Information Sciences 320 (2015) 140–155

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Inconsistency-tolerant temporal reasoning with hierarchical information ☆

Norihiro Kamide

Teikyo University, Faculty of Science and Engineering, Department of Information and Electronic Engineering, Toyosatodai 1-1, Utsunomiya, Tochigi 320-8551, Japan

ARTICLE INFO

Article history:
Received 23 October 2013
Received in revised form 2 March 2015
Accepted 6 May 2015
Available online 12 May 2015

Keywords:
Computation tree logic
Paraconsistent logic
Complexity
Inconsistency-tolerant reasoning
Hierarchical information

ABSTRACT

A formal method is proposed for modeling and verifying inconsistency-tolerant temporal reasoning with hierarchical information. For this purpose, temporal logic called sequential paraconsistent computation tree logic (SPCTL) is obtained from computation tree logic by adding a paraconsistent negation connective and some sequence modal operators. SPCTL can appropriately represent both inconsistency-tolerant reasoning by the paraconsistent negation connective and hierarchical information by the sequence modal operators. The validity, satisfiability, and model-checking problems of SPCTL are shown to be EXPTIME-complete, deterministic EXPTIME-complete, and deterministic PTIME-complete, respectively. Illustrative examples for inconsistency-tolerant temporal reasoning with hierarchical information are presented using the proposed SPCTL.

1. Introduction

1.1. Motivations and results

In this paper, we propose a formal method for modeling and verifying inconsistency-tolerant temporal reasoning with hierarchical information. Many logic-based studies have examined handling inconsistency-tolerant reasoning [3], temporal reasoning [6], and reasoning with hierarchical or ontological information [2]. However, to the best of our knowledge, no study has examined integrating these reasoning mechanisms uniformly, i.e., there is no study about inconsistency-tolerant temporal reasoning with hierarchical information. Such a study is required to extend and integrate existing application areas, such as medical diagnosis.

Integration of these useful reasoning mechanisms requires the combination and extension of useful non-classical logics. Combining and extending useful non-classical logics is a very important issue in mathematical logic [4]. Thus, the aim of this paper is to provide a solution for this issue by combining and extending the following useful logics: *temporal logic*, *paraconsistent logic*, and new modal logic (with a new modal operator). By combining and extending the above logics, we can integrate existing application areas using these logics.

In this paper, a new temporal logic called *sequential paraconsistent computation tree logic* (SPCTL), which is an extension of the well-known *computation tree logic* (CTL) [5], is introduced as a Kripke semantics with a paraconsistent negation connective and sequence modal operators. New illustrative examples of modeling and verification are presented using SPCTL. The

☆ This paper includes the contents of the conference presentation [10].

E-mail address: drnkamide08@kpd.biglobe.ne.jp

http://dx.doi.org/10.1016/j.ins.2015.05.015
0020-0255/

validity, satisfiability, and model-checking problems of SPCTL are shown to be EXPTIME-complete, deterministic EXPTIME-complete, and deterministic PTIME-complete, respectively. These complexity results are proved using theorems for embedding SPCTL into a *paraconsistent CTL* (PCTL) and CTL. These embedding and complexity results for SPCTL allow us to use existing CTL-based algorithms to test satisfiability. Thus, it is shown that SPCTL can be used as an executable logic to model and verify inconsistency-tolerant temporal reasoning with hierarchical information.

1.2. Computation tree logic

Model checking [6] is a formal logic-based method for verifying concurrent systems. Specifications about the underlying system are expressed as temporal logic formulas, and efficient algorithms are used to traverse a model defined by the system and determine whether the specification holds or not. CTL [5] is one of the most useful temporal logics for model checking and one of the most important branching-time temporal logics that uses computation trees to specify and verify concurrent systems.

Some CTL-based *model checking* frameworks [6] for verifying systems with hierarchical structures are more efficient and suitable than other types of frameworks, such as those based on *linear-time temporal logic* (LTL) [17] and *full computation tree logic* (CTL*) [9,8]. However, CTL is not suitable for modeling and verifying "inconsistent" systems. Handling inconsistencies in systems requires a *paraconsistent logic* [16] as a base logic for CTL.

An important feature of CTL is that the existence of paths in computation trees can be specified and verified. A computation tree represents a nondeterministic computation or unwinding of a Kripke structure. A Kripke structure is a directed graph; thus, it can naturally express "simple" hierarchical structures. However, it is unsuitable for representing the "highly complex" and "informative" hierarchical structures of ontologies. This is because "normal" trees are not sufficiently expressive for representing such complex structures. Handling highly complex and informative hierarchies (hereafter *hierarchical information*) requires modal operators called the *sequence modal operators* [13].

1.3. Paraconsistent logic

We use *Nelson's four-valued paraconsistent logic* N4 [1,15] as the base logic for CTL. N4 and its variants have been studied extensively (e.g., [11,18,19]) because they have *paraconsistency* [16]. A satisfaction relation $\models$ is said to be paraconsistent with respect to a negation connective $\sim$ if the following condition holds: $\exists\alpha,\ \beta$, not-$[M,s \models (\alpha \wedge \sim\alpha) \rightarrow \beta]$, where s is the state of Kripke structure M. In contrast to N4, classical logic has no paraconsistency because the formula of the form $(\alpha \wedge \sim\alpha) \rightarrow \beta$ is valid in classical logic.

Paraconsistent logics are more appropriate for inconsistency-tolerant reasoning than other non-classical logics. In addition, paraconsistent logics are useful for modeling and representing medical diagnosis as an example of inconsistency-tolerant reasoning [7,14].

Here, the usefulness of paraconsistency is explained. For example, it is undesirable that $(s(x) \wedge \sim s(x)) \rightarrow d(x)$ be satisfied for any symptom s and disease d, where $\sim s(x)$ means "a person x does not have a symptom s" and $d(x)$ means "a person x suffers from a disease d." Assume a large medical knowledge-base *MKB* of symptoms and diseases. It can be assumed that *MKB* is inconsistent in the sense that there is a symptom predicate $s(x)$ such that $\sim s(x),\ s(x) \in MKB$. This assumption is very realistic because symptom is a vague concept that is difficult to determine by any diagnosis; it may be determined true or false by different doctors with different perspectives. Then, the knowledge-base *MKB* does not derive arbitrary disease $d(x)$, which means "a person x suffers from disease d"; thus, paraconsistent logics ensure the fact that, for some formulas α and β, the formula $\alpha \wedge \sim\alpha \rightarrow \beta$ is not valid. Therefore, the paraconsistent logic-based *MKB* is inconsistency-tolerant. In classical logic, the formula $s(x) \wedge \sim s(x) \rightarrow d(x)$ is valid for any disease d; thus, the non-paraconsistent formulation based on classical logic is considered inappropriate for application to a medical knowledge base.

Combining N4 and CTL has been studied previously [12]. The resulting combined logic was called PCTL [12], and it was used to verify a medical check up system. SPCTL is obtained from PCTL by adding sequence modal operators.

1.4. Sequence modal operator

In this paper, we use a sequence modal operator $[b]$ [13], which represents a sequence b of symbols, to describe the ordered labels in a hierarchy. The reason for using the notion of "sequences" in this modal operator is explained below. The notion of "sequences" is fundamental to practical reasoning in computer science because it can represent concepts such as "data sequences," "action sequences," and "time sequences" appropriately. Therefore, the notion of sequences is useful for representing the notions of "information," "trees," "orders," "preferences," and "ontologies." Thus, "hierarchical information" can be represented by sequences. This is plausible because a sequence structure gives a *monoid* $\langle M, ;, \emptyset\rangle$ with *informational interpretation* [19]:

1. M is a set of pieces of (ordered or prioritized) information (i.e., a set of sequences),
2. $;$ is a binary operator (on M) that combines two pieces of information (i.e., it is a concatenation operator on sequences),
3. $\emptyset$ is an empty piece of information (i.e., the empty sequence).