Combining Paraconsistency and Probability in CTL

Norihiro Kamide [1] and Daiki Koizumi [2]
[1]*Teikyo University, Faculty of Science and Engineering, Department of Human Information Systems, Toyosatodai 1-1, Utsunomiya-shi, Tochigi 320-8551, Japan*
[2]*Aoyama Gakuin University, College of Science and Engineering, Department of Electrical Engineering and Electronics, 5-10-1 Fuchinobe, Chuo-ku, Sagamihara-shi, Kanagawa 252-5258, Japan*

Keywords: Computation Tree Logic, Paraconsistent Reasoning, Probabilistic Reasoning, Model Checking.

Abstract: Computation tree logic (CTL) is known to be one of the most useful temporal logics for verifying concurrent systems by model checking technologies. However, CTL is not sufficient for handling inconsistency-tolerant and probabilistic accounts of concurrent systems. In this paper, a paraconsistent (or inconsistency-tolerant) probabilistic computation tree logic (PpCTL) is derived from an existing probabilistic computation tree logic (pCTL) by adding a paraconsistent negation connective. A theorem for embedding PpCTL into pCTL is proven, which indicates that we can reuse existing pCTL-based model checking algorithms. Some illustrative examples involving the use of PpCTL are also presented.

1 INTRODUCTION

The verification of open, large, randomized, and stochastic concurrent systems is gaining increasing importance in the fields of computer science and engineering. On one hand, verifying open and large concurrent systems, such as web application systems, requires the handling of inconsistency-tolerant (or paraconsistent) reasoning because inconsistencies often appear and are inevitable in such systems (Chen and Wu, 2006). On the other hand, verifying randomized and stochastic concurrent systems, such as fault-tolerant communication systems over unreliable channels, requires the handling of probabilistic reasoning because useful notions of reliability for such systems require probabilistic characterization (Bianco and de Alfaro, 1995). Thus, handling both inconsistency-tolerant and probabilistic reasoning by an appropriate logic is a requirement for verifying such complex concurrent systems.

Computation tree logic (CTL) (Clarke and Emerson, 1981) is widely accepted as one of the most useful temporal logics for verifying concurrent systems by *model checking* technologies (Clarke et al., 1999). CTL-based model checking algorithms are known to be more efficient than model-checking algorithms based on other temporal logics such as *linear-time temporal logic* (LTL) (Pnueli, 1977). However, CTL is not sufficient for handling paraconsistent and probabilistic accounts of concurrent systems because it has no operators that can represent paraconsistency and probability. Thus, the aim of this paper is to construct a paraconsistent and probabilistic extension of CTL. To achieve this aim, a new logic, *paraconsistent probabilistic CTL* (PpCTL), is introduced. To construct PpCTL, the existing useful CTL-variants, namely *paraconsistent CTL* (PCTL) (Kamide and Kaneiwa, 2010; Kaneiwa and Kamide, 2011) and *probabilistic CTL* (pCTL) (Aziz et al., 1995; Bianco and de Alfaro, 1995), are combined on the basis of a theorem for embedding PpCTL into pCTL. Some illustrative examples describing an *SQL injection* attack detection algorithm (Sonoda et al., 2011) that involves the use of PpCTL are also presented in this paper to highlight the virtues of combining paraconsistency (in PCTL) and probability (in pCTL).

Integrating useful reasoning mechanisms is regarded as combining and extending some useful non-classical logics such as *modal logics*. Combining and extending useful non-classical logics are also known to be a very important issue in mathematical logic (see e.g., (Carnielli et al., 2008)). This paper is thus also intended to give a solution for this issue, combining and extending the following useful non-classical logics: *temporal logic*, *paraconsistent (or inconsistency-tolerant) logic* and *probabilistic (or probability) logic*. The proposed embedding-based method is not so technically innovative, but gives a new simple and useful combination mechanisms for these logics. By combining and extending these log-

Kamide N. and Koizumi D..
Combining Paraconsistency and Probability in CTL.
DOI: 10.5220/0005180402850293
In *Proceedings of the International Conference on Agents and Artificial Intelligence* (ICAART-2015), pages 285-293
ISBN: 978-989-758-074-1

ics, we can integrate the existing two application areas concerning PCTL and pCTL, respectively.

PCTL, which was introduced and studied by Kamide and Kaneiwa in (Kamide and Kaneiwa, 2010; Kaneiwa and Kamide, 2011), is a paraconsistent extension of CTL. To appropriately formalize inconsistency-tolerant reasoning, PCTL is based on *Nelson's four-valued paraconsistent logic* N4 (Almukdad and Nelson, 1984; Nelson, 1949), which includes a paraconsistent negation connective. The paraconsistent negation connective in PCTL entails the property of *paraconsistency*. Roughly, a satisfaction relation $\models$ is considered 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 a Kripke structure M. In contrast to PCTL, classical logic has no paraconsistency because the formula of the form $(\alpha\wedge\sim\alpha)\rightarrow\beta$ is valid in classical logic.

Paraconsistent logics, including PCTL, are known to be more appropriate for inconsistency-tolerant and uncertain reasoning than other non-classical logics (Priest and Routley, 1982; Wansing, 1993; Kamide and Wansing, 2012). For example, the following scenario is undesirable: $(s(x)\wedge\sim s(x))\rightarrow d(x)$ is valid for any symptom s and disease d, where $\sim s(x)$ implies that "a person x does not have a symptom s" and $d(x)$ implies that "a person x suffers from a disease d." An inconsistent scenario expressed as $melancholia(john)\wedge\sim melancholia(john)$ will inevitably occur because melancholia is an uncertain concept and the fact "John has melancholia" may be determined to be true or false by different pathologists with different perspectives. In this case, the undesirable formula $(melancholia(john)\wedge\sim melancholia(john))\rightarrow cancer(john)$ is valid in classical logic (i.e., an inconsistency has an undesirable consequence), whereas it is not valid in paraconsistent logics (i.e., these logics are inconsistency-tolerant).

We now give a detailed explanation about the usefulness of paraconsistent reasoning. We assume a large medical database *MDB* of symptoms and diseases. We can also assume that *MDB* is inconsistent in the sense that there is a symptom predicate $s(x)$ such that $\sim s(x), s(x)\in MDB$. This assumption is regarded as very realistic, because symptom is an uncertain concept, which is difficult to determine by any diagnosis. It may be determined to be true or false by different doctors with different perspectives. Then, the database *MDB* does not derive arbitrary disease $d(x)$, which means "a person x suffers form a disease d", since paraconsistent logics ensures the fact that for some formulas α and β, the formula $\sim\alpha\wedge\alpha\rightarrow\beta$ is not valid. The paraconsistent logic-based large *MDB* is thus inconsistency-tolerant. In the classical logic, the formula $\sim s(x)\wedge s(x)\rightarrow d(x)$ is valid for any disease d, and hence the non-paraconsistent formulation based on classical logic is regarded as inappropriate to the application of this medical database. Apart from such a medical database, large and open concurrent systems also require the handling of paraconsistent scenarios because inconsistencies often appear and are inevitable in these systems. This is a reason why we need to combine PCTL and pCTL.

pCTL, which was introduced and studied by Aziz et al. in (Aziz et al., 1995) and Bianco and de Alfaro in (Bianco and de Alfaro, 1995), is a probabilistic extension of CTL. To appropriately formalize probabilistic reasoning, pCTL uses a *probabilistic* or *probability operator* $P_{\geq x}$, where the formula of the form $P_{\geq x}\alpha$ is intended to read "the probability of α holding in the future evolution of the system is at least x." In (Bianco and de Alfaro, 1995), pCTL and its extension, pCTL*, were introduced for verifying the properties of reliability and the performance of the systems modeled by *discrete Markov chains*. pCTL and pCTL* can appropriately express quantitative bounds on the probability of system evolutions. In addition, in (Bianco and de Alfaro, 1995), the complexities of model-checking algorithms for pCTL and pCTL* were clarified. In (Aziz et al., 1995), model-checking algorithms for the extensions of the abovementioned settings of pCTL and pCTL* were proposed for verifying probabilistic nondeterministic concurrent systems, in which the probabilistic behavior coexists with nondeterminism. These algorithms were also shown to exhibit polynomial-time complexity depending on the size of the systems.

The main difference between the pCTL settings by Aziz et al. (Aziz et al., 1995) and Bianco and de Alfaro (Bianco and de Alfaro, 1995) is the setting of the *probability measures* in the *probabilistic Kripke structures* of pCTL. In the present paper, PpCTL is constructed on the basis of a "probability-measure-independent" translation of PpCTL into pCTL. By this translation, a theorem for embedding PpCTL into pCTL is proven, which entails the relative decidability of PpCTL with respect to pCTL, i.e., the decidability of pCTL implies that of PpCTL. This fact indicates that we can reuse the existing pCTL-based verification algorithms by Aziz et al. (Aziz et al., 1995) and Bianco and de Alfaro (Bianco and de Alfaro, 1995)

The structure of this paper is as follows. In Section 2, the new logic PpCTL, which is an extension of both PCTL and pCTL, is introduced on the basis of a *paraconsistent probabilistic Kripke structure* with two types of satisfaction relations. Some remarks on

PpCTL are also provided in this section. In Section 3, a theorem for embedding PpCTL into pCTL is proven using a new translation function. As a corollary of this embedding theorem, a relative decidability theorem for PpCTL, wherein the decidability of pCTL implies that of PpCTL, is obtained. Note that the proposed translation is regarded as a modified extension of the existing translation, which was used by Gurevich (Gurevich, 1977), Rautenberg (Rautenberg, 1979), and Vorob'ev (Vorob'ev, 1952) to embed *Nelson's three-valued constructive logic* (Almukdad and Nelson, 1984; Nelson, 1949) into *positive intuitionistic logic*. In Section 4, some illustrative examples for describing the SQL injection attack detection algorithm proposed by Sonoda et al. (Sonoda et al., 2011) are presented on the basis of the use of PpCTL-formulas. In Section 5, this paper is concluded and some related works are addressed.

2 LOGICS

Formulas of PpCTL are constructed from countably many atomic formulas, $\rightarrow$ (implication) $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (classical negation), $\sim$ (paraconsistent negation), $\mathrm{P}_{\leq x}$ (less than or equal probability), $\mathrm{P}_{\geq x}$ (greater than or equal probability), $\mathrm{P}_{\prec x}$ (less than probability), $\mathrm{P}_{\succ x}$ (greater than probability), X (next), G (globally), F (eventually), U (until), R (release), A (all computation paths) and E (some computation path). The symbols X, G, F, U and R are called *temporal operators*, and the symbols A and E are called *path quantifiers*. The symbols $\mathrm{P}_{\leq x}$, $\mathrm{P}_{\geq x}$, $\mathrm{P}_{\prec x}$ and $\mathrm{P}_{\succ x}$ are called *probabilistic operators* or *probability operators*. A formula $\mathrm{P}_{\leq x}\alpha$ is intended to read "the probability of α is at least x." The symbol ATOM is used to denote the set of atomic formulas. An expression $A \equiv B$ is used to denote the syntactical identity between A and B. An expression $\alpha \leftrightarrow \beta$ is used to represent $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$.

Definition 2.1. *Formulas α are defined by the following grammar, assuming $p \in$ ATOM and $x \in [0,1]$:*

$$\begin{aligned}\alpha ::= {} & p \mid \alpha \rightarrow \alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \neg\alpha \mid \sim\alpha \mid \\ & \mathrm{P}_{\leq x}\alpha \mid \mathrm{P}_{\geq x}\alpha \mid \mathrm{P}_{\prec x}\alpha \mid \mathrm{P}_{\succ x}\alpha \mid \mathrm{AX}\alpha \mid \\ & \mathrm{EX}\alpha \mid \mathrm{AG}\alpha \mid \mathrm{EG}\alpha \mid \mathrm{AF}\alpha \mid \mathrm{EF}\alpha \mid \\ & \mathrm{A}(\alpha \mathrm{U} \alpha) \mid \mathrm{E}(\alpha \mathrm{U} \alpha) \mid \mathrm{A}(\alpha \mathrm{R} \alpha) \mid \mathrm{E}(\alpha \mathrm{R} \alpha).\end{aligned}$$

Note that pairs of symbols like AG and EU are indivisible, and that the symbols $\mathrm{X}, \mathrm{G}, \mathrm{F}, \mathrm{U}$ and R cannot occur without being preceded by an A or an E. Similarly, every A or E must have one of X, G, F, U and R to accompany it. It is remarked that all the connectives displayed above are required to obtain a theorem for embedding PpCTL into pCTL.

Definition 2.2. *A* paraconsistent probabilistic Kripke structure *(*ppk-structure *for short) is a structure $\langle S, S_0, R, \mu_s, L^+, L^- \rangle$ such that*

1. *S is the set of states,*
2. *S_0 is a set of initial states and $S_0 \subseteq S$,*
3. *R is a binary relation on S which satisfies the condition: $\forall s \in S\ \exists s' \in S\ [(s,s') \in R]$,*
4. *μ_s is a certain probability measure (or probability distribution) concerning $s \in S$: a set of paths beginning at s is mapped into a real number in $[0,1]$,*
5. *L^+ and L^- are mappings from S to the power set of a nonempty subset* AT *of* ATOM.

A path *in a ppk-structure is an infinite sequence of states, $\pi = s_0, s_1, s_2, \ldots$ such that $\forall i \geq 0\ [(s_i, s_{i+1}) \in R]$. The symbol Ω_s is used to denote the set of all paths beginning at s.*

Some remarks on the ppk-structure defined above are given as follows.

1. The definition of μ_s is not precisely and explicitly given in this paper since the proposed translation from PpCTL into pCTL is independent of the setting of μ_s.
2. There are many possibilities for defining a probability measure μ_s. Some typical examples of probability measures are addressed as follows.
   (a) In (Bianco and de Alfaro, 1995), two probability measures μ_s^+ and μ_s^-, called *minimal probability* and *maximal probability*, respectively, are adopted in pCTL.
   (b) In (Aziz et al., 1995), a probability measure μ^s concerning some *discrete Markov processes* and *discrete generalized Markov processes* is adopted in pCTL.
3. The probability measures μ_s^+ and μ_s^- used in (Bianco and de Alfaro, 1995) are defined on a Borel σ-algebra $\mathcal{B}_s$ ($\subseteq 2^{\Omega_s}$) as follows: for any $\Delta \in \mathcal{B}_s$, $\mu_s^+(\Delta) = sup\ \mu_{s,\eta}(\Delta)$ and $\mu_s^-(\Delta) = inf\ \mu_{s,\eta}(\Delta)$ where $\mu_{s,\eta}$ with a strategy η concerning nondeterminism is a unique probability measure on $\mathcal{B}_s$.
4. The probability measure μ^s used in (Aziz et al., 1995) is defined as a mapping from $\mathcal{C}^s$ into $[0,1]$ where $\mathcal{C}^s$ is a Borel sigma field, which is the class of subsets of the set of all infinite state sequences starting at s.

Definition 2.3 (PpCTL)**.** *Let* AT *be a nonempty subset of* ATOM. Satisfaction relations $\models^+$ *and* $\models^-$ *on a ppk-structure $M = \langle S, S_0, R, \mu_s, L^+, L^- \rangle$ are defined inductively as follows (s represents a state in S):*

1. *for any $p \in$ AT, $M, s \models^+ p$ iff $p \in L^+(s)$,*

2. $M,s \models^+ \alpha_1 \rightarrow \alpha_2$ *iff* $M,s \models^+ \alpha_1$ *implies* $M,s \models^+ \alpha_2$,
3. $M,s \models^+ \alpha_1 \wedge \alpha_2$ *iff* $M,s \models^+ \alpha_1$ *and* $M,s \models^+ \alpha_2$,
4. $M,s \models^+ \alpha_1 \vee \alpha_2$ *iff* $M,s \models^+ \alpha_1$ *or* $M,s \models^+ \alpha_2$,
5. $M,s \models^+ \neg\alpha_1$ *iff not-*$[M,s \models^+ \alpha_1]$,
6. $M,s \models^+ \sim\alpha$ *iff* $M,s \models^- \alpha$,
7. *for any* $x \in [0,1]$, $M,s \models^+ \mathrm{P}_{\leq x}\alpha$ *iff* $\mu_s(\{w \in \Omega_s \mid M,s \models^+ \alpha\}) \leq x$,
8. *for any* $x \in [0,1]$, $M,s \models^+ \mathrm{P}_{\geq x}\alpha$ *iff* $\mu_s(\{w \in \Omega_s \mid M,s \models^+ \alpha\}) \geq x$,
9. *for any* $x \in [0,1]$, $M,s \models^+ \mathrm{P}_{\prec x}\alpha$ *iff* $\mu_s(\{w \in \Omega_s \mid M,s \models^+ \alpha\}) < x$,
10. *for any* $x \in [0,1]$, $M,s \models^+ \mathrm{P}_{\succ x}\alpha$ *iff* $\mu_s(\{w \in \Omega_s \mid M,s \models^+ \alpha\}) > x$,
11. $M,s \models^+ \mathrm{AX}\alpha$ *iff* $\forall s_1 \in S \quad [(s,s_1) \in R$ *implies* $M,s_1 \models^+ \alpha]$,
12. $M,s \models^+ \mathrm{EX}\alpha$ *iff* $\exists s_1 \in S \quad [(s,s_1) \in R$ *and* $M,s_1 \models^+ \alpha]$,
13. $M,s \models^+ \mathrm{AG}\alpha$ *iff for all paths* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and all states* s_i *along* π, *we have* $M,s_i \models^+ \alpha$,
14. $M,s \models^+ \mathrm{EG}\alpha$ *iff there is a path* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and for all states* s_i *along* π, *we have* $M,s_i \models^+ \alpha$,
15. $M,s \models^+ \mathrm{AF}\alpha$ *iff for all paths* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *there is a state* s_i *along* π *such that* $M,s_i \models^+ \alpha$,
16. $M,s \models^+ \mathrm{EF}\alpha$ *iff there is a path* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and for some state* s_i *along* π, *we have* $M,s_i \models^+ \alpha$,
17. $M,s \models^+ \mathrm{A}(\alpha_1 \mathrm{U} \alpha_2)$ *iff for all paths* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *there is a state* s_k *along* π *such that* $[(M,s_k \models^+ \alpha_2)$ *and* $\forall j\ (0 \leq j < k$ *implies* $M,s_j \models^+ \alpha_1)]$,
18. $M,s \models^+ \mathrm{E}(\alpha_1 \mathrm{U} \alpha_2)$ *iff there is a path* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and for some state* s_k *along* π, *we have* $[(M,s_k \models^+ \alpha_2)$ *and* $\forall j\ (0 \leq j < k$ *implies* $M,s_j \models^+ \alpha_1)]$,
19. $M,s \models^+ \mathrm{A}(\alpha_1 \mathrm{R} \alpha_2)$ *iff for all paths* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and all states* s_j *along* π, *we have* $[\forall i < j$ *not-*$[M,s_i \models^+ \alpha_1]$ *implies* $M,s_j \models^+ \alpha_2]$,
20. $M,s \models^+ \mathrm{E}(\alpha_1 \mathrm{R} \alpha_2)$ *iff there is a path* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and for all states* s_j *along* π, *we have* $[\forall i < j$ *not-*$[M,s_i \models^+ \alpha_1]$ *implies* $M,s_j \models^+ \alpha_2]$,
21. *for any* $p \in \mathrm{AT}$, $M,s \models^- p$ *iff* $p \in L^-(s)$,
22. $M,s \models^- \alpha_1 \rightarrow \alpha_2$ *iff* $M,s \models^+ \alpha_1$ *and* $M,s \models^- \alpha_2$,
23. $M,s \models^- \alpha_1 \wedge \alpha_2$ *iff* $M,s \models^- \alpha_1$ *or* $M,s \models^- \alpha_2$,
24. $M,s \models^- \alpha_1 \vee \alpha_2$ *iff* $M,s \models^- \alpha_1$ *and* $M,s \models^- \alpha_2$,
25. $M,s \models^- \neg\alpha_1$ *iff* $M,s \models^+ \alpha_1$,
26. $M,s \models^- \sim\alpha_1$ *iff* $M,s \models^+ \alpha_1$,
27. *for any* $x \in [0,1]$, $M,s \models^- \mathrm{P}_{\leq x}\alpha$ *iff* $\mu_s(\{w \in \Omega_s \mid M,s \models^- \alpha\}) > x$,
28. *for any* $x \in [0,1]$, $M,s \models^- \mathrm{P}_{\geq x}\alpha$ *iff* $\mu_s(\{w \in \Omega_s \mid M,s \models^- \alpha\}) < x$,
29. *for any* $x \in [0,1]$, $M,s \models^- \mathrm{P}_{\prec x}\alpha$ *iff* $\mu_s(\{w \in \Omega_s \mid M,s \models^- \alpha\}) \geq x$,
30. *for any* $x \in [0,1]$, $M,s \models^- \mathrm{P}_{\succ x}\alpha$ *iff* $\mu_s(\{w \in \Omega_s \mid M,s \models^- \alpha\}) \leq x$,
31. $M,s \models^- \mathrm{AX}\alpha$ *iff* $\exists s_1 \in S \quad [(s,s_1) \in R$ *and* $M,s_1 \models^- \alpha]$,
32. $M,s \models^- \mathrm{EX}\alpha$ *iff* $\forall s_1 \in S \quad [(s,s_1) \in R$ *implies* $M,s_1 \models^- \alpha]$,
33. $M,s \models^- \mathrm{AG}\alpha$ *iff there is a path* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and for some state* s_i *along* π, *we have* $M,s_i \models^- \alpha$,
34. $M,s \models^- \mathrm{EG}\alpha$ *iff for all paths* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *there is a state* s_i *along* π *such that* $M,s_i \models^- \alpha$,
35. $M,s \models^- \mathrm{AF}\alpha$ *iff there is a path* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and for all states* s_i *along* π, *we have* $M,s_i \models^- \alpha$,
36. $M,s \models^- \mathrm{EF}\alpha$ *iff for all paths* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and all states* s_i *along* π, *we have* $M,s_i \models^- \alpha$,
37. $M,s \models^- \mathrm{A}(\alpha_1 \mathrm{U} \alpha_2)$ *iff there is a path* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and for all states* s_j *along* π, *we have* $[\forall i < j$ *not-*$[M,s_i \models^- \alpha_1]$ *implies* $M,s_j \models^- \alpha_2]$,
38. $M,s \models^- \mathrm{E}(\alpha_1 \mathrm{U} \alpha_2)$ *iff for all paths* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and for all states* s_j *along* π, *we have* $[\forall i < j$ *not-*$[M,s_i \models^- \alpha_1]$ *implies* $M,s_j \models^- \alpha_2]$,
39. $M,s \models^- \mathrm{A}(\alpha_1 \mathrm{R} \alpha_2)$ *iff there is a path* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *and for some state* s_k *along* π, *we have* $[(M,s_k \models^- \alpha_2)$ *and* $\forall j\ (0 \leq j < k$ *implies* $M,s_j \models^- \alpha_1)]$,
40. $M,s \models^- \mathrm{E}(\alpha_1 \mathrm{R} \alpha_2)$ *iff for all paths* $\pi \equiv s_0,s_1,s_2,...,$ *where* $s \equiv s_0$, *there is a state* s_k *along* π *such that* $[(M,s_k \models^- \alpha_2)$ *and* $\forall j\ (0 \leq j < k$ *implies* $M,s_j \models^- \alpha_1)]$.

Definition 2.4. *A formula* α *is* valid *(*satisfiable*) in* PpCTL *if* $M,s \models^+ \alpha$ *holds for any (some) ppk-structure* $M = \langle S,S_0,R,\mu_s,L^+,L^- \rangle$, *any (some)* $s \in S$, *and any (some) satisfaction relations* $\models^+$ *and* $\models^-$ *on* M.

Definition 2.5. *Let* M *be a ppk-structure* $\langle S,S_0,R,\mu_s,L^+,L^- \rangle$ *for* PpCTL, *and* $\models^+$ *and* $\models^-$ *be satisfaction relations on* M. *Then, the* positive and negative model checking problems *for* PpCTL *are respectively defined by: for any formula* α, *find the sets* $\{s \in S \mid M,s \models^+ \alpha\}$ *and* $\{s \in S \mid M,s \models^- \alpha\}$.

Some remarks on PpCTL are given as follows.

1. The intuitive meanings of $\models^+$ and $\models^-$ in PpCTL are "verification (or justification)" and "refutation (or falsification)", respectively (Wansing, 1993; Kamide and Wansing, 2012).
2. PpCTL is regarded as a paraconsistent logic. This is explained as follows. Assume a ppk-structure $M = \langle S, S_0, R, \mu_s, L^+, L^- \rangle$ such that $p \in L^+(s)$, $p \in L^-(s)$ and $q \notin L^+(s)$ for any distinct atomic formulas p and q. Then, $M, s \models^+ (p \wedge \sim p) \rightarrow q$ does not hold, and hence $\models^+$ in PpCTL is paraconsistent with respect to $\sim$. For more information on paraconsistency, see e.g., (Priest and Routley, 1982).
3. The positive model checking problem for PpCTL corresponds to the standard "verification-based" model checking problem for pCTL. The negative model checking problem for PpCTL corresponds to the dual of positive one. i.e., it is regarded as a "refutation-based" model checking problem. Both the positive and negative model checking should simultaneously be performed, i.e., only one of them cannot be performed.

Proposition 2.6. *The following formulas concerning probabilistic operators are valid in* PpCTL*: for any formula* α*,*

1. $\sim \mathrm{P}_{\leq x}\alpha \leftrightarrow \mathrm{P}_{\succ x}\sim\alpha$,
2. $\sim \mathrm{P}_{\geq x}\alpha \leftrightarrow \mathrm{P}_{\prec x}\sim\alpha$,
3. $\sim \mathrm{P}_{\prec x}\alpha \leftrightarrow \mathrm{P}_{\geq x}\sim\alpha$,
4. $\sim \mathrm{P}_{\succ x}\alpha \leftrightarrow \mathrm{P}_{\leq x}\sim\alpha$,

Proof. Suppose that $M = \langle S, S_0, R, \mu_s, L^+, L^- \rangle$ is an arbitrary ppk-structure, and that $\models^+$ and $\models^-$ are any satisfaction relations on M. We only show the following case.

(1): We show only that $\sim \mathrm{P}_{\leq x}\alpha \rightarrow \mathrm{P}_{\succ x}\sim\alpha$ is valid in PpCTL. Let s be an arbitrary element of S. Then, we show $M, s \models^+ \sim \mathrm{P}_{\leq x}\alpha \rightarrow \mathrm{P}_{\succ x}\sim\alpha$. To show this, we show that $M, s \models^+ \sim \mathrm{P}_{\leq x}\alpha$ implies $M, s \models^+ \mathrm{P}_{\succ x}\sim\alpha$. Suppose $M, \models^+ \sim \mathrm{P}_{\leq x}\alpha$. Then, we obtain the required fact as follows: $M, \models^+ \sim \mathrm{P}_{\leq x}\alpha$ iff $M, \models^- \mathrm{P}_{\leq x}\alpha$ iff $\mu_s(\{w \in \Omega_s \mid M, w \models^- \alpha\}) > x$ iff $\mu_s(\{w \in \Omega_s \mid M, w \models^+ \sim\alpha\}) > x$ iff $M, s \models^+ \mathrm{P}_{\succ x}\sim\alpha$. **Q.E.D.**

Definition 2.7 (pCTL)**.** *A* probabilistic Kripke structure *(*pk-structure *for short) for* pCTL *is a structure* $\langle S, S_0, R, \mu_s, L \rangle$ *such that*

1. S, S_0, R *and* μ_s *are the same as those in Definition 2.2,*
2. L *is a mapping from* S *to the power set of a nonempty subset* AT *of* ATOM*.*

A satisfaction relation $\models$ *on a pk-structure* $M = \langle S, S_0, R, \mu, L \rangle$ *for* pCTL *is defined by the same conditions for* $\models^+$ *(except the condition 6) as in Definition 2.3 (by deleting the superscript* $+$*). The validity, satisfiability and model-checking problems for* pCTL *are defined similarly as those for* PpCTL*.*

3 EMBEDDABILITY AND RELATIVE DECIDABILITY

Definition 3.1. *Let* AT *be a non-empty subset of* ATOM*, and* AT$'$ *be the set* $\{p' \mid p \in \mathrm{AT}\}$ *of atomic formulas. The language* $\mathcal{L}^\sim$ *(the set of formulas) of* PpCTL *is defined using* AT, $\sim, \rightarrow, \wedge, \vee, \neg$, $\mathrm{P}_{\leq x}, \mathrm{P}_{\geq x}, \mathrm{P}_{\prec x}, \mathrm{P}_{\succ x}$, X, F, G, U, R, A *and* E*. The language* $\mathcal{L}$ *of* pCTL *is obtained from* $\mathcal{L}^\sim$ *by adding* AT$'$ *and deleting* $\sim$*.*

A mapping f *from* $\mathcal{L}^\sim$ *to* $\mathcal{L}$ *is defined inductively by:*

1. *for any* $p \in \mathrm{AT}$, $f(p) := p$ *and* $f(\sim p) := p' \in \mathrm{AT}'$,
2. $f(\alpha \sharp \beta) := f(\alpha) \sharp f(\beta)$ *where* $\sharp \in \{\wedge, \vee, \rightarrow\}$,
3. $f(\sharp\alpha) := \sharp f(\alpha)$ *where* $\sharp \in \{\neg, \mathrm{P}_{\leq x}, \mathrm{P}_{\geq x}, \mathrm{P}_{\prec x}, \mathrm{P}_{\succ x}, \mathrm{AX}, \mathrm{EX}, \mathrm{AG}, \mathrm{EG}, \mathrm{AF}, \mathrm{EF}\}$,
4. $f(\mathrm{A}(\alpha \mathrm{U} \beta))) := \mathrm{A}(f(\alpha) \mathrm{U} f(\beta))$,
5. $f(\mathrm{E}(\alpha \mathrm{U} \beta))) := \mathrm{E}(f(\alpha) \mathrm{U} f(\beta))$,
6. $f(\mathrm{A}(\alpha \mathrm{R} \beta))) := \mathrm{A}(f(\alpha) \mathrm{R} f(\beta))$,
7. $f(\mathrm{E}(\alpha \mathrm{R} \beta))) := \mathrm{E}(f(\alpha) \mathrm{R} f(\beta))$,
8. $f(\sim\sim\alpha) := f(\alpha)$,
9. $f(\sim(\alpha \rightarrow \beta)) := f(\alpha) \wedge f(\sim\beta)$,
10. $f(\sim(\alpha \wedge \beta)) := f(\sim\alpha) \vee f(\sim\beta)$,
11. $f(\sim(\alpha \vee \beta)) := f(\sim\alpha) \wedge f(\sim\beta)$,
12. $f(\sim\neg\alpha) := f(\alpha)$,
13. $f(\sim \mathrm{P}_{\leq x}\alpha) := \mathrm{P}_{\succ x} f(\sim\alpha)$,
14. $f(\sim \mathrm{P}_{\geq x}\alpha) := \mathrm{P}_{\prec x} f(\sim\alpha)$,
15. $f(\sim \mathrm{P}_{\prec x}\alpha) := \mathrm{P}_{\geq x} f(\sim\alpha)$,
16. $f(\sim \mathrm{P}_{\succ x}\alpha) := \mathrm{P}_{\leq x} f(\sim\alpha)$,
17. $f(\sim \mathrm{AX}\alpha) := \mathrm{EX} f(\sim\alpha)$,
18. $f(\sim \mathrm{EX}\alpha) := \mathrm{AX} f(\sim\alpha)$,
19. $f(\sim \mathrm{AG}\alpha) := \mathrm{EF} f(\sim\alpha)$,
20. $f(\sim \mathrm{EG}\alpha) := \mathrm{AF} f(\sim\alpha)$,
21. $f(\sim \mathrm{AF}\alpha) := \mathrm{EG} f(\sim\alpha)$,
22. $f(\sim \mathrm{EF}\alpha) := \mathrm{AG} f(\sim\alpha)$,
23. $f(\sim(\mathrm{A}(\alpha \mathrm{U} \beta))) := \mathrm{E}(f(\sim\alpha) \mathrm{R} f(\sim\beta))$,
24. $f(\sim(\mathrm{E}(\alpha \mathrm{U} \beta))) := \mathrm{A}(f(\sim\alpha) \mathrm{R} f(\sim\beta))$,
25. $f(\sim(\mathrm{A}(\alpha \mathrm{R} \beta))) := \mathrm{E}(f(\sim\alpha) \mathrm{U} f(\sim\beta))$,
26. $f(\sim(\mathrm{E}(\alpha \mathrm{R} \beta))) := \mathrm{A}(f(\sim\alpha) \mathrm{U} f(\sim\beta))$.

Lemma 3.2. *Let* f *be the mapping defined in Definition 3.1. For any ppk-structure* $M := \langle S, S_0, R, \mu_s, L^+, L^- \rangle$ *for* PpCTL*, and any satisfaction*

relations $\models^+$ *and* $\models^-$ *on M, we can construct a pk-structure* $N := \langle S, S_0, R, \mu_s, L\rangle$ *for* CTL *and a satisfaction relation* $\models$ *on N such that for any formula* α *in* $\mathcal{L}^\sim$ *and any state s in S,*

1. $M,s \models^+ \alpha$ *iff* $N,s \models f(\alpha)$,
2. $M,s \models^- \alpha$ *iff* $N,s \models f(\sim\alpha)$.

Proof. Let AT be a nonempty subset of ATOM, and AT′ be the set $\{p' \mid p \in \mathrm{AT}\}$ of atomic formulas. Suppose that M is a ppk-structure $\langle S, S_0, R, \mu_s, L^+, L^-\rangle$ such that L^+ and L^- are mappings from S to the power set of AT. Suppose that N is a pk-structure $M := \langle S, S_0, R, \mu_s, L\rangle$ such that L is a mapping from S to the power set of AT$\cup$AT′. Suppose moreover that for any $s \in S$ and any $p \in$ AT,

1. $p \in L^+(s)$ iff $p \in L(s)$,
2. $p \in L^-(s)$ iff $p' \in L(s)$.

The lemma is then proved by (simultaneous) induction on the complexity of α.

• Base step:

Case $\alpha \equiv p \in$ AT: For (1), we obtain: $M,s \models^+ p$ iff $p \in L^+(s)$ iff $p \in L(s)$ iff $N,s \models p$ iff $N,s \models f(p)$ (by the definition of f). For (2), we obtain: $M,s \models^- p$ iff $p \in L^-(s)$ iff $p' \in L(s)$ iff $N,s \models p'$ iff $N,s \models f(\sim p)$ (by the definition of f).

• Induction step: We show some cases.

Case $\alpha \equiv \beta\to\gamma$: For (1), we obtain: $M,s \models^+ \beta\to\gamma$ iff $M,s \models^+ \beta$ implies $M,s \models^+ \gamma$ iff $N,s \models f(\beta)$ implies $N,s \models f(\gamma)$ (by induction hypothesis for 1) iff $N,s \models f(\beta)\to f(\gamma)$ iff $N,s \models f(\beta\to\gamma)$ (by the definition of f). For (2), we obtain: $M,s \models^- \beta\to\gamma$ iff $M,s \models^+ \beta$ and $M,s \models^- \gamma$ iff $N,s \models f(\beta)$ and $N,s \models f(\sim\gamma)$ (by induction hypothesis for 1 and 2) iff $N,s \models f(\beta)\wedge f(\sim\gamma)$ iff $N,s \models f(\sim(\beta\to\gamma))$ (by the definition of f).

Case $\alpha \equiv \neg\beta$: For (1), we obtain: $M,s \models^+ \neg\beta$ iff not-$[M,s \models^+ \beta]$ iff not-$[N,s \models f(\beta)]$ (by induction hypothesis fotr 1) iff $N,s \models \neg f(\beta)$ iff $N,s \models f(\neg\beta)$ (by the definition of f). For (2), we obtain: $M,s \models^- \neg\beta$ iff $M,s \models^+ \beta$ iff $N,s \models f(\beta)$ (by induction hypothesis for 1) iff $N,s \models f(\sim\neg\beta)$ (by the definition of f).

Case $\alpha \equiv \sim\beta$: For (1), we obtain: $M,s \models^+ \sim\beta$ iff $M,s \models^- \beta$ iff $N,s \models f(\sim\beta)$ (by induction hypothesis for 2). For (2), we obtain: $M,s \models^- \sim\beta$ iff $M,s \models^+ \beta$ iff $N,s \models f(\beta)$ (by induction hypothesis for 1) iff $N,s \models f(\sim\sim\beta)$ (by the definition of f).

Case $\alpha \equiv \mathrm{AX}\beta$: For (1), we obtain: $M,s \models^+ \mathrm{AX}\beta$ iff $\forall s_1 \in S\ [(s,s_1) \in R$ implies $M,s_1 \models^+ \beta]$ iff $\forall s_1 \in S\ [(s,s_1) \in R$ implies $N,s_1 \models f(\beta)]$ (by induction hypothesis for 1) iff $N,s \models \mathrm{AX}f(\beta)$ iff $N,s \models f(\mathrm{AX}\beta)$ (by the definition of f). For (2), we obtain: $M,s \models^- \mathrm{AX}\beta$ iff $\exists s_1 \in S\ [(s,s_1) \in R$ and $M,s_1 \models^- \beta]$ iff $\exists s_1 \in S\ [(s,s_1) \in R$ and $N,s_1 \models f(\sim\beta)]$ (by induction hypothesis for 2) iff $N,s \models \mathrm{EX}f(\sim\beta)$ iff $N,s \models f(\sim\mathrm{AX}\beta)$ (by the definition of f).

Case $\alpha \equiv \mathrm{P}_{\leq x}\beta$: For (1), we obtain: $M,s \models^+ \mathrm{P}_{\leq x}\beta$ iff $\mu_s(\{w \in \Omega_s \mid M,w \models^+ \beta\}) \leq x$ iff $\mu_s(\{w \in \Omega_s \mid N,w \models f(\beta)\}) \leq x$ (by induction hypothesis for 1) iff $N,s \models \mathrm{P}_{\leq x}f(\beta)$ iff $N,s \models f(\mathrm{P}_{\leq x}\beta)$ (by the definition of f). For (2), we obtain: $M,s \models^- \mathrm{P}_{\leq x}\beta$ iff $\mu_s(\{w \in \Omega_s \mid M,w \models^- \beta\}) > x$ iff $\mu_s(\{w \in \Omega_s \mid N,w \models f(\sim\beta)\}) > x$ (by induction hypothesis for 2) iff $N,s \models \mathrm{P}_{\succ x}f(\sim\beta)$ iff $N,s \models f(\sim\mathrm{P}_{\leq x}\beta)$ (by the definition of f).

Case $\alpha \equiv \mathrm{P}_{\prec x}\beta$: For (1), we obtain: $M,s \models^+ \mathrm{P}_{\prec x}\beta$ iff $\mu_s(\{w \in \Omega_s \mid M,w \models^+ \beta\}) < x$ iff $\mu_s(\{w \in \Omega_s \mid N,w \models f(\beta)\}) < x$ (by induction hypothesis for 1) iff $N,s \models \mathrm{P}_{\prec x}f(\beta)$ iff $N,s \models f(\mathrm{P}_{\prec x}\beta)$ (by the definition of f). For (2), we obtain: $M,s \models^- \mathrm{P}_{\prec x}\beta$ iff $\mu_s(\{w \in \Omega_s \mid M,w \models^- \beta\}) \geq x$ iff $\mu_s(\{w \in \Omega_s \mid N,w \models f(\sim\beta)\}) \geq x$ (by induction hypothesis for 2) iff $N,s \models \mathrm{P}_{\geq x}f(\sim\beta)$ iff $N,s \models f(\sim\mathrm{P}_{\prec x}\beta)$ (by the definition of f). **Q.E.D.**

Lemma 3.3. *Let* f *be the mapping defined in Definition 3.1. For any pk-structure* $N := \langle S, S_0, R, \mu_s, L\rangle$ *for* pCTL*, and any satisfaction relation* $\models$ *on N, we can construct a ppk-structure* $M := \langle S, S_0, R, \mu_s, L^+, L^-\rangle$ *for* PpCTL *and satisfaction relations* $\models^+$ *and* $\models^-$ *on M such that for any formula* α *in* $\mathcal{L}^\sim$ *and any state s in S,*

1. $N,s \models f(\alpha)$ *iff* $M,s \models^+ \alpha$,
2. $N,s \models f(\sim\alpha)$ *iff* $M,s \models^- \alpha$.

Proof. Similar to the proof of Lemma 3.2. **Q.E.D.**

Theorem 3.4 (Embeddability)**.** *Let* f *be the mapping defined in Definition 3.1. For any formula* α*,*

> α *is valid (satisfiable) in* PpCTL *iff* $f(\alpha)$ *is valid (satisfiable, resp.) in* pCTL.

Proof. By Lemmas 3.2 and 3.3. **Q.E.D.**

Corollary 3.5 (Relative decidability)**.** *If the model-checking, validity and satisfiability problems for* pCTL *with a probability measure are decidable, then the model-checking, validity and satisfiability problems for* PpCTL *with the same probability measure as that of* pCTL *are also decidable.*

Proof. Suppose that the probability measure μ_s in the underlying ppk-structure $\langle S, S_0, R, \mu_s, L^+, L^-\rangle$ of PpCTL is the same as the underlying pk-structure $\langle S, S_0, R, \mu_s, L\rangle$ of pCTL. Suppose also that pCTL with μ_s is decidable. Then, by the mapping f defined in Definition 3.1, a formula α of PpCTL can be transformed into the corresponding formula $f(\alpha)$ of pCTL. By Lemmas 3.2 and 3.3 and Theorem 3.4, the model checking, validity and satisfiability problems for PpCTL can be transformed into those of pCTL. Since the model checking, validity and satisfiability

problems for pCTL with μ_s are decidable by the assumption, the problems for PpCTL with μ_s are also decidable. **Q.E.D.**

Some remarks on the decidability are given:

1. The logic pCTL with two probability measures μ_s^+ and μ_s^- by Bianco and de Alfaro is decidable (Bianco and de Alfaro, 1995). The logic pCTL with a probability measure μ^s by Aziz et al. is also decidable (Aziz et al., 1995). Thus, the extended PpCTLs based on the above pCTLs are also decidable by Corollary 3.5.
2. Since the mapping f from PpCTL into pCTL is a polynomial-time reduction, the complexity results for PpCTL becomes the same results as those for pCTL., e.g., if the model-checking problem for pCTL is deterministic PTIME-complete, then so is PpCTL.
3. The model-checking, validity and satisfiability problems for both CTL and its paraconsistent extension PCTL (Kaneiwa and Kamide, 2011) are known to be EXPTIME-complete, deterministic EXPTIME-complete and deterministic PTIME-complete, respectively.

4 ILLUSTRATIVE EXAMPLES

4.1 SQL Injection Attack Detection

SQL injection (Clarke, 2009) is one of the numerous malicious attack methods used to exploit security vulnerabilities on SQL database servers. An attacker sends injection codes through a network to illegally obtain stored information from the SQL database servers. An automatic detection method for SQL injection attacks is explained here on the basis of the studies reported by (Sonoda et al., 2011; Matsuda et al., 2011; Koizumi et al., 2012). They utilized the contained rate of suspicious characters over the length of an input string. Consider that an automatic detection program attempts to determine if the ith input string l_i $(i = 1,2,\ldots)$ to an SQL database server is obtained as a result of an SQL injection attack. Then, the *contained rate* p_i can be defined as:

$$p_i = \frac{\#S}{|l_i|}, \tag{1}$$

where $\#S$ is the number of suspicious characters and $|l_i|$ is the length of the ith input string. Automatic detection with p_i is executed on the basis of the following rule:

$$h(p_i) = \begin{cases} 1 & \text{if} \quad p_i > \alpha; \\ 0 & \text{otherwise}, \end{cases} \tag{2}$$

where $h(p_i) = 1$ indicates that the detected result is an attack string, $h(p_i) = 0$ implies that it is a normal string, and α is a predetermined *threshold value*. A set S contains some suspicious characters (e.g., a space, semi-colon, single quotation, etc.) in the input string of some SQL injection attacks.

Example 4.1. *Suppose the unknown input string l_1 as* "`DROP sampletable;--`" *to the SQL server. Let the elements of S be a space, semi-colon, and right parenthesis, and let the threshold value α be 0.08. Then, this input l_1 is detected as attack string because the length $|l_1|$ is 19 and the suspicious characters contained in S is 2, the contained rate $p_1 = 2/19 = 0.105$, and hence p_1 is greater than α.*

In the experiment by (Sonoda et al., 2011), each *attack detection rate* μ_A and *normal detection rate* μ_N for the underlying characters was calculated by changing the threshold α. An *overall detection rate* μ is defined as the weighted average of μ_A and μ_N:

$$\mu = (1-\beta) \times \mu_A + \beta \times \mu_N, \tag{3}$$

where a real number β, which satisfies $0 \leq \beta \leq 1$, is the weight of the normal string over the input strings. The use of the SQL injection attack detection algorithm explained above is assumed in the following discussion.

4.2 Representing Paraconsistency

Now, we consider some example formulas for SQL injection attacks. The paraconsistent negation connective $\sim\alpha$ in PpCTL is used to represent the negation of an uncertain or ambiguous concept "attack". If we cannot determine whether an input string is obtained by an SQL injection attack, then this concept is regarded as uncertain. The uncertain concept *attack* can be represented by asserting the inconsistent formula of the form: $attack \wedge \sim attack$ where $\sim attack$ represents the uncertain negation information that can be true at the same time as *attack*, which represents positive information. This is well-formalized because the formula of the form: $(attack \wedge \sim attack) \rightarrow \bot$ is not valid in PpCTL.

We can also present the following formula: $\mathrm{EF}(attack \wedge \sim attack)$ which implies: "There exists a situation in which a string input is considered to be obtained as both an SQL injection attack and a non-SQL injection attack, i.e., we cannot determine by the algorithm whether a string was obtained from

an attack." In addition, we can present the following formula: EF(*crashed* ∧ AG *crashed*) which implies: "There is a situation in which a crashed database caused by an SQL injection attack will not function again."

4.3 Representing Probability

We can express Example 4.1 as the following formula: $\mathrm{AG}(\mathrm{P}_{\leq 0.08}\alpha \wedge (p_i < \alpha) \rightarrow \sim\mathrm{attack})$ which implies: "If the threshold value α is at the most 8 percent and the contained rate p_i is greater than α, then the string was *probably not* obtained by an SQL injection attack, i.e., it can be regarded as a normal string."

Let μ_A and μ_N be an attack detection rate and a normal detection rate, respectively. Then, we can present the following formula: $\mathrm{AG}(\mathrm{P}_{\geq 0.08}\mu_A \wedge \mathrm{P}_{\geq 0.02}\mu_N \rightarrow \mathrm{attack})$ which implies: "If the attack detection rate μ_A and the normal detection rate μ_N with respect to some fixed characters in the underlying string are at least 8 percent and at least 2 percent, respectively, then the string is obtained by an SQL injection attack, i.e., it is regarded as a malicious attack string."

Similarly, we can present the following formula: $\mathrm{AG}\ (\mathrm{P}_{\prec 0.08}\mu_A \wedge \mathrm{P}_{\prec 0.02}\mu_N \rightarrow \sim\mathrm{attack})$ which implies: "The string entered by someone is *probably not* obtained by an SQL injection attack." In addition, we present the following formula with the classical negation connective $\neg$: $\mathrm{AG}\ (\mathrm{P}_{\prec 0.02}\mu_A \wedge \mathrm{P}_{\prec 0.01}\mu_N \rightarrow \neg\mathrm{attack})$ which implies: "The string entered by someone is *clearly not* obtained by an SQL injection attack, i.e., it is just a normal string."

4.4 Representing Experimental Facts

The single quotation mark "`'`" forms a set with the previous single quotation. A pair of single quotation marks appears, for instance, as "`uid='user01'`" which implies: "the user ID is user01." We can present this situation as the following formula: AG(singleQuotation ∧ EF singleQuotation → ∼attack) which implies: "At any time, if a single quotation "`'`" appears in the string described in a web form, and the corresponding (closed) single quotation "`'`" eventually appears in the same string, then such an input string is probably not obtained as an SQL injection attack."

The statement "`OR 1=1`" is sometimes used in an attack string. Then, we present this situation as the following formula: AG(EF or1=1 → attack) which implies: "At any time, if the statement "`OR 1=1`" eventually appears, then such an input string was probably obtained as an SQL injection attack."

5 CONCLUSIONS AND RELATED WORKS

In this paper, the paraconsistent probabilistic computation tree logic (PpCTL) was introduced and studied. PpCTL was constructed by combining two existing extended temporal logics: paraconsistent computation tree logic (PCTL) and probabilistic computation tree logic (pCTL). Then, a theorem for embedding PpCTL into pCTL was proven using translation, which is independent of the probability measure setting. A relative decidability theorem for PpCTL, which states that the decidability of pCTL implies that of PpCTL, was also obtained as a corollary of this embedding theorem. This relative decidability theorem indicates that we can reuse some existing pCTL-based verification algorithms. Some illustrative examples for describing an SQL injection attack detection algorithm, involving the use of PpCTL, were also presented to highlight the virtues of combining paraconsistency (in PCTL) and probability (in pCTL).

Some remarks are given as follows. A translation from PpCTL into PCTL was not given in this paper, although a translation from PpCTL into pCTL was given. The issue for obtaining a translation from PpCTL (pCTL) into PCTL (CTL, resp) has not been solved yet, because a formula with the probabilistic operators which have the probability measures is difficult to translate into a non-probabilistic formula of PCTL or CTL. In the meantime, we would like to extend the proposed embedding-based method for an extended PpCTL with the *sequence modal operator* which was introduced for expressing ontological or hierarchical information (see e.g, (Kamide, 2013)). This issue is remained as a future work.

The rest of this paper addresses some closely related works. While the idea of combining paraconsistency and probability within a temporal logic is new, the idea of introducing a paraconsistent computation tree logic is not. In this study, PCTL (Kamide and Kaneiwa, 2010; Kaneiwa and Kamide, 2011) was used as a base logic for constructing PpCTL. However, there are some other paraconsistent variants of CTL. For example, a *multi-valued computation tree logic*, χCTL, was introduced by Easterbrook and Chechik (Easterbrook and Chechik, 2001), and a *quasi-classical temporal logic*, QCTL, was proposed by Chen and Wu (Chen and Wu, 2006). PCTL was introduced as an alternative to these logics. In addition, an extension PCTL* of PCTL has also been studied from the viewpoint of bisimulations for paraconsistent Kripke structures in paraconsistent model checking (Kamide, 2006). Another extension of PCTL was also studied in (Kamide, 2013) for verifying student

learning processes in learning support systems.

Compared with paraconsistent CTLs, several studies have been reported on probabilistic temporal logics, including probabilistic CTLs. The study in (Hansson and Jonsson, 1994) is a typical example of such a study. In (Hansson and Jonsson, 1994), a probabilistic and real-time extension of CTL, also called PCTL, was introduced and investigated on the basis of an interpretation of *discrete time Markov chains*. In contrast to the probabilistic frameworks of pCTL and PpCTL, the notion of probability in PCTL is assigned to all the temporal operators in PCTL. For example, a PCTL formula with the form $G^{\leq t}_{\geq p}\alpha$ implies "the formula α holds continuously for t time units with a probability of at least p."

REFERENCES

Almukdad, A. and Nelson, D. (1984). Constructible falsity and inexact predicates. *Journal of Symbolic Logic*, 49:231–233.

Aziz, A., Singhal, V., and Balarin, F. (1995). It usually works: The temporal logic of stochastic systems. In *Proceedings of the 7th Int. Conf. on Computer Aided Verification (CAV 1995), Lecture Notes in Computer Science 939*, pages 155–165.

Bianco, A. and de Alfaro, L. (1995). Model checking of probabilistic and nondeterministic systems. In *Proceedings of the 15th Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 1995), Lecture Notes in Computer Science 1026*, pages 499–513.

Carnielli, W., Coniglio, M., Gabbay, D., Gouveia, P., and Sernadas, C. (2008). *Analysis and synthesis of logics: How to cut and paste reasoning systems, Applied Logic Series, Vol. 35*. Springer.

Chen, D. and Wu, J. (2006). Reasoning about inconsistent concurrent systems: A non-classical temporal logic. In *Lecture Notes in Computer Science*, volume 3831, pages 207–217.

Clarke, E. and Emerson, E. (1981). Design and synthesis of synchronization skeletons using branching time temporal logic. In *Lecture Notes in Computer Science*, volume 131, pages 52–71.

Clarke, E., Grumberg, O., and Peled, D. (1999). *Model checking*. The MIT Press.

Clarke, J. (2009). *SQL injection attacks and defense, 2nd Edition*. Syngress Publishing.

Easterbrook, S. and Chechik, M. (2001). A framework for multi-valued reasoning over inconsistent viewpoints. In *Proceedings of the 23rd International Conference on Software Engineering*, pages 411–420.

Gurevich, Y. (1977). Intuitionistic logic with strong negation. *Studia Logica*, 36:49–59.

Hansson, H. and Jonsson, B. (1994). A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6 (5):512–535.

Kamide, N. (2006). Extended full computation tree logics for paraconsistent model checking. *Logic and Logical Philosophy*, 15 (3):251–276.

Kamide, N. (2013). Modeling and verifying inconsistency-tolerant temporal reasoning with hierarchical information: Dealing with students' learning processes. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC 2013)*, pages 1859–1864.

Kamide, N. and Kaneiwa, K. (2010). Paraconsistent negation and classical negation in computation tree logic. In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence (ICAART 2010), Vol.1*, pages 464–469.

Kamide, N. and Wansing, H. (2012). Proof theory of nelson's paraconsistent logic: A uniform perspective. *Theoretical Computer Science*, 415:1–38.

Kaneiwa, K. and Kamide, N. (2011). Paraconsistent computation tree logic. *New Generation Computing*, 29 (4):391–408.

Koizumi, D., Matsuda, T., Sonoda, M., and Hirasawa, S. (2012). A learning algorithm of threshold value on the automatic detection of SQL injection attack. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2012), Vol.II*, pages 933–937.

Matsuda, T., Koizumi, D., Sonoda, M., and Hirasawa, S. (2011). On predictive errors of SQL injection attack detection by the feature of the single character. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, (SMC 2011)*, pages 1722–1727.

Nelson, D. (1949). Constructible falsity. *Journal of Symbolic Logic*, 14:16–26.

Pnueli, A. (1977). The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, pages 46–57.

Priest, G. and Routley, R. (1982). Introduction: paraconsistent logics. *Studia Logica*, 43:3–16.

Rautenberg, W. (1979). *Klassische und nicht-klassische Aussagenlogik*. Vieweg, Braunschweig.

Sonoda, M., Matsuda, T., Koizumi, D., and Hirasawa, S. (2011). On automatic detection of SQL injection attacks by the feature extraction of the single character. In *Proceedings of the 4th International Conference on Security of Information and Networks (SIN 2011)*, pages 81–86.

Vorob'ev, N. (1952). A constructive propositional calculus with strong negation (in Russian). *Doklady Akademii Nauk SSR*, 85:465–468.

Wansing, H. (1993). The logic of information structures. In *Lecture Notes in Computer Science*, volume 681, pages 1–163.