Paraconsistent Computation Tree Logic[*1]

Ken KANEIWA

Department of Electrical Engineering and Computer Science,
Iwate University
4-3-5 Ueda, Morioka, Iwate 020-8551, JAPAN

kaneiwa@cis.iwate-u.ac.jp

Norihiro KAMIDE

Waseda Institute for Advanced Study, Waseda University
1-6-1 Nishi Waseda, Shinjuku-ku, Tokyo 169-8050, JAPAN

drnkamide08@kpd.biglobe.ne.jp

Abstract It is known that paraconsistent logical systems are more appropriate for inconsistency-tolerant and uncertainty reasoning than other types of logical systems. In this paper, a paraconsistent computation tree logic, PCTL, is obtained by adding paraconsistent negation to the standard computation tree logic CTL. PCTL can be used to appropriately formalize inconsistency-tolerant temporal reasoning. A theorem for embedding PCTL into CTL is proved. The validity, satisfiability, and model-checking problems of PCTL are shown to be decidable. The embedding and decidability results indicate that we can reuse the existing CTL-based algorithms for validity, satisfiability, and model-checking. An illustrative example of medical reasoning involving the use of PCTL is presented.

Keywords: Paraconsistent Logic, Computation Tree Logic, Decidability, Medical Reasoning.

[*1] This paper is a modified extension of the conference presentation.[16)]

§1 Introduction

Logical systems have been investigated as useful tools in the areas of artificial intelligence and computer science.[13, 19, 18, 20] *Computation tree logic* (CTL)[5] is known to be one of the most useful temporal logics for verifying concurrent systems by *model checking*,[7] since some CTL-based model checking algorithms are more efficient than other types of algorithms. However, the use of CTL is not suitable for verifying "inconsistent" concurrent systems since CTL is based on classical logic. Handling inconsistencies in concurrent systems requires the use of a *paraconsistent logic*[2, 24] as a base logic for CTL.

One of the most useful paraconsistent logics is *Nelson's four-valued paraconsistent logic* N4 (or also called N^{-}),[1, 22] which includes a paraconsistent negation connective. The logic N4 and its variants have been studied by many researchers.[28, 29] N4 has been extensively studied since it has the property of *paraconsistency*.[2, 8, 24] Roughly, 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 a state of a 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.

It is known that paraconsistent logical systems are more appropriate for inconsistency-tolerant and uncertainty reasoning than other types of logical systems.[2, 8, 24, 28, 29] For example, it is undesirable that $(s(x) \wedge \sim s(x)) \rightarrow d(x)$ is 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." An inconsistent scenario expressed, for example, 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 undesirable consequences), while it is not valid in paraconsistent logics (i.e., these logics are inconsistency-tolerant).

Inconsistencies often appear and are inevitable when specifying large, complex systems in some CTL-based frameworks. N4 is then useful and appropriate as a base logic for CTL. Moreover, N4 has notable two satisfaction relations $\models^{+}$ (verification) and $\models^{-}$ (refutation) in the Kripke semantics. By using these satisfaction relations, the ideas of "verification (or justification)"

and "refutation (or falsification)" can be simultaneously incorporated into the system. Therefore, the combination of CTL and N4 is regarded as a natural candidate for obtaining a useful paraconsistent temporal logic.

In this paper, a new paraconsistent computation tree logic called PCTL is introduced by combining CTL and N4. While the idea of combining CTL and N4 is new, the idea of introducing a paraconsistent computation tree logic is not. For example, a *multi-valued computation tree logic* χCTL was introduced by Easterbrook and Chechik,[9] and a *quasi-classical temporal logic* QCTL was developed by Chen and Wu.[4] Thus, PCTL is introduced as an alternative to these logics, and N4 replaces the base paraconsistent logic.

As mentioned above, the application for which paraconsistent logics show the greatest promise may be medical informatics. Indeed, it has been pointed out that paraconsistent logics are useful for medical reasoning.[8, 21] Some paraconsistent computation tree logics, including PCTL, may be more useful in medical informatics because the notion of time is necessary in order to appropriately formalize realistic medical reasoning. Against this background, we present an illustrative example of medical reasoning. The proposed illustrative example can also be adapted to other paraconsistent computation tree logics such as χCTL and QCTL.

The results of this paper are summarized as follows. In Section 2, PCTL is introduced as a combination of CTL and N4. In Section 3, a theorem for embedding PCTL into CTL is proved. The validity, satisfiability, and model-checking problems of PCTL are shown to be decidable. The embedding and decidability results show that we can reuse the existing CTL-based algorithms in validity, satisfiability, and model-checking problems.[7] This is an advantage of PCTL. It is thus shown that PCTL is useful as an executable logic for temporal reasoning with paraconsistency. In Section 4, an illustrative example of medical reasoning involving the use of PCTL is presented. In this example, an ambiguous concept "healthy" is suitably handled by using paraconsistent negation. In Section 5, a comparison between PCTL and other proposed logics such as χCTL and QCTL is given, and Section 6 concludes this paper.

§2 Paraconsistent computation tree logic

Formulas of PCTL are constructed from countably many atomic formulas, $\rightarrow$ (implication) $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (classical negation), $\sim$ (paraconsistent negation), 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 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.

Definition 2.1
Formulas α are defined by the following grammar, assuming $p \in \mathrm{ATOM}$:

$$\alpha ::= p \mid \alpha{\rightarrow}\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \neg\alpha \mid {\sim}\alpha \mid \mathrm{AX}\alpha \mid \mathrm{EX}\alpha \mid \mathrm{AG}\alpha \mid \mathrm{EG}\alpha \mid \mathrm{AF}\alpha \mid \mathrm{EF}\alpha \mid \mathrm{A}(\alpha\mathrm{U}\alpha) \mid \mathrm{E}(\alpha\mathrm{U}\alpha) \mid \mathrm{A}(\alpha\mathrm{R}\alpha) \mid \mathrm{E}(\alpha\mathrm{R}\alpha).$$

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. Remark that all the connectives displayed above are needed to obtain an embedding theorem of PCTL into CTL.

Definition 2.2
A *paraconsistent Kripke structure* is a structure $\langle S, S_0, R, L^+, L^- \rangle$ 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. L^+ and L^- are functions from S to the power set of a nonempty subset AT of ATOM.

A *path* in a paraconsistent Kripke structure is an infinite sequence of states, $\pi = s_0, s_1, s_2, ...$ such that $\forall i \geq 0\ [(s_i, s_{i+1}) \in R]$.

The logic PCTL is then defined as a paraconsistent Kripke structure with two satisfaction relations $\models^+$ and $\models^-$. The intuitive meanings of $\models^+$ and $\models^-$ are "verification (or justification)" and "refutation (or falsification)," respectively.[29)]

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

1. for any $p \in \mathrm{AT}$, $M, s \models^+ p$ 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. $M, s \models^+ \mathrm{AX}\alpha$ iff $\forall s_1 \in S$ $[(s, s_1) \in R$ implies $M, s_1 \models^+ \alpha]$,
8. $M, s \models^+ \mathrm{EX}\alpha$ iff $\exists s_1 \in S$ $[(s, s_1) \in R$ and $M, s_1 \models^+ \alpha]$,
9. $M, s \models^+ \mathrm{AG}\alpha$ 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 $M, s_i \models^+ \alpha$,
10. $M, s \models^+ \mathrm{EG}\alpha$ 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 $M, s_i \models^+ \alpha$,
11. $M, s \models^+ \mathrm{AF}\alpha$ 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 $M, s_i \models^+ \alpha$,
12. $M, s \models^+ \mathrm{EF}\alpha$ 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 $M, s_i \models^+ \alpha$,
13. $M, s \models^+ \mathrm{A}(\alpha_1 \mathrm{U} \alpha_2)$ 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 $M, s_j \models^+ \alpha_1)]$,
14. $M, s \models^+ \mathrm{E}(\alpha_1 \mathrm{U} \alpha_2)$ 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 $M, s_j \models^+ \alpha_1)]$,
15. $M, s \models^+ \mathrm{A}(\alpha_1 \mathrm{R} \alpha_2)$ 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 $M, s_j \models^+ \alpha_2]$,
16. $M, s \models^+ \mathrm{E}(\alpha_1 \mathrm{R} \alpha_2)$ 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 $M, s_j \models^+ \alpha_2]$,
17. for any $p \in \mathrm{AT}$, $M, s \models^- p$ iff $p \in L^-(s)$,
18. $M, s \models^- \alpha_1 \rightarrow \alpha_2$ iff $M, s \models^+ \alpha_1$ and $M, s \models^- \alpha_2$,
19. $M, s \models^- \alpha_1 \wedge \alpha_2$ iff $M, s \models^- \alpha_1$ or $M, s \models^- \alpha_2$,
20. $M, s \models^- \alpha_1 \vee \alpha_2$ iff $M, s \models^- \alpha_1$ and $M, s \models^- \alpha_2$,
21. $M, s \models^- \neg\alpha_1$ iff $M, s \models^+ \alpha_1$,
22. $M, s \models^- \sim\alpha_1$ iff $M, s \models^+ \alpha_1$,
23. $M, s \models^- \mathrm{AX}\alpha$ iff $\exists s_1 \in S$ $[(s, s_1) \in R$ and $M, s_1 \models^- \alpha]$,
24. $M, s \models^- \mathrm{EX}\alpha$ iff $\forall s_1 \in S$ $[(s, s_1) \in R$ implies $M, s_1 \models^- \alpha]$,

25. $M, s \models^- \mathrm{AG}\alpha$ 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 $M, s_i \models^- \alpha$,
26. $M, s \models^- \mathrm{EG}\alpha$ 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 $M, s_i \models^- \alpha$,
27. $M, s \models^- \mathrm{AF}\alpha$ 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 $M, s_i \models^- \alpha$,
28. $M, s \models^- \mathrm{EF}\alpha$ 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 $M, s_i \models^- \alpha$,
29. $M, s \models^- \mathrm{A}(\alpha_1 \mathrm{U} \alpha_2)$ 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 $M, s_j \models^- \alpha_2]$,
30. $M, s \models^- \mathrm{E}(\alpha_1 \mathrm{U} \alpha_2)$ 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 $M, s_j \models^- \alpha_2]$,
31. $M, s \models^- \mathrm{A}(\alpha_1 \mathrm{R} \alpha_2)$ 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 $M, s_j \models^- \alpha_1)]$,
32. $M, s \models^- \mathrm{E}(\alpha_1 \mathrm{R} \alpha_2)$ 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 $M, s_j \models^- \alpha_1)]$.

Remark that the conditions 5 and 21 in Definition 2.3 correspond to the axiom scheme $\sim\neg\alpha \leftrightarrow \alpha$ which is used as an axiom for some constructive logics with strong negation. We can adopt the following condition instead of the condition 21 in Definition 2.3:

$21'$. $M, s \models^- \neg\alpha_1$ iff not-$[M, s \models^- \alpha_1]$

which is symmetric to the condition 5 in Definition 2.3. The conditions 5 and $21'$ correspond to the axiom scheme $\sim\neg\alpha \leftrightarrow \neg\sim\alpha$. The logic which is obtained from PCTL by replacing 21 by $21'$ is called here PCTL$'$. The (corresponding) embedding and decidability theorems for PCTL$'$ can also be obtained by using the same way as that for PCTL.

Definition 2.4

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

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

Note that the positive model checking problem for PCTL corresponds to the standard "verification-based" model checking problem for CTL, and that the negative model checking problem for PCTL corresponds to the dual of positive one, i.e., it is regarded as a "refutation-based" model checking problem. Remark that both the positive and negative model checking should simultaneously be performed, i.e., only one of them cannot be performed.

An expression $\alpha \leftrightarrow \beta$ is used to represent $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$.

Proposition 2.6
The following formulas concerning paraconsistent negation are valid in PCTL: for any formulas α and β,

1. $\sim\sim\alpha \leftrightarrow \alpha$,
2. $\sim(\alpha \wedge \beta) \leftrightarrow \sim\alpha \vee \sim\beta$,
3. $\sim(\alpha \vee \beta) \leftrightarrow \sim\alpha \wedge \sim\beta$,
4. $\sim(\alpha \rightarrow \beta) \leftrightarrow \alpha \wedge \sim\beta$,
5. $\sim\neg\alpha \leftrightarrow \alpha$,
6. $\sim\mathrm{AX}\alpha \leftrightarrow \mathrm{EX}\sim\alpha$,
7. $\sim\mathrm{EX}\alpha \leftrightarrow \mathrm{AX}\sim\alpha$,
8. $\sim\mathrm{AG}\alpha \leftrightarrow \mathrm{EF}\sim\alpha$,
9. $\sim\mathrm{EG}\alpha \leftrightarrow \mathrm{AF}\sim\alpha$,
10. $\sim\mathrm{AF}\alpha \leftrightarrow \mathrm{EG}\sim\alpha$,
11. $\sim\mathrm{EF}\alpha \leftrightarrow \mathrm{AG}\sim\alpha$,
12. $\sim\mathrm{A}(\alpha\mathrm{U}\beta) \leftrightarrow \mathrm{E}((\sim\alpha)\mathrm{R}(\sim\beta))$,
13. $\sim\mathrm{E}(\alpha\mathrm{U}\beta) \leftrightarrow \mathrm{A}((\sim\alpha)\mathrm{R}(\sim\beta))$,
14. $\sim\mathrm{A}(\alpha\mathrm{R}\beta) \leftrightarrow \mathrm{E}((\sim\alpha)\mathrm{U}(\sim\beta))$,
15. $\sim\mathrm{E}(\alpha\mathrm{R}\beta) \leftrightarrow \mathrm{A}((\sim\alpha)\mathrm{U}(\sim\beta))$.

Proof. We show some cases. Suppose that $M = \langle S, S_0, R, L^+, L^- \rangle$ is an arbitrary paraconsistent Kripke structure, and that $\models^+$ and $\models^-$ are any satisfaction relations on M.

(5): We show only that $\sim\neg\alpha \rightarrow \alpha$ is valid in PCTL. Let s be an arbitrary

element of S. Then, to show $M, s \models^+ {\sim}\neg\alpha{\rightarrow}\alpha$, we have to show that $M, s \models^+ {\sim}\neg\alpha$ implies $M, s \models^+ \alpha$. Suppose $M, \models^+ {\sim}\neg\alpha$. Then, we obtain the required fact as follows: $M, s \models^+ {\sim}\neg\alpha$ iff $M, s \models^- \neg\alpha$ iff $M, s \models^+ \alpha$.

(12): We show only that ${\sim}\mathrm{A}(\alpha\mathrm{U}\beta){\rightarrow}\mathrm{E}(({\sim}\alpha)\mathrm{R}({\sim}\beta))$ is valid in PCTL. Let s be an arbitrary element of S. Then, we show that $M, s \models^+ {\sim}\mathrm{A}(\alpha\mathrm{U}\beta){\rightarrow}$ $\mathrm{E}(({\sim}\alpha)\mathrm{R}({\sim}\beta))$. To show this, we need to show that $M, s \models^+ {\sim}\mathrm{A}(\alpha\mathrm{U}\beta)$ implies $M, s \models^+ \mathrm{E}(({\sim}\alpha)\mathrm{R}({\sim}\beta))$. Suppose $M, s \models^+ {\sim}\mathrm{A}(\alpha\mathrm{U}\beta)$, i.e., $M, s \models^- \mathrm{A}(\alpha\mathrm{U}\beta)$. Then, we obtain the required fact as follows:

$M, s \models^- \mathrm{A}(\alpha\mathrm{U}\beta)$
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-$[M, s_i \models^- \alpha]$ implies $M, s_j \models^- \beta]$
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-$[N, s_i \models^+ {\sim}\alpha]$ implies $N, s_j \models^+ {\sim}\beta]$
iff $M, s \models^+ \mathrm{E}(({\sim}\alpha)\mathrm{R}({\sim}\beta))$.

∎

In the following, we explain that (1) PCTL is regarded as a four-valued logic, and (2) PCTL is a paraconsistent logic (see more information on paraconsistent logics[2, 24]).

For each $s \in S$ and each formula α, we can take one of the following four cases: (1) α is verified at s, i.e., $M, s \models^+ \alpha$, (2) α is falsified at s, i.e., $M, s \models^- \alpha$, (3) α is both verified and falsified at s, and (4) α is neither verified nor falsified at s. Thus, PCTL is regarded as a four-valued logic.

Assume a paraconsistent Kripke structure $M = \langle S, S_0, R, L^+, L^-\rangle$ 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 PCTL is paraconsistent with respect to $\sim$.

In order to define a translation of PCTL into CTL, a Kripke structure for CTL is defined below.

Definition 2.7 (CTL)
A *Kripke structure* for CTL is a structure $\langle S, S_0, R, L\rangle$ 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. L is a function from S to the power set of a nonempty subset AT of ATOM.

A *satisfaction relation* $\models$ on a Kripke structure $M = \langle S, S_0, R, L\rangle$ for CTL is defined by the same conditions 1–5 and 7–16 as in Definition 2.3 (by deleting the superscript +). The validity, satisfiability, and model-checking problems for CTL are defined similarly as those for PCTL.

Remark that $\models^+$ of PCTL includes $\models$ of CTL, and hence PCTL is an extension of CTL.

§3 Embedding and decidability

In the following, we introduce a translation of PCTL into CTL, and by using this translation, we show an embedding theorem of PCTL into CTL. A similar translation has been used by Gurevich,[12] Rautenberg,[26] and Vorob'ev[27] to embed Nelson's three-valued constructive logic[1, 22] into intuitionistic logic.

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 PCTL is defined using AT, $\sim$, $\neg, \rightarrow, \wedge, \vee$, X, F, G, U, R, A, and E. The language $\mathcal{L}$ of CTL 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 $\sharp \in \{\neg, \mathrm{AX}, \mathrm{EX}, \mathrm{AG}, \mathrm{EG}, \mathrm{AF}, \mathrm{EF}\}$,
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. $f(\sim\mathrm{AX}\alpha) := \mathrm{EX}f(\sim\alpha)$,

14. $f(\sim \mathrm{EX}\alpha) := \mathrm{AX}f(\sim\alpha)$,
15. $f(\sim \mathrm{AG}\alpha) := \mathrm{EF}f(\sim\alpha)$,
16. $f(\sim \mathrm{EG}\alpha) := \mathrm{AF}f(\sim\alpha)$,
17. $f(\sim \mathrm{AF}\alpha) := \mathrm{EG}f(\sim\alpha)$,
18. $f(\sim \mathrm{EF}\alpha) := \mathrm{AG}f(\sim\alpha)$,
19. $f(\sim(\mathrm{A}(\alpha \mathrm{U} \beta))) := \mathrm{E}(f(\sim\alpha)\mathrm{R}f(\sim\beta))$,
20. $f(\sim(\mathrm{E}(\alpha \mathrm{U} \beta))) := \mathrm{A}(f(\sim\alpha)\mathrm{R}f(\sim\beta))$,
21. $f(\sim(\mathrm{A}(\alpha \mathrm{R} \beta))) := \mathrm{E}(f(\sim\alpha)\mathrm{U}f(\sim\beta))$,
22. $f(\sim(\mathrm{E}(\alpha \mathrm{R} \beta))) := \mathrm{A}(f(\sim\alpha)\mathrm{U}f(\sim\beta))$.

Lemma 3.2

Let f be the mapping defined in Definition 3.1. For any paraconsistent Kripke structure $M := \langle S, S_0, R, L^+, L^- \rangle$ for PCTL, and any satisfaction relations $\models^+$ and $\models^-$ on M, we can construct a Kripke structure $N := \langle S, S_0, R, L \rangle$ 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 paraconsistent Kripke structure $\langle S, S_0, R, L^+, L^- \rangle$ such that

L^+ and L^- are functions from S to the power set of AT.

Suppose that N is a Kripke structure $M := \langle S, S_0, R, L \rangle$ such that

L is a function from S to the power set of $\mathrm{AT} \cup \mathrm{AT}'$.

Suppose moreover that for any $s \in S$ and any $p \in \mathrm{AT}$,

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 $\alpha \equiv p \in \mathrm{AT}$: 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 $\alpha \equiv \beta \wedge \gamma$: For (1), we obtain: $M, s \models^+ \beta \wedge \gamma$ iff $M, s \models^+ \beta$ and $M, s \models^+ \gamma$ iff $N, s \models f(\beta)$ and $N, s \models f(\gamma)$ (by induction hypothesis for 1) iff $N, s \models f(\beta) \wedge f(\gamma)$ iff $N, s \models f(\beta \wedge \gamma)$ (by the definition of f). For (2), we obtain: $M, s \models^- \beta \wedge \gamma$ iff $M, s \models^- \beta$ or $M, s \models^- \gamma$ iff $N, s \models f(\sim\beta)$ or $N, s \models f(\sim\gamma)$ (by induction hypothesis for 2) iff $N, s \models f(\sim\beta) \vee f(\sim\gamma)$ iff $N, s \models f(\sim(\beta \wedge \gamma))$ (by the definition of f).

Case $\alpha \equiv \beta \rightarrow \gamma$: For (1), we obtain: $M, s \models^+ \beta \rightarrow \gamma$ 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 $N, s \models f(\beta) \rightarrow f(\gamma)$ iff $N, s \models f(\beta \rightarrow \gamma)$ (by the definition of f). For (2), we obtain: $M, s \models^- \beta \rightarrow \gamma$ 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 $N, s \models f(\sim(\beta \rightarrow \gamma))$ (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 $\alpha \equiv \mathrm{AG}\beta$: For (1), we obtain:

$M, s \models^+ \mathrm{AG}\beta$
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 $M, s_i \models^+ \beta$
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 $N, s_i \models f(\beta)$ (by induction hypothesis for 1)
iff $N, s \models \mathrm{AG}f(\beta)$
iff $N, s \models f(\mathrm{AG}\beta)$ (by the definition of f).

For (2), we obtain:

$M, s \models^{-} \mathrm{AG}\beta$
iff there is a path $\pi \equiv s_0, s_1, s_2, ...$, where $s \equiv s_0$, for some state s_i along π, we have $M, s_i \models^{-} \beta$
iff there is a path $\pi \equiv s_0, s_1, s_2, ...$, where $s \equiv s_0$, for some state s_i along π, we have $N, s_i \models f(\sim\beta)$ (by induction hypothesis for 2)
iff $N, s \models \mathrm{EF}f(\sim\beta)$
iff $N, s \models f(\sim\mathrm{AG}\beta))$ (by the definition of f).

Case $\alpha \equiv \mathrm{A}(\beta\mathrm{U}\gamma)$: For (1), we obtain:

$M, s \models^{+} \mathrm{A}(\beta\mathrm{U}\gamma)$
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 $[M, s_k \models^{+} \gamma$ and $\forall j[i \leq j < k$ implies $M, s_j \models^{+} \beta]$
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 $[N, s_k \models f(\gamma)$ and $\forall j[i \leq j < k$ implies $N, s_j \models f(\beta)]$ (by induction hypothesis for 1)
iff $N, s \models \mathrm{A}(f(\beta)\mathrm{U}f(\gamma))$
iff $N, s \models f(\mathrm{A}(\beta\mathrm{U}\gamma))$ (by the definition of f).

For (2), we obtain:

$M, s \models^{-} \mathrm{A}(\beta\mathrm{U}\gamma)$
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-$[M, s_i \models^{-} \beta]$ implies $M, s_j \models^{-} \gamma]$
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-$[N, s_i \models f(\sim\beta)]$ implies $N, s_j \models f(\sim\gamma)]$ (by induction hypothesis for 2)
iff $N, s \models \mathrm{E}(f(\sim\beta)\mathrm{R}f(\sim\gamma))$
iff $N, s \models f(\sim(\mathrm{A}(\beta\mathrm{U}\gamma)))$ (by the definition of f).

∎

Lemma 3.3
Let f be the mapping defined in Definition 3.1. For any Kripke structure $N := \langle S, S_0, R, L\rangle$ for CTL, and any satisfaction relation $\models$ on N, we can construct a paraconsistent Kripke structure $M := \langle S, S_0, R, L^{+}, L^{-}\rangle$ for PCTL 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. ∎

Theorem 3.4 (Embedding)
Let f be the mapping defined in Definition 3.1. For any formula α, α is valid in PCTL iff $f(\alpha)$ is valid in CTL.

Proof. By Lemmas 3.2 and 3.3. ∎

Theorem 3.5 (Decidability)
The model-checking, validity, and satisfiability problems for PCTL are decidable.

Proof. By the mapping f defined in Definition 3.1, a formula α of PCTL can finitely be transformed into the corresponding formula $f(\alpha)$ of CTL. By Lemmas 3.2 and 3.3 and Theorem 3.4, the model checking, validity, and satisfiability problems for PCTL can be transformed into those of CTL. Since the model checking, validity, and satisfiability problems for CTL are decidable, the problems for PCTL are also decidable. ∎

Since the mapping f from PCTL into CTL is a polynomial-time reduction, the complexity results for PCTL are the same as those for CTL, i.e., the validity, satisfiability, and model-checking problems for PCTL are EXPTIME-complete, deterministic EXPTIME-complete, and deterministic PTIME-complete, respectively.

§4 Illustrative Examples

Paraconsistent logic and temporal logic are known to be useful in medical informatics. We now consider examples of state structures for representing the health of non-smokers and smokers, as shown in Figure 1. In the state structure, the medical state of a person is described in a decision diagram where branching-tree structures and negative connectives from PCTL are employed. In this example, a paraconsistent negation $\sim\alpha$ in PCTL is used to express the negation of ambiguous concepts. For instance, if we cannot determine whether someone is healthy, the ambiguous concept *healthy* can be represented by asserting the inconsistent formula

$$healthy \wedge \sim healthy.$$

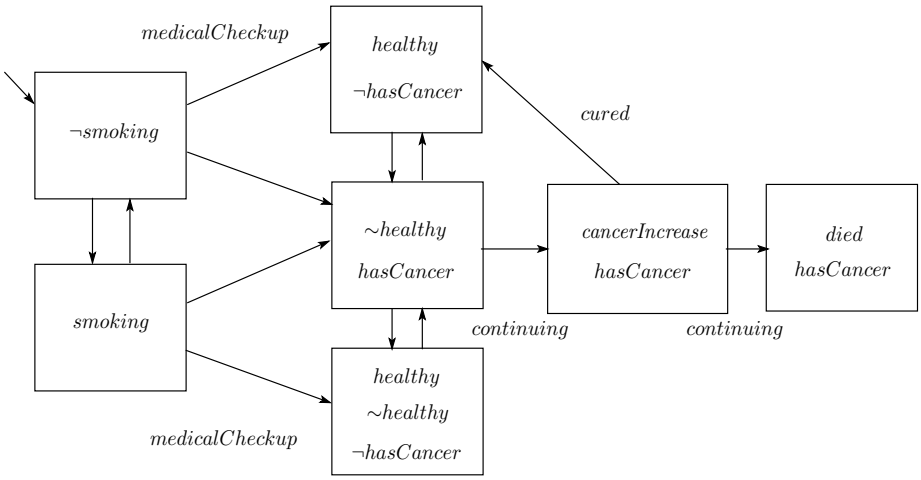

Fig. 1 State structure for representing the health of smokers and non-smokers

This is well formalized because $(healthy \wedge {\sim}healthy) \rightarrow \bot$ is not valid in paraconsistent logic. On the other hand, we can decide whether someone is smoking; the decision is represented by $smoking$ or $\neg smoking$, where $(smoking \wedge \neg smoking) \rightarrow \bot$ is valid in classical logic.

In the state structure of Figure 1, the initial state implies that a person is not smoking ($\neg smoking$ is true). The system can move to the other state to indicate that the person is smoking ($smoking$ is true). When a person undergoes a medical checkup, his or her state changes to one of the two states. Even if no cancer is detected in a smoker during the medical checkup, he or she is both healthy and not healthy, i.e., both $healthy$ and ${\sim}healthy$ are true because smoking is detrimental to health. If cancer is detected ($hasCancer$ is true) in a non-smoker (or smoker), then ${\sim}healthy$ is true. This means that the person is not healthy, but he or she may return to good health if the cancer does not increase. In these states, ${\sim}healthy$ represents ambiguous negative information that can be true at the same time as $healthy$, which represents positive information

Moreover, when the cancer increases, the diagnosis reveals worse cancer. If the cancer is cured, the patient will be healthy. Otherwise, if the cancer is not controlled, the patient will die.

We define a Kripke structure $M = \langle S, S_0, R, L^+, L^- \rangle$ that corresponds to the medical state structure as follows:

1. $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$,
2. $S_0 = \{s_0\}$,
3. $R = \{(s_0, s_1), (s_0, s_2), (s_0, s_3), (s_1, s_0), (s_1, s_3), (s_1, s_4), (s_2, s_3), (s_3, s_2), (s_3, s_4), (s_3, s_5), (s_4, s_3), (s_5, s_2), (s_5, s_6)\}$,
4. $L^+(s_0) = \emptyset$,
5. $L^+(s_1) = \{smoking\}$,
6. $L^+(s_2) = \{healthy\}$,
7. $L^+(s_3) = \{hasCancer\}$,
8. $L^+(s_4) = \{healthy\}$,
9. $L^+(s_5) = \{cancerIncrease, hasCancer\}$,
10. $L^+(s_6) = \{died, hasCancer\}$,
11. $L^-(s_0) = L^-(s_1) = L^-(s_2) = L^-(s_5) = L^-(s_6) = \emptyset$,
12. $L^-(s_3) = L^-(s_4) = \{healthy\}$.

We can verify the existence of a path that represents the required information in the structure M. For example, we can verify the following statement: "Is there a state in which a person is both healthy and not healthy?" This statement is expressed as:

$\mathrm{EF}(healthy \wedge {\sim}healthy)$.

The above statement is true because we have a path $s_0 \rightarrow s_1 \rightarrow s_4$ where $healthy \in L^+(s_4)$ and $healthy \in L^-(s_4)$.

In addition, the statement that "Is there a state in which a dead person will not be alive again?" can be expressed as:

$\mathrm{EF}(died \wedge \neg \mathrm{EF} \neg died)$.

This statement is verified as true because there is a path $s_0 \rightarrow s_3 \rightarrow s_5 \rightarrow s_6$ with $died \in L^+(s_6)$ but there is no path from s_6 to another state (i.e., there is no state x such that $s_6 \rightarrow x$).

In order to justify the usefulness of PCTL, two negative expressions can be differently interpreted as follows:

$\neg healthy$ (definitely unhealthy)
${\sim}healthy$ (not healthy)

The first statement indicates that a person is definitely unhealthy that is inconsistent with his or her healthy. The second statement means that we can say that a person is not healthy but he or she may be healthy.

The interpretation of the two negations leads to some useful verification examples for more complex statements. For example, the statement that "Is

there a state in which a person is not definitely unhealthy?" can be expressed as:

$\mathrm{EF}\neg\neg \mathit{healthy}$.

This statement is verified as true because there is a path from s_0 to two states $s \in \{s_2, s_4\}$ such that $M, s \models^+ \neg\neg \mathit{healthy}$. It is derived from that $M, s \not\models^+ \neg \mathit{healthy}$ iff $\mathit{healthy} \in L^-(s)$. Moreover, the statement that "Is there a state in which it is not true that a person is not healthy?" can be expressed as:

$\mathrm{EF}\neg\sim \mathit{healthy}$.

This statement is verified as true because there is a path $s_0 \rightarrow s_2$ with $M, s_2 \models^+ \neg\sim \mathit{healthy}$ (iff $M, s_2 \not\models^+ \sim \mathit{healthy}$ iff $M, s_2 \not\models^- \mathit{healthy}$ iff $\mathit{healthy} \notin L^-(s_2)$).

Importantly, the statement that "a person is not definitely unhealthy" holds in state s_4 but the statement that "it is not true that a person is not healthy" does not hold in state s_4. The two negations can be used to control inconsistencies and paraconsistencies in the examples.

§5 Related works

We now describe and compare some previous studies. Some CTL-based paraconsistent temporal logics and their variants have been studied by several researchers.[3, 4, 9, 14]

An application of multi-valued, paraconsistent model-checking for requirements elicitation in software engineering was studied by Easterbrook and Chechik[9] based on the *multi-valued computation tree logic* (χCTL) with the algebraic structures called *quasi-Boolean logics*. The Kripke structures for this framework were based on a multi-valued transition relation and a multi-valued valuation (labeling) function. The multi-valued valuation function was a very general setting because it can express n-valued truth values for any natural number n. The present framework, i.e., the N4-based framework, may be treated as a special case of the multi-valued framework since the two valuation functions used in this paper, which are inductively extended to $\models^+$ and $\models^-$, can be transformed into a four-valued valuation function. PCTL is not in a general setting as in χCTL, but it is simpler than χCTL, since it does not use any additional algebraic structures like the quasi-Boolean logics.

The χCTL model-checking framework was extended by Chechik and MacCaull[3] to include reasoning in logics with non-classical negation (in particular, intuitionistic, minimal, and Galois negations). An automatic verification

procedure for the cases in which the number of truth values is finite was developed on the basis of these logics. Some temporal operators in these logics were defined on the basis of χCTL, and they were shown to be computable using fixpoints.

A *quasi-classical temporal logic* QCTL was studied by Chen and Wu[4] in order to formalize reasoning on concurrent systems containing inconsistent information. In the work, *paraKripke structures* were introduced for QCTL, and a proof system that was sound and complete with respect to paraKripke structures was presented. In QCTL, a set of positive and negative objects, which is constructed from a set of atomic formulas, is used; in other words, a positive object $+p$ and a negative object $-p$ are obtained from an atomic formula p. Moreover, two satisfaction relations (a *strong satisfiability relation* $\models_{ts}$ and a *weak satisfiability relation* $\models_{tw}$) are used for QCTL. The positive and negative objects $+p$ and $-p$ in QCTL roughly correspond to p and $\sim p$ in PCTL, respectively. However, $\models_{ts}$ and $\models_{tw}$ in QCTL do not correspond to $\models^{+}$ and $\models^{-}$ in PCTL. Further, in QCTL, some clauses and quasi-clauses are used to define semantics, but this is not the case in PCTL.

In a previous study,[14] we introduced a paraconsistent four-valued "full" computation tree logic 4CTL* and a paraconsistent four-valued "locative full" computation tree logic 4LCTL*. Some bisimulation theorems for these logics and a theorem for embedding 4CTL* into CTL* were presented. However, the proof of the theorem for embedding 4CTL* into CTL* was rather complex and tedious since in the proof, the Kripke semantics (of 4CTL*) with $\models^{+}$ and $\models^{-}$ must be translated into a Kripke semantics with a single satisfaction relation. Namely, the proof in the study[14] needed two steps. Firstly, a single satisfaction semantics, which has only one satisfaction relation $\models$, was introduced. $\models$ includes the following clauses:

$$M, \pi \models \sim\sim\beta_1 \text{ iff } M, \pi \models \beta_1,$$
$$M, \pi \models \sim(\beta_1 \wedge \beta_2) \text{ iff } M, \pi \models \sim\beta_1 \text{ or } M, \pi \models \sim\beta_2,$$
$$M, \pi \models \sim \mathrm{G}\beta_1 \text{ iff } \exists k \geq 0\ [M, \pi^k \models \sim\beta_1]$$

where π represents a path (i.e., a sequences of states) and β_1, β_2 represent path formulas in 4CTL*. Then, an equivalence between this semantics with $\models$ and the dual satisfaction semantics with both $\models^{+}$ and $\models^{-}$ was proved by induction on formulas. Secondly, a theorem for embedding 4CTL* into CTL* was proved based on $\models$. By the equivalence theorem and the embedding theorem w.r.t. $\models$, we finally obtained the required embedding theorem w.r.t. $\models^{+}$ and $\models^{-}$. The

present proof of the theorem for embedding PCTL into CTL is simpler and easier than that of the theorem for embedding 4CTL* into CTL*, since the redundant step for providing a single satisfaction semantics and for showing an equivalence theorem between $\models$ and $(\models^+, \models^-)$ is not required in this proof.

From an implementation point of view, PCTL is more useful and efficient than 4CTL* because of the complexity results of PCTL and 4CTL*. The validity, satisfiability, and model-checking problems for PCTL are EXPTIME-complete, deterministic EXPTIME-complete and deterministic PTIME-complete, respectively. However, the validity, satisfiability, and model-checking problems for 4CTL* are 2EXPTIME-complete, deterministic 2EXPTIME-complete and PSPACE-complete, respectively. This computational difference makes an essential impact on an implementation of reasoning algorithms. Therefore, PCTL is regarded as a computational improvement on 4CTL*. That is, PCTL can use an efficient CTL-based model-checking algorithm but 4CTL* cannot use the algorithm because 4CTL* is CTL*-based. To our best knowledge, well-used model-checkers (e.g., NuSMV) are limited to CTL-based.

Some many-valued (including paraconsistent 4-valued) model-checkers have been studied by Gurfinkel et al.[10, 11] In software model checking, typical model checkers are used for refutation as well as verification because of their high bug-finding abilities. A software model checker Yasm[10, 11] which is based on χCTL is the first approach to combine verification and refutation based on the abstraction technique GEGAR.[6] Since PCTL is regarded as a refined special case of χCTL, the use of two satisfaction relations $\models^+$ and $\models^-$ in PCTL entails a theoretical justification for combining verification and refutation.

§6 Concluding remarks

In this paper, a new paraconsistent computation tree logic, PCTL, was introduced by combining CTL and Nelson's paraconsistent logic N4. This logic could be used appropriately in medical reasoning to deal with inconsistent data and uncertain concepts. The theorem for embedding PCTL into CTL was proved. The validity, satisfiability, and model-checking problems of PCTL were shown to be decidable. The embedding and decidability results indicate that we can reuse the existing CTL-based algorithms to test the validity, satisfiability, and model-checking. Thus, it was shown that PCTL can be used as an executable logic to represent temporal reasoning on paraconsistency. As a future task, we believe that over- and under-approximating abstractions can be

appropriately combined using a PCTL-based model-checker with $\models^+$ and $\models^-$.[6] We also believe that PCTL can be extensively used for inconsistency-tolerant and uncertainty reasoning, since N4 and its variants are known to be very useful for a wide range of applications such as logic programming and knowledge representations.[19, 15, 13, 23, 28]

Although another important property called *paracompleteness*[2, 17] is not discussed so far, the paraconsistent negation connective $\sim$ in PCTL is also paracomplete. The paracompleteness is regarded as the dual notion of paraconsistency: a paracomplete negation $\sim$ is a unary operator that does not satisfy the law of *excluded middle* $\alpha \vee \sim\alpha$. In (extensions of) standard classical propositional logic, $\alpha \vee \neg\alpha$ is valid. This means that the information represented by the classical negation connective $\neg$ is *complete information*: every formula α is either true or not true in a model. Representing only complete information is plausible in classical mathematics, which is a discipline handling eternal truth and falsehood. The statements of classical mathematics do not change their truth value in the course of time. The assumption of complete information is, however, inadequate when it comes to representing the information available to real world agents. We wish to explore the consequences of *incomplete* information about computer and information systems, and then it is desirable to avail of a paracomplete negation connective.

A limitation or demerit of extending CTL with paraconsistency and paracompleteness may be that to construct a good proof (or deductive) system (e.g., Gentzen-type sequent calculus) for such a logic is difficult. In order to avoid such a difficulty, we need other base temporal logics such as *linear-time temporal logic* (LTL).[25]

We conclude with some remarks on alternatives to PCTL. In a Kripke structure $\langle S, S_0, R, L^+, L^- \rangle$, imposing the condition $\forall s \in S\ [L^+(s) \cap L^+(s) = \emptyset]$ is equivalent to the asserting that the underlying logic is a non-paraconsistent three-valued logic. Now, the logic obtained by imposing the condition on PCTL is called 3CTL. Although 3CTL is not paraconsistent, the corresponding embedding and decidability results can be obtained in a similar manner. The results for PCTL and 3CTL can also be adapted and applied to LTL. Our framework for paraconsistent negation may thus be applicable to a wide range of temporal and non-classical logics.

Acknowledgment This research was supported in part by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B), No.17700164 and No.20700015, and was also supported by the Alexander von Humboldt Foundation.

References

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

2) J.-Y. Béziau, The future of paraconsistent logic, *Logical Studies* 2, Online available, 1999.

3) M. Chechik and W. MacCaull, CTL model-checking over logics with non-classical negations, Proceedings of the 33rd IEEE International Conference on Multi-valued logics (ISMVL'03), pp. 293–300, 2003.

4) D. Chen and J. Wu, Reasoning about inconsistent concurrent systems: A non-classical temporal logic, Lecture Notes in Computer Science 3831, pp. 207-217, 2006.

5) E.M. Clarke and E.A. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic, Lecture Notes in Computer Science 131, pp. 52–71, 1981.

6) E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, Counterexample-guided abstraction refinement for symbolic model checking, *Journal of the ACM* 50 (5), pp. 752–794, 2003.

7) E.M. Clarke, O. Grumberg, and D.A. Peled, Model checking, The MIT Press, 1999.

8) N.C.A. da Costa, J. Béziau and O.A. Bueno, Aspects of paraconsistent logic, *Bulletin of the IGPL* 3 (4), 597–614, 1995.

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

10) A. Gurfinkel, O. Wei, and M. Chechik, Yasm: a software model-checker for verification and refutation, Proceedings of the 18th International Conference, Computer Aided Verification (CAV'06), Lecture Notes in Computer Science 4144, pp. 170–174, 2006.

11) A. Gurfinkel, and M. Chechik, Why wast a perfectly good abstraction?, Proceedings of the 12th International Conference, Tools and Algorithms for Construction and Analysis of Systems (TACAS'06), Lecture Notes in Computer Science 3920, pp. 212–226, 2006.

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

13) N. Kamide, Linear and affine logics with temporal, spatial and epistemic operators, *Theoretical Computer Science* 353 (1-3), pp. 165–207, 2006.

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

15) N. Kamide, A uniform proof-theoretic foundation for abstract paraconsistent logic programming, *Journal of Functional and Logic Programming*, pp. 1–36, 2007.

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

17) N. Kamide and H. Wansing. Combining linear-time temporal logic with constructiveness and paraconsistency, *Journal of Applied Logic*, 8:33–61, 2010.

18) K. Kaneiwa, Order-sorted logic programming with predicate hierarchy, *Artificial Intelligence* 158 (2), pp. 155–188, 2004.

19) K. Kaneiwa, Description logics with contraries, contradictories, and subcontraries, *New Generation Computing* 25 (4), pp. 443–468, 2007.

20) K. Kaneiwa and K. Satoh, On the complexities of consistency checking for restricted UML class diagrams, *Theoretical Computer Science* 411(2), pp. 301–323, 2010.

21) T. Murata, V.S. Subrahmanian and T. Wakayama, A Petri net model for reasoning in the presence of inconsistency, *IEEE Transactions on Knowledge and Data Engineering* 3 (3), pp. 281–292, 1991.

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

23) S.P. Odintsov and H. Wansing, Inconsistency-tolerant description logic: Motivation and basic systems, In: V.F. Hendricks and J. Malinowski, Editors, Trends in Logic: 50 Years of Studia Logica, Kluwer Academic Publishers, Dordrecht, pp. 301–335, 2003.

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

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

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

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

28) G. Wagner, Logic programming with strong negation and inexact predicates, *Journal of Logic and Computation* 1 (6), pp. 835–859, 1991.

29) H. Wansing, The logic of information structures, Lecture Notes in Artificial Intelligence 681, 163 pages, 1993.