

Proposition 2.1. *Let $\omega(t)$ and $\phi(t)$ be two pieces of strict (partial) consistent knowledge and $J^*\neq\emptyset$, then*

$$E(\omega(t),\phi(t))=C(\omega(t),\phi(t))|_{J^*}. \tag{10}$$

Proof: For any $c\in E(\omega(t),\phi(t))$, we have $c_1\in\omega(t)$ and $c_2\in\phi(t)$ such that c is the common section of c_1 and c_2. Then, there is a string $\tilde{c}$ linking $\omega(t)$ and $\phi(t)$.

Hence, $\tilde{c} \in C(\omega(t), \phi(t))$. By Definition 2.3, we have $c|_{J^*} = c$. Obviously, $c \in C(\omega(t), \phi(t))|_{J^*}$ and $E(\omega(t), \phi(t)) \subseteq C(\omega(t), \phi(t))|_{J^*}$.

For any $\tilde{c} \in C(\omega(t), \phi(t))|_{J^*}$, there exists $c_1 \in \omega(t)$ and $c_2 \in \phi(t)$ such that $\tilde{c}$ is the common section of them. Notice that $\tilde{c} = \omega(t)|_{J^*}$ and $\tilde{c} = \phi(t)|_{J^*}$, $\tilde{c} \in E(\omega(t), \phi(t))$. Therefore, $C(\omega(t), \phi(t))|_{J^*} \subseteq E(\omega(t), \phi(t))$. □

Definition 2.5 and Definition 2.6 are two methods of obtaining new knowledge from existed knowledge base because $E(\omega(t), \phi(t))$ and $C(\omega(t), \phi(t))$ themselves are two pieces of knowledge by Definition 2.1. In the following, the fact that a piece of knowledge $\psi(t)$ is obtained from a set of knowledge $\Psi(t)$ by the two methods is denoted by $\Psi(t) \models_D \psi(t)$. Hence, $\{\omega(t), \phi(t)\} \models_D E(\omega(t), \phi(t))$ and $\{\omega(t), \phi(t)\} \models_D C(\omega(t), \phi(t))$. Moreover, these two methods have close relation with the consistency of two pieces of knowledge seen from Definition 2.4. The relation can be expressed by the following proposition.

Proposition 2.2. *Let $\omega(t)$ and $\phi(t)$ be two pieces of inconsistent knowledge, then either $E(\omega, \phi)$ or $C(\omega, \phi)$ is empty sets.*

Proof: We consider two possible situations. Firstly, suppose J^* is a non-empty set. In this case, $\omega(t)$ and $\phi(t)$ are inconsistent if for some $j \in J^*$, $S_j^{(\omega(t))} \cap S_j^{(\phi(t))} = \emptyset$. This means for any $c_1 \in \omega(t)$ and any $c_2 \in \phi(t)$, $c_1|_{J^*} \neq c_2|_{J^*}$. Hence, $c_1|_{J^*} \notin \phi(t)|_{J^*}$ and then $\omega(t)|_{J^*} \cap \phi(t)|_{J^*} = \emptyset$. Therefore, $E(\omega(t), \phi(t))$ is an empty set. By Proposition 2.1, $C(\omega(t), \phi(t))$ is an empty set. Secondly, suppose J^* is an empty set. In this case, there is not a string c which links $\omega(t)$ and $\phi(t)$. Hence, $C(\omega(t), \phi(t))$ is an empty set. □

Proposition 2.2 and Definition 2.4 indicate that the consistency between two pieces of knowledge can be implemented through checking whether empty knowledge can be derived from them. Next, we extend this idea to a set of knowledge.

As $C(\omega, \phi)$ is a consequence of the knowledge ω and ϕ, and ω (or ϕ) is a consequence of $E(\omega, \phi)$. In the following, $C(\omega, \phi)$ and $E(\omega, \phi)$ will be denoted by $\omega \sqcap \phi$ and $\omega \sqcup \phi$ respectively.

Based on Definition 2.6, let $\Omega^*(t) \subseteq \Omega(t)$, and define $C(\Omega^*(t))$ as follows

- $\Omega^*(t) \in C(\Omega^*(t))$;
- for any $\omega(t)$, $\phi(t) \in C(\Omega^*(t))$, $\omega(t) \sqcap \phi(t) \in C(\Omega^*(t))$;
- for any $\omega(t)$, $\phi(t) \in C(\Omega^*(t))$, $\omega(t) \sqcup \phi(t) \in C(\Omega^*(t))$.

Definition 2.7. *A set of knowledge $\Omega^*(t)$ is called consistent if $C(\Omega_i^*(t))$ does not include empty knowledge; otherwise, it is called inconsistent.*

Generally speaking, that checking a set of knowledge is consistent or not is a time-consuming task. However, we have a simplified strategy here.

First, we introduce the concept of knowledge covering to illustrate the relationship between two pieces of knowledge.

Definition 2.8. *Two pieces of knowledge $\omega(t)$ and $\phi(t)$ are said to be equivalent and denoted by $\omega(t) \equiv \phi(t)$ if $J(\omega(t)) = J(\phi(t))$ and $S_j^{(\omega(t))} = S_j^{(\phi(t))}$ for any $j \in J(\omega(t))(= J(\phi(t)))$.*

Definition 2.8 depicts the phenomena when a piece of knowledge can be expressed in many ways. For instance, both "2000 Sydney Olympic Game" and "the 26th Olympic Game" refer to the same Game hold in Sydney in October, 2000.

Definition 2.9. *A piece of knowledge $\omega(t)$ is said to be a logical consequence of knowledge $\phi(t)$ if the following conditions hold:*

(1) $J(\omega(t)) \subseteq J(\phi(t))$,

(2) $\omega(t) = \phi(t)|_{J(\omega(t))}$.

In the following, we shall denote $\phi(t) \models \omega(t)$ if knowledge $\omega(t)$ is a logical consequence of knowledge $\phi(t)$. Obviously, two equivalent knowledge ω and ϕ are logical consequence of each other.

By Definition 2.4 and Definition 2.9, if two pieces of knowledge $\omega(t)$ and $\phi(t)$ are strict consistent, then the following conclusion holds.

Proposition 2.3. *If $\omega(t)$ and $\phi(t)$ are two pieces of strict consistent knowledge, then*

- $\omega(t) \models E(\omega(t), \phi(t))$ *and* $\phi(t) \models E(\omega(t), \phi(t))$*; or*
- $C(\omega(t), \phi(t)) \models \omega(t)$ *and* $C(\omega(t), \phi(t)) \models \phi(t)$.

It is easy to verify that the following conclusions hold.

Proposition 2.4. *Let $\omega(t)$, $\phi(t) \in \Omega(t)$ and $\omega(t)$, $\phi(t) \notin \mho$*

(1) $\omega(t) \models \omega(t)$ *for any* $\omega(t) \in \Omega(t)$.
(2) $\omega(t) \equiv \phi(t)$ *if* $\omega(t) \models \phi(t)$ *and* $\phi(t) \models \omega(t)$.
(3) $\omega(t) \models \psi(t)$ *if* $\omega(t) \models \phi(t)$ *and* $\phi(t) \models \psi(t)$. □

The logical consequence relationship $\models$ gives a hierarchical structure among a set of knowledge at time t. We draw the hierarchical structure in a graph according to the following principle:

$$\omega(t) \preccurlyeq \phi(t) \text{ if and only if } \phi(t) \models \omega(t), \tag{11}$$

where $\omega(t) \preccurlyeq \phi(t)$ means that $\phi(t)$ covers $\omega(t)$. The relationship $\preccurlyeq$ is a partial order, which is called knowledge covering relationship. The knowledge covering relationship among $C(\omega, \phi)$, $\omega(t)$, $\phi(t)$, and $E(\omega, \phi)$ is shown in Figure 1.

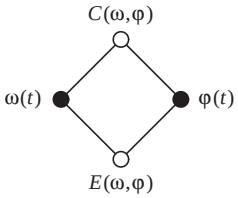

Figure 1: Covering among knowledge pieces

Notice from Definition 2.9 and Fig. 1, the length of state combinations in $C(\Omega^*(t))$ is increasing but the number of those combinations is decreasing. By this feature, we can simplify the search of empty knowledge from $C(\Omega^*(t))$.

We call a set of knowledge Ω is indivisible if there doesn't exist a division of $J(\Omega) = J_1 \cup J_2 \cup \cdots \cup J_m$ such that

- $J_i \cap J_k = \emptyset$ if $i \neq k$, and
- for any $\omega \in \Omega$, there exists unique l, $J(\omega(t)) \subseteq J_l$,

where i, k, $l \in \{1, 2, \ldots, m\}$.

Suppose $\Omega^*(t)$ is divided into m indivisible parts, $\Omega_1^*(t)$, $\cdots$, $\Omega_m^*(t)$. We have

Proposition 2.5. *Let $\Omega^*(t)$ be a set of knowledge, then*

$$C(\Omega^*(t)) = \bigcup_{i=1}^{m} C(\Omega_i^*(t)). \tag{12}$$

Proposition 2.5 indicates the empty knowledge will occurs in some $C(\Omega_i^*(t))$s. Thus, the searching space is reduced.

Remark 2.2. *That coupling a set of knowledge aims at finding out all possible combinations of states of the related objects. This operation depicts the inner dependencies among objects which exactly are the knowledge we have about a specific domain. Hence, we can use these combinations to detect inconsistency in the real-time information.*

3. An information logical inconsistency detection method

Based on the knowledge representation approach proposed, we give an LID method in this section.

First, the problem of information logical inconsistency detection for real-time information is:

In a situation, related knowledge $\Omega = \cup_{t \in T} \Omega(t)$ has been stored, which involves in a set of objects $O = \{O_1, \ldots, O_n\}$. At a given time t, we collect a set of observations (i.e., real-time information), $S^* = \{S_j^* \mid j \in J\}$ about some objects $O^* = \{O_j^* \mid j \in J\} \subseteq O$, $J \subseteq \{1, 2, \ldots, n\}$. Then, we shall know if these observations are information logical inconsistent with the knowledge stored in a knowledge base.

To deal with this problem, we suppose the stored knowledge $\Omega(t)$ at time t is consistent. Therefore, a potential information logical inconsistency must be introduced by S^*.

The LID method is composed of five steps as follows.

Step 0: Check $C(S^*)$. If $C(S^*)$ is empty knowledge, then these observations are logical inconsistent and this method stops; otherwise, go to Step 1. Step 0 aims at finding information logical inconsistency in these observations themselves.

Step 1: Compare $J(S^*)$ and $J(C(\Omega(t)))$. If $J(C(\Omega(t))) \supseteq J(S^*)$, then go to Step 2; otherwise, go to Step 3. This step aims to determine whether the stored knowledge adapts to the needs of a logical inconsistency detection task. If the stored knowledge and these observations involve the same objects, then the stored knowledge meets the requirements of the detecting task. Otherwise, some observations cannot be detected by the stored knowledge.

Step 2: Check whether $S^* \cap (C(\Omega(t)))|_{J(S^*)} \neq \emptyset$. If $S^* \cap (C(\Omega(t)))|_{J(S^*)} \neq \emptyset$, then these observations are logical consistent; otherwise, they are logical inconsistent and the detecting is ended. This step aims at identifying information logical inconsistency in these observations when the stored knowledge is sufficient enough.

Step 3: Divide S^* into two parts S_1^* and S_2^* such that

$$S_1^* = \{\omega|_{J(C(\Omega(t)))} \mid \omega \in S^*\}$$
$$S_2^* = \{\omega|_{J(\omega(t))\setminus J(C(\Omega(t)))} \mid \omega \in S^*\}$$

For S_1^*, let $S^* = S_1^*$, go to Step 2. For S_2^*, we cannot use the stored knowledge to detect logical inconsistency of observations in it. Hence, these observations in S_2^* are treated as new data and detected by the related data inconsistency methods as presented in [5, 21, 3]. For each observation $s \in S_2^*$, if there exists an observation in S_2^* which is inconsistent, then the observations S^* is inconsistent. Go to Step 4. When the stored knowledge is insufficient for detecting all observations, current process is applied. This step is used on the basis that if a part of these observations are logical inconsistent, then all of them as a whole must be logical inconsistent.

Step 4: For any consistent observation $s \in S_2^*$, construct $C(C(\Omega(t)), s)$ and add it to $\Omega(t)$. This step is an additional work on the consideration of updating the stored knowledge in order to preserve the completeness and effectiveness of a knowledge base in a real warning system. Notice that the added new knowledge is consistent with itself and the stored knowledge; hence, the obtained knowledge base is still consistent after updating.

Step 5: Explain conclusion and end.

By above steps, we can implement the information logical inconsistency detection for real-time information.

4. Illustration examples

In this section, the effectiveness and possible applications of the proposed ILID method are illustrated through two examples.

First, we use the ILID method for single object with boolean states, i.e., the object has two opposite states.

Example 4.1. *A power station is an important industrial department for emergency response. Its function is always under monitoring. Suppose a power station has a monitoring system which has 14 lookouts distributed in different places. Each lookout will report the states of its local place every hour. Let a knowledge base $\Omega(t)$ about the power station's function shown in Table 1, where ω_i, $i = 1, 2, \ldots, 10$, is a piece of knowledge, and p_j $(j = 1, 2, \ldots, 14)$ is the i-th lookout and each of them reports two possible states* 1 *(for abnormal function) and* 0 *(for normal function).*

Let $S^* = \{(p_1 = 1,\ p_2 = 1,\ p_3 = 1,\ p_4 = 1)\}$ be a set of observations.

Using the presented ILID method, we have:

Step 0: Obviously, $C(S^*)$ isn't empty. Goto Step 1.

Step 1: By coupling the knowledge bases $C(\Omega(t))$, we

Table 1: A knowledge base $\Omega(t)$.

No.	Knowledge
ω_1	$\{(p_1 = 1, p_2 = 1, p_5 = 1, p_6 = 1)\}$
ω_2	$\{(p_2 = 1, p_{14} = 1)\}$
ω_3	$\{(p_6 = 1, p_{10} = 1)\}$
ω_4	$\{(p_3 = 1, p_4 = 1, p_7 = 1)\}$
ω_5	$\{(p_{10} = 1, p_{14} = 1)\}$
ω_6	$\{(p_7 = 1, p_{10} = 1, p_{11} = 1)\}$
ω_7	$\{(p_8 = 1, p_{11} = 1)\}$
ω_8	$\{(p_8 = 1, p_7 = 1)\}$
ω_9	$\{(p_{10} = 1, p_{12} = 1)\}$
ω_{10}	$\{(p_{10} = 1, p_{13} = 1)\}$

have a piece of knowledge:

$$\{(p_1 = 1, p_2 = 1, p_3 = 1, p_4 = 1, p_5 = 1, p_6 = 1, p_7 = 1, p_8 = 1, p_{10} = 1, p_{11} = 1, p_{12} = 1, p_{13} = 1, p_{14} = 1)\}.$$

As $J(S^*) \subseteq J(C(\Omega(t)))$*, then goto Step 2.*

Step 2: By Definition 2.3, we have

$$(C(\Omega(t)))|_{\{1,2,3,4\}} = \{(p_1 = 1, p_2 = 1, p_3 = 1, p_4 = 1)\}. \tag{13}$$

So, $S^* = C(\Omega(t))|_{\{1,2,3,4\}}$*. The observations are consistent, which means the power station is functioning abnormally. Stop.*

Now, let $S^* = \{(p_1 = 1,\ p_2 = 1,\ p_3 = 1,\ p_9 = 1)\}$*. By the proposed ILID method, we have*

Step 1: $J(S^*) \nsubseteq J(C(\Omega(t)))$*. Then goto Step 3.*

Step 3: Dividing S^* *into* $S_1^* = \{(p_1 = 1, p_2 = 1, p_3 = 1)\}$ *and* $S_2^* = \{(p_9 = 1)\}$*. For* S_1^**, goto Step 2. For* S_2^**, without loss of generality, suppose it is consistent by the rule map technique, then goto Step 4.*

Step 2: For S_1^**, we know it is consistent.*

Step 4: Because the S^* *is a set of consistent observations, we shall update our knowledge by coupling* $\{(p_9 = 1)\}$ *and* $C(C(\Omega(t)))$ *and have*

$$\{(p_1 = 1, p_2 = 1, p_3 = 1, p_4 = 1, p_5 = 1, p_6 = 1, p_7 = 1, p_8 = 1, p_9 = 1, p_{10} = 1, p_{11} = 1, p_{12} = 1, p_{13} = 1, p_{14} = 1)\}. \tag{14}$$

Step 5. End.

From this example, we can see that the presented ILID method is very effective as it only needs to detect a subset of possible combinations of states of related objects. This feature is suitable for a real problem since it can reduce the searching space and save the searching time by avoiding detection for insignificant combinations.

Second, we use the ILID method for objects with multiple states. In this situation, a piece of knowledge may cover multiple combinations of states.

Example 4.2. *Suppose another warning system monitoring the changes of three objects,* A*,* B*, and* C*. Each object can take observation values from {slow (1), medium (2), fast (3)}. Let* $r(A, B)$ *be the knowledge "A's change is greater than B's change." Now we have a knowledge base* $\Omega = \{\omega = r(A, B),\ \phi = r(B, C)\}$ *and a set of real-time observations* $S^* = \{(A = 2, B = 2, C = 1)\}$*. Hence we have*

$$\omega = \{(A = 2, B = 1), (A = 3, B = 1), (A = 3, B = 2)\}$$
$$\phi = \{(B = 2, C = 1), (B = 3, C = 1), (B = 3, C = 2)\}.$$

Using the ILID method, we have the following steps to detect logical inconsistency for the real-time information.

Step 1: By coupling these two pieces of knowledge, we have

$$C(\Omega(t)) = \{(A = 3, B = 2, C = 1)\}. \tag{15}$$

Because $J(S^*) = J(C(\Omega(t)))$*, goto Step 2.*

Step 2: Notice that $(C(\Omega(t)))|_{\{A,B,C\}} = \{(A = 3, B = 2, C = 1)\}$ *and* $S^* \nsubseteq (C(\Omega(t)))|_{\{A,B,C\}}$*, therefore, the observations are logical inconsistent. Then goto Step 5 and stop.*

The presented ILID method is not only be used for the real-time observations but also be used for detecting logical inconsistency in knowledge bases. By taking knowledge in a knowledge base as a set of observations, we can treat the knowledge base as being generated from an empty knowledge base. Hence, applying the proposed method, we can detect the logical inconsistency of the knowledge in the knowledge base.

Continuing Example 4.2, suppose we have the third piece of knowledge $\psi = r(C, A)$. We have

$$\begin{aligned} \psi = \{&(C = 2, A = 1), \\ &(C = 3, A = 2), \\ &(C = 3, A = 1)\}. \end{aligned} \tag{16}$$

Now $C(\Omega) = \mho$, which means the knowledge base is inconsistent.

By taking this advantage, we can use the ILID method to detect logical inconsistency in both real-time and stored knowledge for a warning system. Obviously, this can improve the facility and function of a warning system.

5. Conclusion

Detecting logical inconsistency in information is an important aspect to develop real applications in a people-centered warning system. Since these applications are always applied in specific domains, detection approaches should be domain-oriented and should be based on domain knowledge which is *ad hoc*, decentralized, and contextualized [20]. Considering domain knowledge is object-state related, this paper first presented a state-based domain knowledge representation approach, and then proposed the ILID method for domain-specific information.

In the state-based domain knowledge representation approach, a piece of domain knowledge is represented by some state combinations of relevant objects. Thus, logical relationship between knowledge is defined through those state combinations. Furthermore, the strict and partial consistency of domain knowledge base is also defined on those state combinations. This knowledge representation approach has flexibility to depict domain-specific knowledge.

The developed ILID method can be seen as an application of the state-based domain knowledge representation approach. The ILID method includes five main steps which are implemented through the coupling and extracting operations on state combinations in the relevant knowledge. This implementation combines the merits of reasoning-based and graph-based inconsistency detection approaches. To test and illustrate the efficiency of the ILID method, two examples are described. Results indicate that the ILID method can be used to detect logical inconsistency of real-time observations, and can also be used to detect logical inconsistency of a stored knowledge base. This is important because both situations exist in warning systems.

Based on current results, our future study includes integrating and applying the proposed ILID method to information process tasks in a people-centered warning system in specific domains.

Acknowledgement

The work presented in this paper was supported by Australian Research Council (ARC) under Discovery Project DP0880739.

References

[1] Amgoud, L., Kaci, S., 2007. An argumentation framework for merging conflicting knowledge bases. International Jounral of Approximate Reasoning 45, 321–340.

[2] Basher, R., 2006. Global early warning systems for natural hazards: systematic and people-centred. Philosophical Transactions of the Royal Society A 364, 2167–2182.

[3] Beliakov, G., Warren, J., 2001. Appropriate choice of aggregation operators in fuzzy decision support systems. IEEE Trans. on Fuzzy Systems 9 (6), 773–784.

[4] Botten, N., 1992. Complex knowledge-base verification using matrices. Lecture Notes in Artificial Intelligence 604, 225–235.

[5] Bruni, R., 2004. Discrete models for data imputation. Discrete Applied Mathematics 144, 59–69.

[6] Chang, C. L., Lee, R. C. T., 1973. Symbolic and Mechanical Theorem Proving. Academic Press, Inc., Orlando, FL, USA.

[7] de Amo, S., Pais, M. S., 2007. A paraconsistent logic programming approach for querying inconsistent databases. International Jounral of Approximate Reasoning 46, 366–386.

[8] Hunter, A., 1998. Paraconsisten logics. In: Gabbay, D., Smets, P. (Eds.), Handbook of Defeasible Reasoning and uncertain Information. Kluwer Academic Publishers, pp. 13–43.

[9] Hunter, A., 2003. Evaluating the significance of inconsistencies. In: Proceedings of the 2003 International Joint Conference on AI (IJCAI03). pp. 468–473.

[10] Mazure, B., Saïs, L., Grégoire, E., June 1997. Checking several forms of consistency in nonmonotonic knowledge-bases. In: Gabbay, D. M., Kruse, R., Nonnengard, A., Ohlbach, H. J. (Eds.), Qualitative and Quantitative Practical Reasoning, First International Joint Conference on Qualitative and Quantitaitve Practical Reasoning ECSQARU-FAPR'97, Bad Honnef, Germany. Vol. 1244 of Lecture Notes in Computer Sciences. Springer, pp. 122–130.

[11] Molodtsov, D., 1999. Soft set theory – first results. Computers and Mathematics with Applications, An International Journal 37, 19–31.

[12] Mues, C., Vanthienen, J., 2004. Efficient rule base verification using binary decision diagrams. In: Proc. of Database and Expert Systems Application. Vol. 3180 of Lecture Notes in Computer Science. pp. 445–454.

[13] Mues, C., Vanthienen, J., 2004. Improving the scalability of rule base verification using binary decision diagrams: an empirical study. In: Proc. of Advances in Artificial Intelligence. Vol. 3238 of Lecture Notes in Computer Science. pp. 381–395.

[14] Nguyen, N. T., feb 2005. Processing inconsistency of knowledge on semantic level. Journal of Universal Computer Science 11 (2), 285–302.

[15] Park, J. H., Seong, P. H., 2002. An integrated knowledge base development tool for knowledge acquisition and verification for NPP dynamic alarm processing systems. Annals of Nuclear Energy 29 (4), 447–463.

[16] Pawlak, Z., Grzymala-Busse, J., Slowinski, R., Ziarko, W., 1995. Rough sets. Communications of the ACM 38 (11), 89–95.

[17] Pawlak, Z., Skowron, A., 2006. Rough sets and Boolean reasoning. Information Sciences 177 (1), 41–73.

[18] Pawlak, Z., Skowron, A., 2006. Rudiments of rough sets. Information Sciences 177 (1), 3–27.

[19] Polat, F., 1993. UVT: A unification-based toold for knowledge base verification. IEEE Expert–Intelligent Systems & Their Applications 8 (3), 69–75.

[20] Raman, M., Ryan, T., Olfman, L., 2006. Knowledge management system for emergency preparedness: An action research study. In: HICSS'06: Proceedings of the 39th Annual Hawaii International Conference on System Sciences. Vol. 2. IEEE Computer Society, Washington, DC, USA, p. 37b.

[21] Russell, S. J., Norvig, P., 1995. Artificial Intelligence: A Modern Approach, 1st Edition. Prentice Hall, NJ.

[22] Scarpelli, H., Gomide, F., 1994. A high level net approach for discovering potential inconsistencies in fuzzy knowledge bases. Fuzzy Sets and Systems 64, 175–193.

[23] Wu, P., Su, S. Y. W., 1993. Rule validation based on logical deduction. In: Proc. 2nd International Conference on Information and Knowledge Management. pp. 164–173.

[24] Yang, S. J. H., Tsai, J. J. P., Chen, C.-C., 2003. Fuzzy rule base systems verification using high-level Petri Nets. IEEE Transactions on Knowledge and Data Englineering 15 (2), 457–473.

[25] Zhang, D., Luqi, 1999. Approximate declarative semantics for rule base anomalies. Knowledge-Based Systems 12, 341–353.