On automatic knowledge validation for Bayesian knowledge bases

Eugene Santos Jr. a,⁎, Hang T. Dinh b

a Thayer School of Engineering, Dartmouth College, Hanover, NH 03755, United States
b Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269-2155, United States

Received 19 September 2006; received in revised form 28 June 2007; accepted 9 July 2007
Available online 15 August 2007

Abstract

Knowledge validation, as part of knowledge base verification and validation, is a critical process in knowledge engineering. The ultimate goal of this process is to make the knowledge base satisfy all test cases given by human experts. This is further complicated by factors such as uncertainty and incompleteness. Our paper covers theoretical results in knowledge validation for Bayesian Knowledge Bases (BKBs), a probabilistic model extended from Bayesian Networks for representing knowledge in uncertain domains. First, we study the consistency of test case sets by identifying the necessary and sufficient conditions for a test case set such that there exists a knowledge base satisfying all of its test cases. Second, we analyze the thrashing problem which is the interminable oscillation of the knowledge base’s state when validating by parameter refinement. We propose an approach to validating BKBs that effectively eliminates thrashing under certain conditions of the original knowledge base and the test case set.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Bayesian knowledge-bases; Uncertainty; Incomplete information; Knowledge validation; Knowledge engineering

1. Introduction

Typically, the development of a knowledge-base starts with the knowledge acquisition stage, in which the knowledge engineers have to collect data and information from domain experts and other sources. This task is so complex, both objectively and subjectively, that we cannot guarantee the reliability of the knowledge base built after acquisition. Hence, the knowledge base must go through another stage called verification & validation (V&V) [14,12]. The verification step is to check whether the knowledge base satisfies the specifications regarding its model of knowledge representation. Once the knowledge base is verified to be free from
anomalies with respect to the designated specifications, it will be sent to the validation process, which is defined by most researchers in V&V for intelligent systems as a process of ensuring that the output of the intelligent system is equivalent to those of human experts when given the same input [2,11]. In particular, the experts provide a set of test cases, each of which is a pair consisting of an input query and the corresponding expected answer. The equivalence here means that when running the query given in each test case, the knowledge base must infer the same answer as the experts’ expected one associated with the test case.

Originally, the process of knowledge validation only involves evaluating the knowledge base’s ability to reach correct conclusions [7,24]. However, the purpose of knowledge validation has gone beyond evaluation. Gonzalez and Barr [11], when clarifying the definitions of V&V for intelligent systems, argue that V&V must include the aspect of eliminating errors (or refinement). So, for the validation portion, it also has to detect errors in the knowledge base and correct the knowledge base so that all test cases provided by the human experts are satisfied. Several recent approaches in knowledge validation, e.g. [17,40,20,30], concentrate on correcting the knowledge base, which is a considerably difficult task, e.g. theory revision [13,21].

Our focus in this work is on knowledge validation for knowledge bases that handle uncertainty. In particular, we examine Bayesian Knowledge Bases [36]. Briefly, Bayesian Knowledge Bases (BKBs) are a generalization of the well-known probabilistic model called Bayesian Networks [25]. The main advantages of BKBs over Bayesian Networks (BNs) are the ability to handle information incompleteness and to represent cyclic dependencies among random variables. In addition, a BKB semantically represents uncertain knowledge in the style of “if–then” rules [37,27], which is familiar to most domain experts and knowledge engineers. As pointed out in [37],

Probabilistic models exhibiting significant local structure are common. In such models, explicit representation of that structure as done in BKBs, is advantageous, as the resulting representation is much more compact than the full table representation of the conditional probability tables (CPTs) in a BN. For example, consider the following setting: $X$, a binary variable, is known to be true if any of the variables $Y_i$ is true, for $1 \leq i \leq n$, and $X$ is false with probability $p$ otherwise. The global structure here is that $X$ depends on all the $Y_i$ and in a BN one might represent this with a set of arcs $\{(Y_i, X)\}_{1 \leq i \leq n}$. The representation of the distribution in the “standard” form of a CPT would require $O(2^n)$ entries. However, the (partially) given distribution also exhibits “local” structure, as when $Y_i$ is known to be true for some $i$, $X$ no longer depends on the value of $Y_j$ for $j \neq i$. The size of the representation of the conditional probabilities in terms of rules is only $O(n)$. Although work has been done on representing local structure using other methods, such as local decision trees and default tables [4], rules have significant advantages in size of the representation, as well as their better explainability. For example, contrasting rules with decision trees as a representation of local structure, every decision tree is compactly representable as a set of rules, while the reverse is not necessarily true – the decision tree may be exponentially larger than the set of rules [1]. Although rule-based systems for representing an exact distribution exist (e.g. [26]), these systems are a (compact) notational variant of Bayes networks, and are thus less flexible than BKBs, as they do not allow for incompleteness or cyclicity.

BKBs are currently being applied in a number of domains including engine diagnosis [30], user intent inferencing [5], adversarial decision making [35], and data mining [3].

The study of BKB validation is part of the ongoing PESKI1 project which is aimed at providing an integrated environment of intelligent tools for building expert systems whose knowledge bases are modeled in terms of BKBs. Each tool in PESKI [6,15,31,34,30] performs the task of some phase during knowledge engineering, assisting the experts to directly build the knowledge-based systems without the need of knowledge engineers. BKB validation is also implemented as a key tool of PESKI, called BVAL [34]. It is supposed to help expert users automatically validate the BKB after its initial acquisition. Note that BKB validation is not only to evaluate the accuracy of the knowledge base but also to automatically correct the knowledge base according to the given test case set.

---

1 Probabilities, Expert Systems, Knowledge, and Inferencing.
An approach for BKB validation has been proposed earlier in [34,30], but it has limited validation performance because it focuses only on the method for validating a single test case, without considering how test cases influence each other. Under this approach, modifying the BKB to correctly validate a test case can make the BKB fail to satisfy other test cases that were previously validated. As a result, none of the test cases, except for the last one, is guaranteed to be satisfied. Thus, the validation effectiveness is too low for this approach to be generally applicable.

The ultimate goal of BKB validation is that 100% of test cases are correctly validated. However, this will never be achievable if there is some contradiction in the given set of test cases. The contradiction here means that there is no BKB satisfying all test cases in the given set. The simplest example of such a contradiction is any set that includes two distinct test cases with the same query but with different expected answers. Our goal is to define and detect the contradiction in a test case set that will be used for validation.

Another problem of BKB validation that has also been pointed out in [34,30] is the thrashing phenomenon, which can occur when validating all test cases through only parameter adjustment. Thrashing typically happens if there are two test cases that result in inconsistent conditions on the parameters of the knowledge base. For example, to correctly validate one test case, the parameters of the knowledge base must satisfy the constraint $a > b$, while to correctly validate the other test case, they have to satisfy the opposite constraint $b < a$. Of course, the knowledge base will end up oscillating between parameter values. The fundamental questions still open regarding thrashing are (1) What conditions of the knowledge base and the test case set will lead to thrashing? and (2) How to resolve thrashing?. However, this requires a formal analysis of thrashing since the existing approach to BKB validation [34,30] had not identified the root cause of thrashing.

In this work, we provide a formal and rigorous approach to validation for BKBs, in which we analyze mathematically the inconsistency in test case sets and the problem of thrashing. First, we determine the necessary and sufficient conditions for a test case set to be contradiction-free. A test case set meeting that condition will be said to be consistent. We also examine the computational complexity of deciding the consistency of test case sets. Second, we address the thrashing problem, investigating the conditions leading to thrashing. Putting it altogether, we provide a method of validating a BKB with a consistent test case set by both incrementally changing the structure of the BKB and adjusting parameters in the BKB. Updating the BKB structure prevents thrashing from occurring during parameter adjustment. Our method is guaranteed to properly validate a wide variety of BKBs and test case sets.

This paper is organized as follows: We briefly present some background on BKBs in the following section and provide the formal definitions for BKB validation in Section 3. In Section 4, we describe an algorithm for validating BKBs by refining parameters. The consistency of test case sets is presented in Section 5. Section 6 consists of our analysis of the thrashing problem and our approach for validating BKBs by modifying structure to resolve thrashing. The last two Sections 7 and 8, cover our discussion of related work and conclusions, respectively.

2. Background

In this section, we will review the Bayesian Knowledge-Base model. More specific details on this model as well as discussion on its semantic soundness and applications can be found in [36,37,27].

2.1. Representing uncertain knowledge with BKBs

Bayesian Knowledge-Bases (abbrev. BKBs) [36] are a probabilistic model that extends Bayesian Networks in two aspects: The first one is the allowance for more context-specific dependence (independence) by maintaining conditional probabilities at the instantiation level instead of the random variable level as in Bayesian Networks. The second is the allowance of cyclic dependence relationships among random variables. These two features can be seen clearly in the graph structure of BKBs. More specifically, the structure of a BKB is represented by a directed bipartite graph consisting of two different kinds of nodes, namely the instantiation nodes (or I-nodes) and the support nodes (or S-nodes). Each I-node is an instantiation of some random variable in the domain, and is written as $R = v$, where $R$ is a random variable and $v$ is a value (or a set of values) for variable $R$. Each S-node corresponds to the conditional probability of an I-node, which is the child of the S-node, given
a set of I-nodes, which are the direct parents of the S-node. Fig. 1 shows an example of such a graph, which is referred to as a correlation-graph and is formally defined in Definition 2.1.

**Definition 2.1** [36]. **A correlation-graph** over a set \( I \) of I-nodes is a directed graph \( G = (I \cup S, E) \), in which \( S \) is a set of S-nodes disjoint from \( I \), and \( E \) is a subset of \( \{I \times S\} \cup \{S \times I\} \), and for each \( q \in S \) there exists one and only one \( x \in I \) such that \((q, x) \in E\). Furthermore, if there is a link from an S-node \( q \) to an I-node \( x \), we say that \( q \) supports \( x \).

**Notation** For each S-node \( q \) in a correlation-graph \( G = (I \cup S, E) \), we denote \( \text{Tail}_G(q) \) as the set of all incoming I-nodes (or parent nodes) to \( q \), i.e., \( \text{Tail}_G(q) = \{x \in I | x \rightarrow q \in E\} \), and \( \text{Head}_G(q) \) as the I-node supported by \( q \) in \( G \) (or child node), i.e., \( \text{Head}_G(q) \) is the I-node \( x \) where \( q \rightarrow x \in E \). Throughout the paper, we will use \( a \rightarrow b \) instead of \((a, b)\) in order to represent the link from \( a \) to \( b \) in directed graphs.

The parameters of a BKB are given by the conditional probability values associated with its S-nodes. In particular, each S-node \( q \) in the correlation-graph \( G \) of a BKB will be assigned a number \( w(q) \) which is called the weight of \( q \) and serves as the actual conditional probability value \( P(\text{Head}_G(q) | \text{Tail}_G(q)) \).

Hence, a BKB captures a collection of conditional probabilities. However, there are two restrictions on the forms of conditional probabilities represented in BKBs. First, BKBs exclude conditional probabilities of the form \( P(A = a | B = b, B = b', \ldots) \) where \( b \neq b' \), since in such a case the conditioning event becomes empty. Second, BKBs do not allow two conditional probabilities of the form \( P(A = a | I_1) \) and \( P(A = a | I_2) \) in which \( I_1 \) and \( I_2 \) are not mutually exclusive. Two sets of I-nodes, \( I_1 \) and \( I_2 \), are said to be mutually exclusive if there is an I-node \( R = v_1 \) in \( I_1 \) and an I-node \( R = v_2 \) in \( I_2 \) for which \( v_1 \neq v_2 \). For example, the two I-node sets \( \{A = 1, B = 1\} \) and \( \{A = 0, B = 1, C = 0\} \) are mutually exclusive. Two I-node sets which are not mutually exclusive are said to be compatible. Note that the mutual exclusivity of I-node sets implies they are disjoint events in the probability space. Enforcing mutual exclusivity in BKBs is a potential limitation in this representation. However, this can be overcome by increasing the level of detail or specificity of the “rules” involved, when knowledge engineering BKBs for various domains [36]. Finally, with regards to the weights attached to each S-node, to ensure that they are consistent probabilistically [36], any set of S-nodes \( \{q_1, q_2, \ldots, q_n\} \) where \( \text{Head}_G(q_i) = \{R = i\} \) and the tails of the \( q_i \)'s are not mutually exclusive, then the sum of the weights must be less than or equal to 1. For example, for \( q_1 \) with \( \text{Head}_G(q_1) = \{R = \text{Low}\} \) and \( \text{Tail}_G(q_1) = \{A = 0, B = 1\} \), \( q_2 \) with \( \text{Head}_G(q_2) = \{R = \text{Medium}\} \) and \( \text{Tail}_G(q_2) = \{B = 1, C = 0\} \), and \( q_3 \) with \( \text{Head}_G(q_3) = \{R = \text{High}\} \) and \( \text{Tail}_G(q_3) = \{D = 0\} \), \( w(q_1) + w(q_2) + w(q_3) \leq 1 \).

**Definition 2.2.** A **Bayesian knowledge-base** \( K \) is a tuple \((G, w)\), where \( G = (I \cup S, E) \) is a correlation-graph, and \( w \) is a function from \( S \) to \([0, 1]\) such that the following conditions hold:

- For any S-node \( q \in S \), \( \text{Tail}_G(q) \) contains at most one instantiation of each random variable.
- For any two distinct S-nodes \( q_1, q_2 \in S \) that support the same I-node, then \( \text{Tail}_G(q_1) \) and \( \text{Tail}_G(q_2) \) are mutually exclusive. Furthermore, such S-nodes \( q_1 \) and \( q_2 \) are said to be mutually exclusive.
- For any \( Q \subseteq S \) such that (i) \( \text{Head}_G(q_1) \) and \( \text{Head}_G(q_2) \) are mutually exclusive, and (ii) \( \text{Tail}_G(q_1) \) and \( \text{Tail}_G(q_2) \) are not mutually exclusive for all \( q_1 \) and \( q_2 \) in \( Q \),

\[
\sum_{q \in Q} w(q) \leq 1.
\]

For example, in the correlation-graph given in Fig. 1, three S-nodes \( s_7, s_8 \) and \( s_9 \) are pairwise mutually exclusive since they all support the I-node \((C = 1)\), and the three I-node sets \( \text{Tail}_G(s_7) = \{A = 0, B = 0, D = 1\} \), \( \text{Tail}_G(s_8) = \{A = 2, B = 0\} \), and \( \text{Tail}_G(s_9) = \{B = 1\} \) are pairwise mutually exclusive.

Beside the graphical representation, BKBs can also be represented in terms of “if–then” rules, which helps us to better understand the semantics of BKBs. Each S-node \( q \) in a BKB \( K = (G, w) \) corresponds to a conditional probability rule (CPR) [27] of the form \( \text{Tail}_G(q) \Rightarrow \text{Head}_G(q) \), which has the intuitive meaning: if \( \text{Tail}_G(q) \) then \( \text{Head}_G(q) \) with probability \( w(q) \). So a BKB can be described as a set of conditional probability rules (CPRs) corresponding to its S-nodes (see also [37] for a treatment of CPRs). The weight \( w(q) \) will be omitted from the CPRs when we concentrate only on the structure of the BKB. This means the correlation-graph \( G = (I \cup S, E) \) of a BKB can be specified by the set of rules \( \{\text{Tail}_G(q) \Rightarrow \text{Head}_G(q) | q \in S\} \).
As an example, Fig. 2 presents a sample BKB fragment for fresh water aquarium management in terms of conditional probability rules. This sample BKB fragment has the equivalent graphical representation depicted in Fig. 3. The graph structure of our sample BKB is also an example of cyclic dependence among random variables. Such a cycle in the underlying random variable relationships, as shown in Fig. 4, is problematic in Bayesian Networks, but is allowed in our BKB framework. As a final note, BKBs also permit directed cycles in the correlation-graph (see [36]).

2.2. Reasoning with BKBs

To the extent of this work, reasoning with BKBs involves determining the state of all random variables that is most likely to happen conditioned on a given evidence. Reasoning of this type is a fundamental problem in probabilistic reasoning and is referred to as most probable explanation (MPE), or belief revision [25,36,27].

In our BKB setting, we define a state as an I-node set which contains at most one I-node of each random variable. A state is said to be complete if it contains exactly one I-node of each random variable in the given domain. By convention, we assume that the set of random variables appearing in a BKB is also the set of random variables in the domain for which the BKB represents. For simplicity, from now on, we fix our knowledge domain to consist of the random variables $R_1, R_2, \ldots, R_N$. We also assume that the set of I-nodes in a BKB is the set of all I-nodes of those random variables, unless otherwise specified. Hence, a complete state is a set of I-nodes of the form $(R_1 = v_1, R_2 = v_2, \ldots, R_N = v_N)$. An evidence used in reasoning with BKBs is also represented by a state which is usually incomplete. Under this setting, the task of belief revision is to find the most probable complete state that exists and contains the given evidence. Information necessary to

![Fig. 1. Example of correlation-graph of a BKB. The area within the dotted rectangle is an inference.](image1)

![Fig. 2. A sample BKB fragment for fresh water aquarium management.](image2)
compute the probability of such a state in a BKB is captured as a subset of the correlation-graph, called an inference, whose I-node set coincides with the state.

**Definition 2.3.** Let $K = (G, w)$ be a BKB with correlation-graph $G = (I \cup S, E)$. A subgraph $\tau = (I' \cup S', E')$ of $G$ is called an inference over $K$ if

(i) $\tau$ is acyclic.
(ii) (Well-supported) $\forall z \in I', \exists q \in S', q \rightarrow z \in E'$, i.e. each I-node in $\tau$ must have an incoming S-node in $\tau$.
(iii) (Well-founded) $\forall q \in S', \text{Tail}(q) = \text{Tail}(\tau(q))$.
(iv) (Well-defined) $\forall q \in S', \text{Head}(q) = \text{Head}(\tau(q))$.
(v) $I'$ is a state. Thus, $I'$ is referred to as the state of the inference. Furthermore, if $I'$ is a complete state then $\tau$ is said to be a complete inference over $K$.

The well-founded and well-defined properties mean that each conditional probability rule in an inference over a BKB $K$ must be a conditional probability rule of $K$. Hence, an inference over a BKB can be described by a subset of conditional probability rules in the BKB, as long as the corresponding subgraph satisfies conditions (i), (ii) and (v) in the definition of inference.

The subgraph within the dotted rectangle in Fig. 1 is an example of a complete inference which has the complete state $\{A = 2, B = 0, C = 1, D = 0\}$.
The joint probability of an inference $\tau$, denoted by $P(\tau)$, is calculated by multiplying the weights of all S-nodes in the inference, that is, $P(\tau) = \prod_{q \in S} w(q)$ where $S$ is the set of all S-nodes in $\tau$. It has been shown that $P(\tau)$ is also the joint probability of the state of inference $\tau$ [36].

**Remark.** Since a BKB is supposed to represent incomplete knowledge, not every state can be associated with an inference in a BKB. We call a state $\theta$ well-represented in a BKB $K$ if there exists an inference over $K$ whose I-node set coincides with $\theta$. It is also easy to see that any two distinct inferences must have different states.

Reasoning with a BKB can be implemented in numerous ways from A* search to integer linear programming [36,27]. The computational complexity of reasoning with BKBs has been shown to be NP-hard [27]. However, this is worst-case complexity and does not occur in all cases. Furthermore, there is a class of BKBs with worst-case polynomial time reasoning as described in [36].

### 3. BKB validation

The goal of validation for BKBs is to make necessary changes to the given BKB such that the modified BKB will meet all test cases specified by human experts. Before formalizing BKB validation, we need to figure out how an expert system built on BKBs works from the user’s point of view. Basically, the user provides a query, which is evidence, to the inference engine. The inference engine will then perform belief revision over the BKB, and output the solution which is the most probable complete state given the evidence provided. However, users are typically concerned with only a portion of the solution relating to some random variables. Such a portion is referred to as the answer [34]. For example, the user enters the query ($A = 1, B = 0$) and needs the answer for random variable $C$. Suppose the most probable complete state for the evidence ($A = 1, B = 0$) is ($A = 1, B = 0, C = 0, D = 1, E = 1$), then the answer displayed to the user will be $C = 0$. The whole solution is displayed to the user only when he or she needs the explanation of why the system has come up with the answer. This type of inference and explanation is used most often in diagnostic domains such as in medicine. The focus here is on the inference structure that explicitly links the query to the answer, following the if–then chaining within the BKB. In essence, the structure returned represents the diagnostic path explaining the answer that is of highest probability. For example, if a patient exhibits symptoms of chest pains, his doctor will have to diagnose the most likely combination of all diseases and their effects and interdependencies given such evidence. The doctor may find that if “chest-pain = true”, the most likely state of the patient might be a combination of “congestive-heart = true, pneumonia = true, and meningitis = false, etc.” and the links involved. However, the doctor may report only the presence/absence of a certain disease that the patient has concerns about, such as congestive heart failure. In this case, the patient does not care about the exact probability of congestive heart disease given chest pains, but rather just cares about whether he has that disease as a part of his “most probable complete state” of health when he feels chest pains. On the other hand, when the doctor consults with a colleague about the case, the underlying link structure behind his/her diagnosis (from the most probable complete state) is a critical element that must be communicated especially for determining additional tests to confirm other potential interacting diseases and hypothesized links.

We now provide our formal definitions for BKB validation. Each test case is a pair of evidence and expected answer, where evidence is a set of I-nodes and expected answer is a single I-node. Let $Evi(t)$ and $Ans(t)$ denote the evidence and the expected answer of test case $t$, respectively. A test case with evidence $e$ and expected answer ($R = v$), written as $\langle e, R = v \rangle$, expresses the requirement that the knowledge base must return the answer $R = v$ for random variable $R$ if evidence $e$ is used as a query to the inference engine. For example, a test case for a BKB representing a medical domain may have the set of I-nodes {chest-pain = true, blood-pressure = high} as evidence, and have the I-node “congestive-heart = true” as the expected answer. This test case means that if the user enters the query (chest-pain = true, blood-pressure = high) and needs

---

2 There is a subtle distinction here between computing $P(H|E)$ and the most probable complete state containing both $H$ and $E$ where $E$ is the evidence and $H$ is the target answer. Like in BNs [29,28], it is possible in BKBs that $P(H = 1|E)$ might have the highest marginal probability while the most probable complete state contains $H = 0$. We note that the marginal computation is essentially an aggregation of all possible states of the remaining random variables. Since our focus is on diagnosis and explanation, we are interested in the latter and, specifically, the link structure.
to know the answer for random variable “congestive-heart”, then the answer displayed to the user is expected to be “congestive-heart = true”. If the actual answer returned by the knowledge-base system does not match this expected answer, i.e. “congestive-heart = false”, then the system is considered to contain some flaw. Note that, in practice, the expected answer may be a set of I-nodes that contains at most one I-node of each random variable. However, a test case with multiple I-nodes in the expected answer can be replaced by an equivalent set of test cases with a single I-node expected answer. In particular, a test case with evidence \( e \) and multiple I-node expected answers (\( R_1 = v_1, \ldots, R_k = v_k \)) is equivalent to the set of test cases \( \langle e, R_1 = v_1 \rangle, \ldots, \langle e, R_k = v_k \rangle \), because in both cases, the answers are extracted from the same solution, which is the most probable state given evidence \( e \).

We say that a state \( h \) is relevant to test case \( t \) if \( h \) contains both the evidence and expected answer of \( t \). Given a test case \( t \), any complete inference whose state is relevant to \( t \) is called a correct inference of \( t \), while any complete inference whose state contains \( Evi(t) \) and is mutually exclusive with \( Ans(t) \) is called an incorrect inference of \( t \). For example, part (a) and part (b) in Fig. 5 show a correct inference and an incorrect inference (over \( K \)) of test case \( \langle \{A = 0, B = 0\}, C = 0 \rangle \), respectively, where \( K \) is a BKB with the correlation-graph given in Fig. 1. Note that in the case of inference \( \{C = 0, s_6, A = 0, s_1, B = 0, s_3\} \), while having a state relevant to \( t \), it is not a complete inference. It is obvious that the most probable complete state resulting from belief revision for the evidence of a test case \( t \) is either a correct inference or an incorrect inference of \( t \). Moreover, a Bayesian-Knowledge Base meets the requirement expressed by \( t \) if there is a correct inference of \( t \) that has probability greater than the probability of any incorrect inference of \( t \).

Definition 3.1. A Bayesian Knowledge-Base \( K \) is said to satisfy the test case \( t \) if there exists a correct inference \( \tau_c \) over \( K \) of \( t \) such that \( P(\tau_c) > P(\tau) \) for any incorrect inference \( \tau \) over \( K \) of \( t \). Furthermore, such an inference \( \tau_c \) is called a correct solution (over \( K \)) for the test case \( t \).

A Bayesian Knowledge-Base is said to satisfy a set \( T \) of test cases if it satisfies every test case in \( T \). We assume every test case must be valid in the sense that the evidence and the expected answer of each test case must be compatible. If there is a test case \( t \) such that \( Evi(t) \) and \( Ans(t) \) are not compatible, then there will be no state relevant to \( t \), and, as a result, there will be no BKB satisfying \( t \).

4. Validation for BKBs by refining parameters

Our challenge is to find an algorithm for modifying a given BKB such that it will satisfy all test cases specified by the user. Given any set of valid test cases, however, there is no guarantee that there exists some BKB satisfying all test cases of the given set. We start with the assumption that during the initial construction of the BKB, the underlying correlation-graph (network structure) is correct, but there is uncertainty about the correctness of the associated parameters (conditional probabilities). Thus, the validation algorithm we are going to present here is to refine only parameters of a given BKB so that the BKB with updated parameters will satisfy a portion of the given test case set. This algorithm (see Algorithm 1) will serve to determine a sufficient condition for a test case set to be contradiction-free.

Fig. 5. Example of correct inference (a) and incorrect inference (b) of test case \( \langle \{A = 0, B = 0\}, C = 0 \rangle \).
Before presenting the algorithm, we make another assumption that the initial BKB, as an input for Algorithm 1, must contain a correct inference for each test case. If the initial BKB does not contain a correct inference for some test case, the knowledge base is considered to be structurally incomplete at the portion of the domain highlighted by the test case. This incompleteness needs to be handled by another tool [32] because in such a case, there is no way to modify the weights of S-nodes to make the BKB satisfy that test case.

**Algorithm 1 (for validating BKB by refining parameters)**

- **Input:** A BKB \( K = (G, w) \); a set \( T = \{t_1, \ldots, t_n\} \) of test cases. Assume \( K \) contains a correct inference for each test case in \( T \).
- **Output:** A BKB \( K' = (G, w') \) and a subset of \( T \) that \( K' \) satisfies.

1. Initialize \( G_0 \) as an empty graph, and set \( \text{Fail} = \emptyset \).
2. For \( i = 1 \) to \( n \)
   (a) Find a correct inference \( C_i \) (over \( K \)) of \( t_i \) with maximal probability.
   (b) Let\(^3\) \( G_i = G_{i-1} \cup C_i \).
   (c) If \( C_i \) is not a correct solution over \( K \) for \( t_i \), then
      - If \( G_i \) contains an incorrect inference \( \tau \) of \( t_i \) such that \( P(\tau) \geq P(C_i) \), then add \( t_i \) to \( \text{Fail} \). Otherwise, do the followings:
        (i) Let \( M_i \) be the maximal joint probability of any incorrect inference over \( K \) of \( t_i \).
        (ii) For every S-node \( q \) in \( G_i \) but not in \( G_0 \), update \( w(q) \leftarrow w(q) \times \frac{P(C_i)}{M_i} \) where \( \epsilon > 0 \) is a small constant.
3. For every S-node \( q \) in \( G_i \), set \( w'(q) = w(q) \).
4. Return \( K' = (G, w') \) and \( T \setminus \text{Fail} \).

Fig. 6 shows an example of running Algorithm 1 to validate the BKB given in Fig. 1 with the first test case \( t_1 = \{A = 0, B = 0, C = 0\} \). The numbers in regular font are the original weights of S-nodes, and those in bold are values obtained after validating \( t_1 \). The only correct inference \( C_1 \) and the only incorrect inference \( D_1 \) of \( t_1 \) are depicted in part (a) and part (b) of Fig. 5, respectively. So \( G_1 = C_1 \) and \( M_1 = P(D_1) \). The current joint probability of these inferences are \( P(C_1) = 152 \times .750 \times .388 \times .485 \approx .021 \) and \( P(D_1) = 152 \times .750 \times .388 \times .721 \approx .032 \). A simple comparison (\( P(C_1) < M_1 \)) shows that \( C_1 \) is initially not a correct solution for \( t_1 \). Since \( G_1 \) does not contain any incorrect inference of \( t_1 \), the test case \( t_1 \) is not moved to the set \( \text{Fail} \). Instead, the weights of all S-nodes but those in \( G_1 \) are reduced by a multiplicative factor of \( \frac{P(C_1)}{M_1} \approx .568 \), if \( \epsilon \) is chosen to be 0.005. Once these changes are made, the inference \( C_1 \) becomes a correct solution for the test case \( t_1 \).

Recall that Algorithm 1 adjusts only the weights of S-nodes in the BKB, keeping the BKB structure unchanged. This algorithm is similar to the approach presented in [34,30] in the way that the weights of S-nodes are reduced so that the joint probability of the chosen correct inference of a test case becomes greater than the joint probability of any incorrect inference of the test case, and thus the chosen correct inference becomes a correct solution for the test case. The notable difference between the two approaches is with the set of S-nodes to be updated during the validating of a test case \( t_i \). While our algorithm updates only the S-nodes that are not in any correct inference being chosen so far, the approach in [34,30] would update every S-node except for those in the chosen correct inference \( C_i \) of the current test case \( t_i \). The problem with the latter is that \( C_i \) may no longer be a correct solution for \( t_i \) when the next test case is validated. Therefore, there is no guarantee that a test case will be satisfied after the whole validation process terminates. In our approach, \( C_i \) remains a correct solution for the test case \( t_i \) since the weights of S-nodes in \( C_i \) will not be reduced when the later test cases are validated. However, we may encounter a situation where there is an incorrect inference of \( t_i \) which lies entirely inside the union \( G_i \) of all the correct inference being chosen so far and has joint probability greater than or equal to that of \( C_i \). In such a case, due to our policy of not reducing the weights of S-nodes in \( G_i \), we do not guarantee that test case \( t_i \) is validated successfully and we place it into the set \( \text{Fail} \). For this rea-

---

3 The union of two correlation-graphs \( G_1 = (I_1 \cup S_1, E_1) \) and \( G_2 = (I_2 \cup S_2, E_2) \) is a correlation-graph \( G = (I \cup S, E) \) where \( I = I_1 \cup I_2 \), \( S = S_1 \cup S_2 \), \( E = E_1 \cup E_2 \).
son, we choose $C_i$ of maximal probability in order to reduce the chance of encountering a situation that forces us to place $t_i$ into FAIL. By choosing $C_i$ of maximal probability, we also minimize the weight changes in case $t_i$ is validated successfully (i.e., $t_i$ is not placed in FAIL). The soundness of our algorithm is shown in the following lemma, for which the proof can be found in Appendix:

**Lemma 4.1.** The BKB $K'$ returned by Algorithm 1 satisfies every test case that is not in FAIL.

The set of test cases validated successfully by Algorithm 1 depends on the order in which the test cases are validated and the chosen correct inferences (at step 2a). In the following section, we will show that under certain conditions, there is a way to choose the test case order and the correct inferences so that all the test cases will be successfully validated, i.e. the set FAIL is empty when the algorithm terminates.

Clearly, there are computational complexity issues with the algorithm above since it is NP-hard in the worse-case given the simple fact that reasoning over BKBs is NP-hard [27] but that there is also a class of BKBs with worst-case polynomial time reasoning [36]. Our goal here is to demonstrate a feasible and provably sound algorithm for test-case set validation. Thus, although we are trying to find a validation algorithm that can validate correctly all test cases for any input set of test cases, can we be sure that there is such an algorithm? The problem is, given any set of test cases, does there exist a BKB satisfying all test cases of the given set? The solution to these questions are provided in the following section.

Finally, we note that similar to the remark as stated in [34,30], after validation, the weights of S-nodes may drop down to very small values that are far from the original probabilities specified. While this issue is currently outside the scope of this paper, we will now provide some initial discussion here on the problem with potential directions for future work. This issue could be resolved by raising all S-node weights with some single multiplicative factor once the validation is done in order to preserve the validation process. Clearly though, the single multiplicative factor, say $\alpha$, is essentially bounded above by the third condition for BKBs (Definition 2.2). Thus, for any maximal set $Q$ of S-nodes satisfying the third condition, $\alpha \cdot \sum_{q \in Q} w(q) \leq 1$. Thus, it is possible that even after $\alpha$ is applied, the weights for a given set of S-nodes remain very small. If we allowed $\alpha$ to be a function of $Q$, then to avoid undoing the validation, one possibility is to examine the dependencies of the test-cases (as we shall see below) together with the dependencies between required S-node weight modifications. However, if this is infeasible, the issue with the small weights after application of $\alpha$ is likely to be indicative of incompleteness in the structure of the current BKB. Basically, validation of the test-cases has resulted in the fact that additional BKB S-nodes and possibly I-nodes are needed before all test-cases can be satisfied.
In such a case, we could turn the problem over to another tool as we mentioned earlier [32] which could then involve an analysis of the situation resulting from the validation and request aid from a knowledge engineer to resolve the problem.

5. Consistency of test case sets

Suppose for some reason, the validating user provides a test case set which contains two test cases with the same evidences but with mutually exclusive expected answers. For example, \( t_1 = (e, R = 0) \) and \( t_2 = (e, R = 1) \). That is a trivial example of a test case set which is not satisfiable. In this section, we are going to determine if there is some property or condition of a test case set such that there will exist a BKB satisfying it. Throughout this section, we will refer to test case set \( T = \{t_1, \ldots, t_n\} \). We will start with investigating a necessary condition for \( T \) to be satisfiable or contradiction-free. So for now, assume that the test case set \( T \) is satisfied by a BKB \( K \). This means there exists a correct solution \( \tau_i \) over \( K \) for each test case \( t_i \in T \). One difficulty that may prevent successful validation of all test cases is the case where the expected correct solution for one test case is an incorrect inference of another. Therefore, to study properties of the test case set when it is satisfiable, we create a graph \( G_T \) on \( T \) which describes the constraints among the test cases that are tied by the given correct solutions. Precisely, \( G_T = (T, E_T) \), where \( E_T \) is defined as follows:

\[
E_T = \{ t_i \rightarrow t_j | \tau_j \text{ is an incorrect inference of } t_i \}
\]

Lemma 5.1. The graph \( G_T = (T, E_T) \) is acyclic.

Briefly, the acyclicity of graph \( G_T \) is due to the natural ordering among the joint probabilities of correct solutions \( \tau_i \)'s. Each link \( t_i \rightarrow t_j \) in \( G_T \) represents the ordering constraint that the joint probability of the correct solution for \( t_i \) must be greater than that of the correct solution for \( t_j \), i.e. \( P(\tau_i) > P(\tau_j) \).

Although the graph \( G_T \) defined above provides an initial property of the satisfiable test case set, it depends on the structure of the BKB. We need to build another graph on \( T \) that is independent of any BKB structure in order to characterize a necessary condition for \( T \) to be satisfiable. Our new graph will be constructed by ignoring structures of the correct solutions used in the graph \( G_T \) and only utilizing the complete states of those correct solutions to tie the test cases together.

Let \( \theta_j \) be the state of the correct solution \( \tau_j \) for test case \( t_j \). Clearly, \( \theta_j \) is a complete state relevant to \( t_j \). Since \( \theta_j \) is a complete state, it is mutually exclusive with the answer \( \text{Ans}(t_i) \) of another test case \( t_i \) if and only if \( \text{Ans}(t_i) \notin \theta_j \). Thus, the correct solution \( \tau_j \) is an incorrect inference of \( t_i \) if and only if \( \text{Ev}(t_i) \subset \theta_j \) and \( \text{Ans}(t_i) \notin \theta_j \), by definition of incorrect inference. So if we replace the condition defining a link from \( t_i \) to \( t_j \) in the graph \( G_T \) by the condition that \( \text{Ev}(t_i) \subset \theta_j \) and \( \text{Ans}(t_i) \notin \theta_j \), we will obtain another acyclic graph on \( T \) which depends only on the given collection of complete states. This leads to the following definition of consistency graphs over the test case set. A collection\(^4\) of complete states \( \{\theta_1, \ldots, \theta_n\} \) such that \( \theta_j \) is relevant to \( t_j \) for all \( j \) will be referred to as a relevant collection of complete states for the test case set \( T \).

Definition 5.1. Let \( \mathcal{C} = \{\theta_1, \ldots, \theta_n\} \) be a relevant collection of complete states for \( T \). The consistency graph over \( T \) with respect to \( \mathcal{C} \) is defined as a graph \( G_{T, \mathcal{C}} = (T, E_{T, \mathcal{C}}) \), where \( E_{T, \mathcal{C}} \) is determined as follows:

\[
E_{T, \mathcal{C}} = \{ t_i \rightarrow t_j | \text{Ev}(t_i) \subset \theta_j \text{ and } \text{Ans}(t_i) \notin \theta_j \}
\]

For example, suppose \( T \) is the set of test cases given in Table 1 and \( \mathcal{C} \) is the collection of complete states given in Table 2, then the graph shown in Fig. 7 is the consistency graph over \( T \) w.r.t. \( \mathcal{C} \). The order of test cases, \( t_1, \ldots, t_s \), in both tables is a topological sort of those test cases with respect to \( G_{T, \mathcal{C}} \).

\(^4\) A collection of complete states \( \{\theta_1, \ldots, \theta_n\} \) can be viewed as a mapping from \( T \) to the set of complete states which maps each \( t_i \) to \( \theta_i \), since there may be the case that \( \theta_i = \theta_j \) for some \( i \neq j \).
As discussed earlier, if we use the collection of complete states of correct solutions for test cases in $T$ to construct a consistency graph over $T$, then such a consistency graph must be acyclic. This implies the following corollary of Lemma 5.1:

**Corollary 1.** If there exists a BKB satisfying the test case set $T$, then there exists a relevant collection $\mathcal{C}$ of complete states for $T$ such that the consistency graph $G_{T,\mathcal{C}}$ is acyclic.

Corollary 1 establishes the necessary condition for a test case set to be contradiction-free. For convenience, we will give this condition a name:

**Definition 5.2.** A set $T = \{t_1, \ldots, t_n\}$ of test cases is said to be consistent if there exists a relevant collection $\mathcal{C}$ of complete states for $T$ such that the consistency graph over $T$ w.r.t. $\mathcal{C}$ is acyclic.

So it is clear that if a test case set is not consistent then there is no BKB satisfying it. The question arising here is whether the reverse direction holds, i.e., for any consistent set of test cases, does there exist a BKB satisfying it?

Now suppose the test case set $T$ is consistent. Let $\mathcal{C} = \{\theta_1, \ldots, \theta_n\}$ be a relevant collection of complete states for $T$ such that the consistency graph $G_{T,\mathcal{C}}$ is acyclic. Without loss of generality, we can assume that $t_1, t_2, \ldots, t_n$ is a topological order of test cases in $T$ with respect to $G_{T,\mathcal{C}}$. Let $\sigma_k$ denote the set of all I-nodes of random variable $R_k$, i.e., $\sigma_k = \{R_k = v | v$ is a value of $R_k\}$. The following lemma will lead to the existence of a BKB satisfying the given consistent test case set.

**Lemma 5.2.** Let $K = (G, \omega)$ be a BKB with $G = (I \cup S, E)$. If there is a random variable $R_v$ such that every state $\theta_i \setminus \sigma_i$ is well-represented in $K$, then there exists a BKB $K' = (G', \omega')$ with $G' = (I \cup S', E')$ such that $S \subset S'$, $E \subset E'$ and $K'$ satisfies $T$.

The idea of constructing a BKB satisfying $T$ comes from Lemma 4.1. According to that, if for each test case $t_i$ we can choose a correct inference $C_i$ of $t_i$ such that the accumulative correlation-graph $G_i = \bigcup_{j=1}^{i} C_j$ does not contain any incorrect inference of $t_i$, then by applying Algorithm 1 we can adjust the weights of the S-nodes to obtain a new BKB which satisfies the given test case set. So to prove Lemma 5.2, let $\tau_i$ be an inference whose state is $\theta_i \setminus \sigma_i$, we build each correct inference $C_i$ of test case $t_i$ by adding to the inference $\tau_i$ the following items: the I-node $(R_v = v_i)$ which is the instantiation of $R_v$ in $\theta_i$, an S-node $q_i$, the link from $q_i$ to $(R_v = v_i)$, and all links from each I-node in $I_i$ to $q_i$. Note that $q_i$ may be a new S-node or coincident with some S-node $q$ in $G$ such that $q$ supports $(R_v = v_i)$ and that $Tail_{\omega}(q)$ is not mutually exclusive with $I_i$. As an example, Fig. 8 shows how correct inference $C_i$ (in part (b)) is obtained from inference $\tau_i$ (in part (a)), with $G$ being the correlation-graph in Fig. 1, and complete state $\theta_i = \{A = 2, B = 0, C = 1, D = 0\}$. Correlation-graph $G'$ is obtained from $G$ while building those inferences $C_i$’s. Then, using Algorithm 1 with the specified order $t_1, t_2, \ldots, t_n$ of test cases and the constructed correct inferences $C_i$’s, we can assign the weights of S-nodes in $G'$ so that each $C_i$ becomes a correct solution for test case $t_i$.

**Theorem 5.3.** There exists a BKB satisfying the set $T$ of test cases if and only if $T$ is consistent.

By Theorem 5.3, the necessary and sufficient condition for a test case set to be contradiction-free has been established. Unfortunately, detecting whether a test case set meets that condition is NP-complete, as shown in Theorem 5.4.

**Theorem 5.4.** Deciding consistency of test case sets is NP-complete.
Table 2  
Example of relevant complete states of test cases in Table 1

<table>
<thead>
<tr>
<th>i</th>
<th>Complete state $\theta_i$ relevant to $t_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>${ A = 0, B = 1, C = 0, D = 1 }$</td>
</tr>
<tr>
<td>2</td>
<td>${ A = 1, B = 1, C = 1, D = 0 }$</td>
</tr>
<tr>
<td>3</td>
<td>${ A = 2, B = 0, C = 1, D = 0 }$</td>
</tr>
<tr>
<td>4</td>
<td>${ A = 0, B = 0, C = 0, D = 0 }$</td>
</tr>
<tr>
<td>5</td>
<td>${ A = 0, B = 0, C = 1, D = 1 }$</td>
</tr>
</tbody>
</table>

It is easy to see that deciding consistency of test case sets is in NP since a nondeterministic algorithm only needs to guess a collection of complete states relevant to the test cases and check in polynomial time whether the corresponding consistency graph is acyclic. To complete the proof for Theorem 5.4, we will show that deciding consistency of a test case set is NP-hard by the reduction from SAT, which is given in Appendix.

Deciding the consistency of a test case set is intractable because there are too many consistency graphs. In some special cases, however, we are able to determine whether a test case set is consistent without having to examine any consistency graph over it. Our idea for those special cases is to find a lower bound and an upper bound of all consistency graphs over a given test case set. While looking for such bounds, we keep in mind that the best lower (resp. upper) bound should be the intersection (resp. union) of all the consistency graphs over the test case set.

**Proposition 5.1.** For any two test case $t_i, t_j$ in $T$, the link $t_i \rightarrow t_j$ is contained in every consistency graph over $T$ if and only if

![Fig. 7. Example of consistency graph $G_{T,\theta}$ constructed from test cases in Table 1 and their relevant complete states given in Table 2.](image)

![Fig. 8. Proof of Lemma 5.2: Constructing the correct inference $C_i$ (part (b)) from the inference $\tau_i$ (part (a)), in which $R_i$ is $D$, the I-node ($R_i = v_i$) is $D = 0$, and the S-node $s_{11}$ in $G$ is selected to be $q_i$.](image)
(a) \( Evi(t_i) \subset Evi(t_j) \cup \{Ans(t_j)\} \), and
(b) \( Ans(t_i) \) is mutually exclusive with \( Evi(t_j) \cup \{Ans(t_j)\} \).

**Proposition 5.2.** For any two test case \( t_i, t_j \) in \( T \), the link \( t_i \rightarrow t_j \) is contained in some consistency graph over \( T \) if and only if

(c) \( Evi(t_i) \) is compatible with \( Evi(t_j) \cup \{Ans(t_j)\} \), and
(d) \( Ans(t_i) \notin Evi(t_j) \cup \{Ans(t_j)\} \).

It follows from Proposition 5.1 that the intersection of all consistency graphs over \( T \) is the graph on \( T \) denoted \( G^0_T = (T, E^0_T) \), which consists of all links \( t_i \rightarrow t_j \) satisfying the two conditions (a) and (b) in Proposition 5.1. Similarly, the union of all consistency graphs over \( T \) is the graph on \( T \) defined as \( G^1_T = (T, E^1_T) \), where \( E^1_T \) consists of all links \( t_i \rightarrow t_j \) that satisfies the two conditions (c) and (d) in Proposition 5.2.

\[
E^0_T = \left\{ t_i \rightarrow t_j \mid \begin{array}{c}
\text{Evi}(t_i) \subset \text{Evi}(t_j) \cup \{\text{Ans}(t_j)\}, \\
\text{and} \\
\text{Ans}(t_i) \text{ is mutually exclusive with} \\
\text{Evi}(t_j) \cup \{\text{Ans}(t_j)\}
\end{array} \right\}
\]

\[
E^1_T = \left\{ t_i \rightarrow t_j \mid \begin{array}{c}
\text{Evi}(t_i) \text{ is compatible with} \\
\text{Evi}(t_j) \cup \{\text{Ans}(t_j)\}, \\
\text{and} \\
\text{Ans}(t_i) \notin \text{Evi}(t_j) \cup \{\text{Ans}(t_j)\}
\end{array} \right\}
\]

As an illustration, suppose \( T \) is the set of all test cases given in Table 1. Fig. 9 depicts the combination of \( G^0_T, G^1_T \) and the consistency graph \( G_{T,e} \) which has been drawn in Fig. 7. In Fig. 9, all links represented by solid lines are in the consistency graph \( G_{T,e} \).

For any consistency graph \( G_{T,e} = (T, E_{T,e}) \) over \( T \), we have \( E^0_T \subset E_{T,e} \subset E^1_T \). Moreover, \( G^0_T \) is the maximum lower bound and \( G^1_T \) is the minimum upper bound of all consistency graphs over the test case set \( T \). Note that the two lower graph \( G^0_T \) and the upper bound graph \( G^1_T \) are constructed depending only on the test case set \( T \). This yields a polynomial-time method to check the consistency of a the test case set in the following special cases:

(a) If \( G^0_T \) is cyclic, then \( T \) is not consistent.
(b) If \( G^1_T \) is acyclic, then \( T \) is consistent.

In the case where neither \( G^0_T \) is cyclic nor \( G^1_T \) is acyclic, as the case in Fig. 9 for instance, no conclusion about the consistency of the test case set \( T \) can be made, thus we are unable to predict the effectiveness of validation before actually running the validation algorithm.

**6. Thrashing in BKB validation**

The study of consistency of test case sets is originally motivated in an attempt to deal with the problem of thrashing, which has been pointed out in previous work on BKB validation [30,34]. Generally, thrashing occurs if there is a group of inconsistent inequalities on the weights of S-nodes that must be ensured to be true, in order to validate correctly all test cases without modifying the knowledge base structure. A simple form of

![Fig. 9. Example of \( G^0_T \) and \( G^1_T \), where \( T \) is the set of all test cases in Table 1. All lines are in \( G^1_T \), while only bold lines are in \( G^0_T \).](image-url)
such inconsistent inequalities is $w(q_1) > w(q_2)$ and $w(q_1) < w(q_2)$, for some S-nodes $q_1$, $q_2$. In that situation, when the weights of the S-nodes are adjusted to meet those inequality constraints, the knowledge base will oscillate between two states: one of which satisfies $w(q_1) > w(q_2)$ and the other satisfies $w(q_1) < w(q_2)$. This oscillation of the knowledge base may be discovered during validation. However, the phenomenon of knowledge base oscillation is not sufficient evidence for thrashing because it also depends on the parameter refinement strategy. Given the same BKB and the same test case set, the oscillation may take place with a certain refinement method, but does not take place with another. On the other hand, the occurrence of thrashing means the knowledge base will oscillate regardless of refinement strategy.

The first problem is thrashing detection, which is to determine whether thrashing will occur when validating a BKB with a given test case set by refining parameters. Note that this detection must be performed before validation.

### 6.1. Conditions leading to thrashing

To deal with thrashing detection, we first have to define the conditions for both BKBs and test case sets that will lead to thrashing. As stated in [30], the occurrence of thrashing is due to the incompleteness allowed in BKBs. By incompleteness, we mean the lack of an inference graph for some state. However, that statement is incomplete because thrashing is also caused by the inconsistency of the test case set itself. If a test case set $T$ is not consistent then every consistency graph over $T$ is cyclic. Thus, regardless of what inferences are chosen to be expected correct solutions for test cases, the ordering constraints among joint probabilities of those expected correct solutions are also cyclic. This obviously results in inconsistent inequality constraints on the weights of S-nodes, and therefore causes thrashing to occur. So, if the test case set is not consistent then thrashing will always occur even when a complete BKB is given. In other words, the inconsistency of the test case set is a sufficient condition for the occurrence of thrashing.

However, the inconsistency of a test case set is not the necessary condition for thrashing. For consistent test case sets, thrashing still possibly occurs. For example, if the BKB is constructed as in Fig. 10 and the test case set includes two test cases: the first one with evidence “chest-pain = true” and expected answer “congestive-heart = true”, and the second one with evidence “chest-pain = false” and expected answer “congestive-heart = false”, then, the BKB satisfies these two test cases only if the following constraints hold simultaneously: $w(s_5) > w(s_6)$ and $w(s_5) < w(s_6)$. In such a case, thrashing results from incompleteness or incorrectness in the structure of the input BKB. The question arising is how to define a descriptive condition of structural incompleteness and incorrectness of BKBs that will lead to thrashing. Instead of seeking a direct answer for

![Diagram](image_url)

**Fig. 10. Example of a BKB that will cause thrashing in validation with two test cases: ([chest-pain = true], congestive-heart = true) and ([chest-pain = false], congestive-heart = false).**
this difficult question, we will identify the necessary and sufficient condition leading to thrashing in terms of the existence of parameter assignments for the BKB.

Clearly, if thrashing occurs then there is no way to modify the weights of S-nodes to make the BKB satisfy the given test case set, and vice versa. So, the condition leading to thrashing in validation for BKBs can be formally defined as follows:

**Proposition 6.1.** Thrashing occurs in validation for a BKB $K = (G, w)$ with a test case set $T$ if and only if there do not exist any weight assignment $w'$ for S-nodes in $K$ such that the BKB $K' = (G, w')$ satisfies $T$.

Such a weight assignment $w'$ is called the satisfying weight assignment for BKB $K$ with respect to test case set $T$. Hence, the thrashing detection problem becomes deciding if there does not exist a satisfying weight assignment for a BKB with respect to a given test case set.

### 6.2. Resolving thrashing

Due to the potential intractability of the thrashing detection problem, we will not try to detect thrashing. Instead, we go ahead and revise the structure of the knowledge base in such a way that it will guarantee no thrashing. Here, we assume that the given test case set is known to be consistent. Otherwise, there is no way to resolve thrashing as no BKB can satisfy an inconsistent test case set.

Recall that if the test case set is consistent, then there exists a BKB satisfying it. Of course, we may easily create a simple BKB satisfying the given consistent test case set, as shown in the proof of Theorem 5.3. However, the structure of the BKB created in such a way will be far different from that of the original BKB. This may cause the loss of much correct information about the properties of relations among domain entities that has been built up before and represented in the original BKB. The problem here is how to minimally modify the structure of BKB for validation. More formally, suppose a nonnegative function $f(G', G)$ is the measure for the modification of correlation-graph $G$ that results in correlation-graph $G'$, in which $f(G', G) = 0$ if and only if $G'$ and $G$ are identical to each other. Let $K = (G, w)$ be the original BKB and $T$ be the consistent test case set. We need to find a BKB $K' = (G', w')$ such that $K'$ satisfies $T$ and $f(G', G)$ is minimal. Suppose there were a polynomial-time deterministic algorithm for finding such a BKB $K'$, we then could also decide in polynomial time if there exists a satisfying weight assignment for $K$ with respect to $T$ by comparing $G'$ against $G$ and answering yes only if $G'$ is identical to $G$. However, since deciding the existence of a satisfying weight assignment may be too difficult, determining the minimal modification in the structure of BKB for validation is probably also intractable, regardless of how measure $f$ is defined. Therefore, we attempt to find a way to modify the structure of BKB as little as possible prior to adjusting parameters when validating the BKB.

For simplicity, we restrict structural modification of BKBs to the form of adding more structural elements (S-nodes or links). We think that such incremental revision may help retain much of information in the structure of BKB. Let us fix the given input BKB $K = (G, w)$ with $G = (I \cup S, E)$ and a consistent test case set $T$. Our task now is to find a BKB $K' = (G', w')$ such that

(i) $G' = (I \cup S', E')$, where $S \subset S'$ and $E \subset E'$, and
(ii) $K'$ satisfies $T$.

Suppose we have found a relevant collection $\mathcal{C}$ of complete states for $T$ such that the consistency graph over $T$ w.r.t. $\mathcal{C}$ is acyclic. Let $\theta_i$ denote the complete state in $\mathcal{C}$ that is assigned to test case $i \in T$, which means $\theta_i$ is relevant to $i$. We know from **Lemma 5.2** that under certain conditions of $K$ and $\mathcal{C}$, there is a simple way to create a BKB $K'$ satisfying (i) and (ii). That condition is involved in checking the well-represented property of each state $\theta_i$, i.e. the state obtained from the complete state $\theta_i$ by omitting the I-node of some fixed random variable $R$. Therefore, in order to apply the method from **Lemma 5.2**, we develop an algorithm for checking whether a given state $\theta$ is well-represented in the BKB $K$.

The idea of our algorithm is to build up an increasing sequence of well-represented subsets of $\theta$. We can always assume that the empty state is well-represented, so our sequence will start with the emptyset. Now let $\mu$ be a well-represented subset of $\theta$ and let $\tau$ be an inference whose state equals $\mu$. If we add to $\tau$ an I-node
x and an S-node q which supports x and has $\text{Tail}_q(q) \subseteq \mu$ (and also add to $\tau$ the links connecting q to I-nodes in its tail), then we obviously get another inference with larger state. The condition $\text{Tail}_q(q) \subseteq \mu$ is necessary to maintain the acyclicity of the new graph. If x is an I-node and q is an S-node with $\text{Head}_q(q) = x$ and $\text{Tail}_q(q) \subseteq \mu$, we say x is $\mu$-supported by q. Note that since $\mu$ is a state and due to the mutual exclusivity, any I-node x can be $\mu$-supported by at most one S-node. We call x an $\mu$-supported I-node if it is $\mu$-supported by some S-node, in which case such an S-node is unique. The $\mu$-supported property is monotonic in the sense that if $\mu_1 \subset \mu_2$ are states, then any $\mu_1$-supported I-node is also $\mu_2$-supported. So by adding $\mu$-supported I-nodes of $\theta$ into $\mu$, we will get a larger well-represented subset of $\theta$. For this reason, we specify the increasing sequence of well-represented subsets of $\theta$ as follows:

$$\mathcal{I}_0(\theta) = \emptyset$$
$$\mathcal{I}_k(\theta) = \{x \in \theta | x \text{ is } \mathcal{I}_{k-1}(\theta) - \text{ supported}\} \text{ for } k \geq 1$$

The following are examples of such a sequence, where the BKB has the correlation-graph given in Fig. 1.

**Example 1.** Suppose $\theta = \{A = 2, B = 0, C = 1, D = 0\}$. Then we have

$$\mathcal{I}_0(\theta) = \emptyset$$
$$\mathcal{I}_1(\theta) = \{A = 2, B = 0\}$$
$$\mathcal{I}_2(\theta) = \{A = 2, B = 0, C = 1\}$$
$$\mathcal{I}_k(\theta) = \{A = 2, B = 0, C = 1, D = 0\} \quad k = 3, 4, \ldots$$

**Example 2.** Suppose $\theta = \{A = 0, B = 2, C = 0, D = 0\}$. Then we have

$$\mathcal{I}_0(\theta) = \emptyset$$
$$\mathcal{I}_1(\theta) = \{A = 0, B = 2\}$$
$$\mathcal{I}_2(\theta) = \{A = 0, B = 2, C = 0\}$$
$$\mathcal{I}_k(\theta) = \{A = 0, B = 2, C = 0\} \quad k = 3, 4, \ldots$$

A noticeable characteristic of this sequence is that once it stops (strictly) increasing, the later subsets stay the same. In other words, there is a “stopping-point” $k$ such that

$$\mathcal{I}_0(\theta) \subset \mathcal{I}_1(\theta) \subset \cdots \subset \mathcal{I}_k(\theta) = \mathcal{I}_{k+1}(\theta) = \mathcal{I}_{k+2}(\theta) = \cdots \subset \theta$$

The stopping-point of the sequence in Example 1 is 3, and that of the sequence in Example 2 is 2.

**Lemma 6.1.** A state $\theta$ is well-represented in K if and only if there exists $k$ such that $\mathcal{I}_k(\theta) = \emptyset$.

The stopping-point of the sequence $\{\mathcal{I}_k(\theta)\}$ is bounded by the size of $\theta$ (i.e. the number of I-nodes in $\theta$). So by specifying sequence $\{\mathcal{I}_k(\theta)\}$, we have a polynomial-time algorithm to determine if $\theta$ is well-represented. Furthermore, this method also allows us to find the inference associated with $\theta$ in the case $\theta$ is well-represented by recording the S-node q which $\mathcal{I}_{k-1}(\theta)$-supports the I-node x whenever x is added to $\mathcal{I}_{k-1}(\theta)$.

Now we are back to our problem of looking for a BKB $K'$ satisfying (i) and (ii). This can be done by first applying the above algorithm to determine if every state $\theta_i \setminus \sigma_i$ is well-represented in $K$. If there is a random variable $R_i$ for which such condition holds, we modify $K$ to obtain $K'$ as shown in the proof of Lemma 5.2. Otherwise, we may use another tool, which may involve human interaction, to make the BKB more complete so that it contains an inference for each state $\theta_i \setminus \sigma_i$.

So, in constructing such a BKB $K' = (G', w')$ from the input BKB $K = (G, w)$, we have to pass through two stages. The first stage is structural modification which is to construct $G'$ from $G$, and the second stage is parameter refinement, i.e. determining $w'$, using Algorithm 1. The first stage, which is responsible for eliminating thrashing, can be done in polynomial time. However, the second one depends on the time com-
plexity of the reasoning algorithm, which is applied at step 2(c)i to find the incorrect inference with maximum joint probability of each test case. Recall that reasoning with BKBs is NP-hard in the worst case, yet may be efficient in practice and is tractable for a restricted class of BKBs. Thus, if we restrict the BKB to be in such a class, our approach to validation can be performed efficiently.

7. Related work

Approaches and tools for automatic knowledge validation in the literature can be categorized into two groups based on the task they perform: (1) evaluate the knowledge base and (2) detect errors and refine the knowledge base. Knowledge validation tools in the first group, e.g. SAVES [38], KVAT [23] and KJ3 [41], carry out the simplest of automatic jobs. They only run test cases, compare the outputs against the human experts' answers, and then produce a certification about the accuracy of the knowledge base. The tasks of error detection and knowledge refinement in this case are loaded onto knowledge engineers. Our approach presented here falls into the second group. Although, automatic validation in this group is supposed to locate and eliminate errors from the knowledge base based on the evaluation result, almost no system can perform well in this fashion because of the complexity of error detection. Knauf et al. [19,18] proposed a methodology for identifying "guilty rules" in rule based systems and refining the system by fixing those rules. However, it lacks formal analysis for assessing the efficiency and accuracy of their proposed technique.

Our underlying task of correcting the knowledge base to make it satisfy test cases is also the main objective of most research in knowledge refinement, such as SEEK2 [10] and KRUST [9,8]. While many of these systems are built for traditional rule based knowledge without uncertainty, our work deals with graph-based uncertain knowledge. There is no doubt that different models of knowledge representation and inference require different specific techniques of knowledge engineering.

There are a couple of related works in refinement of knowledge bases with uncertainty. Ling and Valtorta [22] study the refinement of uncertain rule bases. This model has similar structure to BKBs, since BKBs can also be viewed as the set of conditional probability rules. However, the inference in uncertain rule bases, which is based on the forward chaining reasoning of normal rule bases, is totally different from that in BKBs. Thus, their technique of refinement cannot be applied to BKBs. Closer work to our BKB validation is the refinement of Bayesian Networks given by Valtorta and Loveland [39]. In that research, the authors also prove the NP-hardness for the problem of finding an assignment of parameters in a BN such that the BN satisfies all cases. However, a case defined in [39] is different from test case in BKB validation. Each case is also a pair of an evidence and the desired answer, but the answer here is the belief of the evidence, which is defined as the conditional probability of the evidence given all available evidence, rather than an instantiation of the concerned random variable, i.e. an I-node.

Finally, with regards to Bayesian Network refinement, Jensen [16] discussed a method for tuning conditional probabilities (or parameters) in a Bayesian Network so that given a variable A and an evidence e, the distribution $P(A|e)$ is close to a desired distribution. To some extent, this is related to our validation algorithm since we also alter conditional probabilities. We could consider applying this tuning method as part of our validation process only if the set of desired distributions is known ahead of time while satisfying the test case set. However, to find such desired distributions may not be easy and requires further study. Another concern with applying this method is that it may work for one desired distribution but may not work when several desired distributions are involved. For example, suppose we want to tune the parameters so that $P(A_1|e_1)$ is close to distribution $y_1$ and $P(A_2|e_2)$ is close to distribution $y_2$. If we apply the tuning algorithm for $P(A_1|e_1)$ and then for $P(A_2|e_2)$, it is not clear how we can guarantee that changing the parameters to obtain $P(A_2|e_2) \approx y_2$ will not damage the earlier $P(A_1|e_1) \approx y_1$.

8. Conclusion

We have opened the black-box of fundamental problems, which directly affect the result of validation for BKBs, including the consistency of test case sets and thrashing. The former is motivated by the question: “Does there exist a BKB satisfying a given test case set?”, while the latter is involved in the question:
“Does there exist a weight assignment for a given BKB so that it makes the BKB satisfy a given test case set?” For the former problem, a key result we have determined is the necessary and sufficient condition for a given test case set to be contradiction-free in the sense that there exists a BKB satisfying it. Thus, to correctly validate all test cases, we must make sure that the given test case set meets that condition, i.e., being consistent, or equivalently, there exists an acyclic consistency graph over the test case set. Naturally, the inconsistency of the test case set is a sufficient reason to cause thrashing. But if the test case set is consistent, thrashing still possibly happens because of incorrectness or incompleteness in the BKB structure. When the test case is known to be consistent, detecting the condition leading to thrashing is necessary in order to determine whether or not the structure of the knowledge base need to be changed in validation.

Unfortunately, we have also found that it is computationally intractable to decide the consistency of a test case set as well as to detect the condition leading to thrashing in validation for a BKB with a consistent test case set. Because of these difficulties, we attempted to address the problems by looking for special cases in which they can be solved in reasonable time. For the consistency of a test case set, we consider only the lower bound and upper bound of all consistent graphs. In some cases, the acyclicity/cyclicity of those bounds are sufficient to determine the consistency of the test case set. The thrashing problem is much more complicated. Thus, instead of detecting the condition leading to thrashing, we ignore it and proceed to refine the BKB, both structurally and parametrically, so long as the refined BKB satisfies the given (consistent) test case set. Under certain conditions, the method for validating a BKB with a consistent test case set that has been presented in the paper works efficiently and correctly validates 100% of test cases.

Appendix A

A.1. Proof of Lemma 4.1

Let $w_0(q)$ denote the initial weight of the S-node $q$ and let $w_i(q)$ denote the weight of the S-node $q$ immediately after the $i$th test case is validated. Let $P_i(\cdot)$ denote the probability defined by the weight function $w_i$, that is $P_i(\tau) = \prod_{q \in S_c} w_i(q)$, where $S_c$ denote the set of S-nodes in a complete inference $\tau$.

Suppose test case $t_i$ is not put into FAIL, we will show that (i) the correct inference $C_i$ becomes a correct solution for $t_i$ immediately after the validating of $t_i$ is done, and that (ii) $C_i$ remains a correct solution for $t_i$ when the algorithm terminates.

According to the algorithm, if $C_i$ is a correct solution of $t_i$ at the time immediately before $t_i$ is validated, i.e. $P_{i-1}(C_i) > P_{i-1}(\tau)$ for every incorrect inference $\tau$ of $t_i$, then nothing is changed during the validating of $t_i$. Thus, in this case, $C_i$ is still a correct solution of $t_i$ when the validating of $t_i$ is done. Now consider the opposite case that $C_i$ is not a correct solution of $t_i$ by the time $t_i$ is validated. This means $M_i \geq P_{i-1}(C_i)$, where $M_i$ is the maximal value of $P_{i-1}(D)$ for any incorrect inference $D$ of $t_i$. Let $D$ be any incorrect solution of $t_i$. For any S-node $q$ in $D$, we have $w_i(q) = w_{i-1}(q) \frac{P_{i-1}(C_i)}{M_i + \epsilon} < w_{i-1}(q)$ if $q$ is not in $G_i = \bigcup_{j=1}^i C_j$, and $w_i(q) = w_{i-1}(q)$ otherwise. Since $t_i$ is not put in FAIL, every incorrect inference of $t_i$ must has an S-node that lies outside of the graph $G_i$. Hence, multiplying the weights of all S-nodes in $D$ gives

$$P_i(D) < P_{i-1}(D) \frac{P_{i-1}(C_i)}{M_i + \epsilon} < P_{i-1}(C_i)$$

The last inequality is due to $P_{i-1}(D) \leq M_i$. Note that the weights of S-nodes in $C_i$ are unchanged, so $P_i(C_i) = P_{i-1}(C_i)$. Hence $P_i(D) < P_i(C_i)$. This shows $C_i$ is a correct solution of $t_i$ at the time the validating of $t_i$ is done.

To show (ii), observe that the S-nodes in $C_i$ will be untouched since $t_i$ is validated. Thus the joint probability of $C_i$ is kept the same since then, i.e. $P_i(C_i) = P_{i+1}(C_i) = \cdots = P_n(C_i)$. On the other hand, no S-node’s weight will ever be increased. Therefore, there is no way that an incorrect solution of $t_i$ can have its joint probability be raised after validating of $t_i$. So $C_i$ will be a correct solution of $t_i$ when the algorithm terminates. 

A.2. Proof of Lemma 5.1

Assume otherwise that $G_T$ has a cycle $t_i \to t_j \to \cdots \to t_m \to t_i$. For any two test cases $t_i$ and $t_j$ such that $t_i \to t_j \in E_T$, we have $P(\tau_i) > P(\tau_j)$, since $\tau_j$ is a correct solution for $t_i$ and $\tau_j$ is an incorrect inference of $t_i$.

Hence

$$P(\tau_i) > P(\tau_j) > \cdots > P(\tau_{m-1}) > P(\tau_m) > P(\tau_i)$$

This contradiction implies that $G_T$ must be acyclic. □

A.3. Proof of Lemma 5.2

Without loss of generality, let $R_N$ be the random variable such that for each $i = 1, \ldots, n$, there exists an inference $\tau_i = (I_i \cup S_i, E_i)$ over $K$, where $I_i = \emptyset \setminus \sigma_N$.

Let $(R_N = v_i)$ be the I-node of $R_N$ that appears in $\theta_i$.

For each $i = 1, \ldots, n$,

- if there exists an S-node $q \in S$ supporting $(R_N = v_i)$ and $\text{Tail}_C(q) \subseteq I_i$, then let $q_i \equiv q$. Note that there do not exist two distinct S-nodes $q, q' \in S$ such that they both support $(R_N = v_i)$, $\text{Tail}_C(q) \subseteq I_i$, and $\text{Tail}_C(q') \subseteq I_i$ because of the mutual exclusivity.
- otherwise, let $q_i$ be a new S-node, i.e. $q_i \not\in S$ and $q_i \neq q_j$ if $i \neq j$.

Let $G' = (I \cup S', E')$, where

$$S' = S \cup \{q_1, q_2, \ldots, q_n\}$$

$$E' = E \cup \{z \to q_i | z \in I_i, 1 \leq i \leq n\} \cup \{q_i \to (R_N = v_i) | 1 \leq i \leq n\}$$

Then $K'' = (G', \omega'')$ is a BKB, in which $\omega''$ is a function from $S'$ to $[0, 1]$ such that $\omega''(q) = \omega(q) \forall q \in S$.

Let $C_i = (I_i \cup \{R_N = v_i\} \cup S_i \cup \{q_i\}, E'_i)$ be the subgraph of $G'$ in which

$$E'_i = E_i \cup \{z \to q_i | z \in I_i\} \cup \{q_i \to (R_N = v_i)\}$$

Thus $C_i$ is a complete inference over the BKB $K''$ that contains $\theta_i \forall i = 1, 2, \ldots, n$. Then $C_i$ is a correct inference of the test case $t_i \forall i = 1, 2, \ldots, n$.

Let $G_i = \bigcup_{j=1}^{n} C_j, i = 1, 2, \ldots, n$. We'll prove that for each $i = 1, \ldots, n$, $G_i$ does not contain any incorrect inference of $t_i$.

Assume otherwise that $G_i$ contains an incorrect inference $\gamma$ of $t_i$ for some $1 \leq i \leq n$. Let $\theta$ be the set of all I-nodes in $\gamma$, thus $\theta$ is a complete state such that

$$\text{Ans}(t_i) \not\in \theta \text{ and } \text{Evi}(t_i) \subseteq \theta$$

(2)

Let $z$ be the instantiation of $R_N$ in $\theta$, and let $q$ be the S-node in $\gamma$ that supports $z$. Thus $q \in \{q_1, q_2, \ldots, q_i\}$. Then $q \equiv q_j$ for some $j \leq i$. Then $\theta_j \not\subseteq \theta$ because $q$ must be well-founded in $\gamma$. It follows that $\theta = \theta_j$. Then $j < i$ since (2) and since $\text{Ans}(t_i) \in \theta$. Thus $t_i \to t_j \not\in E_T$ because of topological order. Thus either $\text{Ans}(t_i) \not\subseteq \theta_j$ or $\text{Evi}(t_i) \not\subseteq \theta_j$ (by (1)). This contradicts (2).

Hence $G_i$ does not contain any incorrect inference of $t_i$ for any $1 \leq i \leq n$. Applying Algorithm 1 for the BKB $K'' = (G', \omega''), \text{ with the order of test cases } t_1, t_2, \ldots, t_n$, choosing $C_i$ as the correct inference of $t_i$ at step 2a, we get the resulted BKB $K'' = (G', \omega')$ which satisfies all test cases in $T$. □

A.4. Proof of Theorem 5.3

The “only if” direction can be proved straightforwardly from Corollary 1. The proof for the “if” side of Theorem 5.3 is taken by simply applying Lemma 5.2 for a BKB with correlation-graph $G = (I \cup S, E)$, in which $S$ contains a unique S-node $q[x]$ corresponding to each I-node $x$ in $I$, and $E$ consists of only links from $q[x]$ to $x$ with $x$ in $I$. Precisely, let $K = (G, \omega)$ be a BKB, where the correlation-graph $G = (I \cup S, E)$ is specified as follows:
• \( I \) is the set of all instantiations of all random variables \( R_1, \ldots, R_N \).
• \( S = \{ q(x) \mid x \in I \} \), where \( q(x) \) denotes a unique \( S \)-node corresponding to \( x \).
• \( E = \{ q(x) \rightarrow x \mid x \in I \} \).

Let \( \mathcal{C} = \{ \Theta_1, \ldots, \Theta_n \} \) be a relevant collection of complete states for \( T \) such that \( G_{T,\mathcal{C}} \) is acyclic. For each \( i = 1, n \), let \( \tau_i = (I_i \cup S_i \cup E_i) \) be a subgraph of \( G \), where

- \( I_i = \emptyset \setminus \sigma_N \)
- \( S_i = \{ q(x) \mid x \in I_i \} \)
- \( E_i = \{ q(x) \rightarrow x \mid x \in I_i \} \)

It is clear that \( \tau_i \) is an inference over \( K \) for any \( i = 1, 2, \ldots, n \). Hence, applying Lemma 5.2 for \( K \) we can imply that there exists a BKB satisfying all test cases in \( T \). \( \square \)

### A.5. Proof of Theorem 5.4

Here we will provide the proof for the NP-hardness of deciding test case set consistency by reduction from the SAT problem.

Let \( \Phi = \bigwedge_{i=1}^{m} c_i \) be an arbitrary formula in CNF over the set of boolean variables \( \{x_1, x_2, \ldots, x_N\} \), where each clause \( c_i \) is assumed not to contain both literals \( x_k \) and \( \bar{x}_k \) for any \( k \).

We specify the test case set \( T = \{t_0, t_1, \ldots, t_n\} \) as follows:

- \( t_0 = (\emptyset, A = \text{yes}) \),
- \( t_i = (e_i, A = \text{no}) \) for all \( i = 1, \ldots, n \), where
  \[ e_i = \{ R_k = 0 \mid 1 \leq k \leq N, x_k \in c_i \} \cup \{ R_k = 1 \mid 1 \leq k \leq N, \bar{x}_k \in c_i \} \]

Here, \( R_k \)'s are random variables taking values 0 and 1. Each \( R_k \) corresponds to the boolean variable \( x_k \), and the I-node \( R_k = 1 \) (resp. \( R_k = 0 \)) will represent the event that “\( x_k \) is true” (resp. false). Random variable \( A \), which has two values yes and no, will be used to represent expected answers of the test cases. Each evidence \( e_i \) represents the event that “clause \( c_i \) is false”.

Note that any consistency graph over \( T \) contains the set of arcs \( \{ t_0 \rightarrow t_i \mid i = 1, \ldots, n \} \) and does not contain any arc between any two nodes in \( \{t_1, \ldots, t_n\} \). It follows that a consistency graph \( G_{T,\mathcal{C}} = (T, E_{T,\mathcal{C}}) \) over \( T \) is acyclic if and only if it does not contain any link from some test case \( t_i \) to \( t_0 \). Recall that

\[ t_i \rightarrow t_0 \in E_{T,\mathcal{C}} \iff \text{Evi}(t_i) \subset \theta_0 \quad \text{and} \quad \text{Ans}(t_i) \notin \theta_0 \quad (3) \]

where \( \theta_0 \) is the state relevant to \( t_0 \) in the collection of complete states \( \mathcal{C} \) on which \( G_{T,\mathcal{C}} \) is built. Since \( \text{Ans}(t_i) \notin \theta_0 \forall i = 1, \ldots, n \),

\[ t_i \rightarrow t_0 \notin E_{T,\mathcal{C}} \iff e_i \notin \theta_0 \quad \forall i = 1, \ldots, n \quad (4) \]

Hence, \( T \) is consistent if and only if there exists a complete state \( \theta_0 \) which contains the I-node \( (A = \text{yes}) \) but does not contains any expected answer \( e_i \), with \( i \neq 0 \).

Consider a complete state \( \theta_0 = \{ A = \text{yes}, R_1 = b_1, \ldots, R_N = b_N \} \), with \( b_1, \ldots, b_N \in \{0, 1\} \), which does not contain any expected answer \( e_i \). Intuitively, the fact that \( \theta_0 \) does not contain \( e_i \) means clause \( c_i \) is true “under \( \theta_0 \)”. This gives the rise to a satisfying truth assignment \( \lambda \) for \( \Phi \), where \( \lambda(x_k) = \text{true} \) if and only if \( b_k = 1 \) for all \( k \). Conversely, if \( \lambda \) is a satisfying truth assignment for \( \Phi \), then we can determine the following complete state \( \theta_0 \) which does not contain any expected answer \( e_i \) with \( i \neq 0 \):

\[ \theta_0 = \{ A = \text{yes} \} \cup \{ R_k = 1 \mid 1 \leq k \leq N, \lambda(x_k) = \text{true} \} \]
\[ \cup \{ R_k = 0 \mid 1 \leq k \leq N, \lambda(x_k) = \text{false} \} \]

Thus, \( T \) is consistent if and only \( \Phi \) is satisfiable, completing the proof. \( \square \)
### Proof of Lemma 6.1

The “if” part is obvious, since we have known that every state in the sequence \( \mathcal{I}_k(\theta) \) is well represented. So now we are going to prove the “only if” part.

Suppose \( \theta = \{x_1, \ldots, x_M\} \) is well represented, and let \( \tau \) be the inference whose state equals \( \theta \). Then the set of S-nodes in \( \tau \) must be of the form \( \{q_1, \ldots, q_M\} \), where \( q_i \) is the unique S-node which \( \theta \)-supports \( x_i \). Hence, if \( x_i \) is \( \mu \)-supported by S-node \( q \) for some subset \( \mu \subset \theta \), then \( q \) must coincide with \( q_i \). It follows that \( x_i \) is added to \( \mathcal{I}_{k-1}(\theta) \) if and only if it is \( \mathcal{I}_k(\theta) \)-supported by \( q_i \) for all \( i, k \). Therefore, the sequence \( \{\mathcal{I}_k(\theta)\} \) can be rewritten as

\[
\mathcal{I}_k(\theta) = \{x_i | \text{Tail}_G(q_i) \subset \mathcal{I}_{k-1}(\theta), 1 \leq j \leq M\} \text{ for } k \geq 1
\]

Since \( \tau \) is acyclic, without loss of generality, assume that \( x_1, x_2, \ldots, x_M \) is a topological order of I-nodes in \( \tau \). Then \( \text{Tail}_G(q_1) = \emptyset \) and \( \text{Tail}_G(q_j) \subset \{x_1, \ldots, x_{j-1}\} \) for any \( j > 1 \).

Let \( k \) be the stopping-point of the sequence \( \{\mathcal{I}_k(\theta)\} \). We have \( x_i \in \mathcal{I}_1(\theta) \subset \mathcal{I}_k(\theta) \). If \( \{x_1, \ldots, x_{j-1}\} \subset \mathcal{I}_k(\theta) \), then \( \text{Tail}_G(q_j) \subset \mathcal{I}_k(\theta) \), which implies \( x_j \in \mathcal{I}_{k+1}(\theta) = \mathcal{I}_k(\theta) \). Hence, by deduction, we get \( x_j \in \mathcal{I}_k(\theta) \) for all \( j \). This shows \( \theta = \mathcal{I}_k(\theta) \). \( \square \)

### References


Hang Dinh is a Ph.D. student in Computer Science and a master student in Mathematics at the University of Connecticut. She received a B.S. in Computer Science from Vietnam National University Hanoi and a M.S. in Computer Science from Asian Institute of Technology. Her current research interests are in theoretical computer science and fundamental artificial intelligence, specifically including heuristic search, computational complexity, algorithm analysis and design, and the use of randomness in computation. She has coauthored two conference papers in ICTAI-04 and AAAI-07.