PAPER

Complexity of Strong Satisfiability Problems for Reactive System Specifications*

Masaya SHIMAKAWA^{†a)}, Shigeki HAGIHARA[†], *Nonmembers*, and Naoki YONEZAKI[†], *Member*

SUMMARY Many fatal accidents involving safety-critical reactive systems have occurred in unexpected situations, which were not considered during the design and test phases of system development. To prevent such accidents, reactive systems should be designed to respond appropriately to any request from an environment at any time. Verifying this property during the specification phase reduces the development costs of safety-critical reactive systems. This property of a specification is commonly known as realizability. The complexity of the realizability problem is 2EXPTIME-complete. We have introduced the concept of strong satisfiability, which is a necessary condition for realizability. Many practical unrealizable specifications are also strongly unsatisfiable. In this paper, we show that the complexity of the strong satisfiability problem is EXPSPACE-complete. This means that strong satisfiability offers the advantage of lower complexity for analysis, compared to realizability. Moreover, we show that the strong satisfiability problem remains EXPSPACE-complete even when only formulae with a temporal depth of at most 2 are allowed.

key words: reactive system, strong satisfiability, realizability, complexity, LTL specification

1. Introduction

A reactive system is a system that responds to requests from an environment in a timely fashion. The systems used to control elevators or vending machines are typical examples of reactive systems. Many safety-critical systems, such as the systems that control nuclear power plants and air traffic control systems, are also considered reactive systems.

In designing a system of this kind, the requirements are analyzed and then described as specifications for the system. If a specification has a flaw, such as inappropriate case-splitting, a developed system may encounter unintended situations. Indeed, many fatal accidents involving safety-critical reactive systems have occurred in unexpected situations, which were not considered during the design and test phases of system development. It is therefore important to ensure that a specification does not possess this kind of flaw [1].

More precisely, a reactive system specification must have a model that can respond in a timely fashion to any request at any time. This property is called realizability, and was introduced in [2], [3]. In [3], A. Pnueli and R. Rosner

showed that a reactive system can be synthesized from a realizable specification.

In [4], [5], we introduced the concept of strong satisfiability, which is a necessary condition for realizability. Many practical unrealizable specifications are also strongly unsatisfiable [5]. In [6], we presented a method for checking whether or not a specification satisfies strong satisfiability. Furthermore, in [7], we proposed techniques for identifying the flaws in strongly unsatisfiable specifications. Another approach for checking strong satisfiability was introduced in [8].

However, there has been no discussion of the complexity of the strong satisfiability problem, which is an important consideration, because such knowledge would be useful for obtaining an efficient verification procedure for strong satisfiability.

In this paper, we show that the complexity of the strong satisfiability problem for a specification written in linear temporal logic (LTL) is EXPSPACE-complete. Because the complexity of the realizability problem is 2EXPTIME-complete, this means that strong satisfiability offers the advantage of lower complexity for analysis, compared to realizability. Moreover, we show that the strong satisfiability problem remains EXPSPACE-complete even when only formulae with a temporal depth of at most 2 are allowed.

The remainder of this paper is organized as follows. In Sect. 2, we introduce the concepts of reactive system, LTL as a specification language, and strong satisfiability, which is a necessary condition for the realizability of a reactive system specification. In Sect. 3, we show that the strong satisfiability problem for a specification written in LTL is EXPSPACE-complete. In Sect. 4, we show that the strong satisfiability problem remains EXPSPACE-complete even when only formulae with a temporal depth of at most 2 are allowed. In Sect. 5, we discuss the complexity of the strong satisfiability problem for LTL in relation to that of other problems for LTL and other temporal logics. We present our conclusions in Sect. 6.

2. Specifications for Reactive Systems and Their Properties

2.1 Reactive Systems

A reactive system (illustrated in Fig. 1) is a system that responds to requests from an environment in a timely fashion.

Definition 1 (Reactive system): A reactive system RS is a

Manuscript received March 4, 2013.

[†]The authors are with the Department of Computer Science, Tokyo Institute of Technology, Tokyo, 152-8552 Japan.

*This work is based on "Complexity of Checking Strong Satisfiability of Reactive System Specifications", by M. Shimakawa, S. Hagihara, and N. Yonezaki which appeared in the proceeding of International Conference on Advances in Information Technology and Communication, AIT 2012

a) E-mail: masaya@fmx.cs.titech.ac.jp

DOI: 10.1587/transinf.E96.D.2187

Fig. 1 A reactive system.

triple $\langle X, Y, r \rangle$, where X is a set of events caused by an environment, Y is a set of events caused by the system, and $r : (2^X)^+ \rightarrow 2^Y$ is a reaction function.

We refer to events caused by the environment as ‘input events,’ and those caused by the system as ‘output events.’ The set $(2^X)^+$ is the set of all finite sequences of sets of input events. A reaction function r relates sequences of sets of previously occurring input events with a set of current output events.

2.2 Language for Describing Reactive System Specifications

The timing of input and output events is an essential element of a reactive system. Modal logics are widely used in computer science. Among these, temporal logics have often been employed in the analysis of reactive systems, following the application of such logics to program semantics by Z. Manna and A. Pnueli [9]. A propositional linear temporal logic (LTL) [10] with an ‘until’ operator is a suitable language for describing the timing of events. In this paper, we use LTL to describe the specifications of reactive systems. We treat input events and output events as atomic propositions.

2.2.1 Syntax

Formulae in LTL are inductively defined as follows:

- Atomic propositions are formulae; i.e., input events and output events are formulae.
- $f \wedge g$, $\neg f$, $\mathbf{X}f$, $f\mathbf{U}g$ are formulae if f and g are formulae.

Intuitively, $f \wedge g$ and $\neg f$ represent the statements ‘both f and g hold’ and ‘ f does not hold,’ respectively. The notation $\mathbf{X}f$ means that ‘ f holds the next time,’ while $f\mathbf{U}g$ means that ‘ f always holds until g holds.’ The notations $f \vee g$, $f \rightarrow g$, $f \leftrightarrow g$, $f \oplus g$, $\top$, $\mathbf{F}f$, and $\mathbf{G}f$ are abbreviations for $\neg(\neg f \wedge \neg g)$, $\neg(f \wedge \neg g)$, $\neg(f \wedge \neg g) \wedge \neg(\neg f \wedge g)$, $\neg(f \leftrightarrow g)$, $\neg\perp$, $\top\mathbf{U}f$, and $\neg\mathbf{F}\neg f$ respectively, where $\perp$ is an atomic proposition representing ‘falsity.’

2.2.2 Semantics

A behavior is an infinite sequence of sets of events. Let i be an index such that $i \geq 0$. The i -th set of a behavior σ is denoted by $\sigma[i]$. The i -th suffix of a behavior σ is denoted by $\sigma[i \dots]$. When a behavior σ satisfies a formula f , we write $\sigma \models f$, and inductively define this relation as follows:

- $\sigma \models p$ iff $p \in \sigma[0]$
- $\sigma \not\models \perp$
- $\sigma \models f \wedge g$ iff $\sigma \models f$ and $\sigma \models g$
- $\sigma \models \neg f$ iff $\sigma \not\models f$
- $\sigma \models \mathbf{X}f$ iff $\sigma[1 \dots] \models f$
- $\sigma \models f\mathbf{U}g$ iff $\exists j \geq 0. (\sigma[j \dots] \models g) \text{ and } \forall k (0 \leq k < j. \sigma[k \dots] \not\models f)$

We say that f is satisfiable if there exists a σ that satisfies f .

2.3 Properties of Reactive System Specifications

It is important for reactive system specifications to satisfy realizability. Realizability requires that there exist a reactive system such that for any input events with any timing, the system produces output events such that the specification holds.

Definition 2 (Realizability): A specification $Spec$ is *realizable* if the following holds:

$$\exists RS \forall \tilde{a} (behave_{RS}(\tilde{a}) \models Spec),$$

where $\tilde{a}$ is an infinite sequence of sets of input events; i.e., $\tilde{a} \in (2^X)^\omega$. $behave_{RS}(\tilde{a})$ is the infinite behavior of $\tilde{a}$ caused by RS , defined as follows. If $\tilde{a} = a_0 a_1 \dots$,

$$behave_{RS}(\tilde{a}) = (a_0 \cup b_0)(a_1 \cup b_1) \dots,$$

where b_i is a set of output events caused by RS ; i.e., $b_i = r(a_0 \dots a_i)$, and $\cup$ denotes the union of two sets.

The following property was shown to be a necessary condition for realizability in [4].

Definition 3 (Strong satisfiability): A specification $Spec$ is *strongly satisfiable* if the following holds:

$$\forall \tilde{a} \exists \tilde{b} (\langle \tilde{a}, \tilde{b} \rangle \models Spec),$$

where $\tilde{b}$ is an infinite sequence of sets of output events; i.e., $\tilde{b} \in (2^Y)^\omega$. If $\tilde{a} = a_0 a_1 \dots$ and $\tilde{b} = b_0 b_1 \dots$, then $\langle \tilde{a}, \tilde{b} \rangle$ is defined by $\langle \tilde{a}, \tilde{b} \rangle = (a_0 \cup b_0)(a_1 \cup b_1) \dots$.

Intuitively, strong satisfiability is the property that if a reactive system is given an infinite sequence of sets of future input events, the system can determine an infinite sequence of sets of future output events. Strong satisfiability is a necessary condition for realizability; i.e., all realizable specifications are strongly satisfiable. Conversely, many practical strongly satisfiable specifications are also realizable.

Example 1: Let us consider a simple example of a door

control system. The initial specification is as follows.

1. The door has two buttons: an open button and a close button.
2. If the open button is pushed, the door eventually opens.
3. While the close button is pushed, the door remains shut.

The events ‘the open button is pushed’ and ‘the close button is pushed’ are both input events. We denote these events by x_1 and x_2 , respectively. The event ‘the door is open (closed)’ is an output event. We denote this event by y (resp., $\neg y$). The initial specification is then represented by $Spec_1 : \mathbf{G}((x_1 \rightarrow \mathbf{F}y) \wedge (x_2 \rightarrow \neg y))$ in LTL. This specification is not strongly satisfiable, and consequently unrealizable, due to the fact that there is no response that satisfies $Spec_1$ for the environmental behavior in which the close button is still being pushed after the open button has been pushed. Formally, for $\tilde{a} = \{x_1, x_2\}\{x_2\}\{x_2\} \dots$, $\exists \tilde{b}(\langle \tilde{a}, \tilde{b} \rangle \models Spec_1)$ does not hold. Hence $\forall \tilde{a} \exists \tilde{b}(\langle \tilde{a}, \tilde{b} \rangle \models Spec_1)$ does not hold.

However, suppose the constraint 3 in the initial specification can be weakened to 3’:

- 3’. If the close button is pushed, the door eventually closes.

Then the modified specification is represented by $\mathbf{G}((x_1 \rightarrow \mathbf{F}y) \wedge (x_2 \rightarrow \mathbf{F}\neg y))$, and this is both strongly satisfiable and realizable.

3. Complexity of Checking Strong Satisfiability

In this section, we show that the strong satisfiability problem (i.e., whether or not a specification written in LTL satisfies strong satisfiability) is EXPSPACE-complete. In other words, (1) the strong satisfiability problem is in the class EXPSPACE (the class of problems solvable in $O(2^{p(n)})$ space by a deterministic Turing machine, where $p(n)$ is a polynomial function of the input size n), and (2) all the problems in EXPSPACE are reducible to the strong satisfiability problem.

3.1 Upper Bound

First, we show that the strong satisfiability problem is in EXPSPACE. We prove the existence of a procedure that uses $O(2^{p(n)})$ space to check strong satisfiability. This procedure is a modified version of the one introduced in [6].

A *non-deterministic Büchi automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, F \rangle$, where Σ is an alphabet, Q is a finite set of states, q_0 is an initial state, $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation, and $F \subseteq Q$ is a set of final states. A run of $\mathcal{A}$ on an ω -word $\alpha = \alpha[0]\alpha[1] \dots$ is an infinite sequence $\gamma = \gamma[0]\gamma[1] \dots$ of states, where $\gamma[0] = q_0$ and $(\gamma[i], \alpha[i], \gamma[i+1]) \in \delta$ for all $i \geq 0$. We say that $\mathcal{A}$ accepts α , if there is a run γ on α such that $\text{In}(\gamma) \cap F \neq \emptyset$ holds, where $\text{In}(\gamma)$ is the set of states that occur infinitely often in γ . The set of ω -words accepted by $\mathcal{A}$ is called the language accepted by $\mathcal{A}$, and is denoted by $L(\mathcal{A})$.

Let $Spec$ be a specification written in LTL. We can

check the strong satisfiability of $Spec$ via the following procedure.

1. We obtain a non-deterministic Büchi automaton $\mathcal{A} = \langle 2^{X \cup Y}, Q, q_0, \delta, F \rangle$ such that $L(\mathcal{A}) = \{\sigma \mid \sigma \models Spec\}$ holds.
2. Let $\mathcal{A}' = \langle 2^X, Q, q_0, \delta', F \rangle$ be a non-deterministic Büchi automaton obtained by restricting $\mathcal{A}$ to only input events, where $\delta' = \{(q, a, q') \mid \exists b (q, a \cup b, q') \in \delta\}$. Note that $L(\mathcal{A}') = \{\tilde{a} \mid \exists \tilde{b} \langle \tilde{a}, \tilde{b} \rangle \in L(\mathcal{A})\}$ holds due to the definition of δ' .
3. We check whether or not $\mathcal{A}'$ is universally acceptable (which means that $L(\mathcal{A}') = (2^X)^\omega$). If it is universally acceptable, we conclude that $Spec$ is strongly satisfiable. If it is not universally acceptable, we conclude that $Spec$ is not strongly satisfiable.

$\mathcal{A}$ can be constructed within $O(2^{|Spec|})$ space, and the size of $\mathcal{A}$ is also $O(2^{|Spec|})$ [11], where $|Spec|$ is the length of $Spec$. Since $\mathcal{A}'$ is obtained by projection, $\mathcal{A}'$ can be constructed within $O(|\mathcal{A}|)$ space, and the size of $\mathcal{A}'$ is $O(|\mathcal{A}|)$, where $|\mathcal{A}|$ is the size of $\mathcal{A}$. The universality problem for a Büchi automaton is in PSPACE [12], and Step 3 is accomplished within $O(p(|\mathcal{A}'|))$ space. Therefore, we can check strong satisfiability in $O(2^{|Spec|})$ space, and we can conclude that the strong satisfiability problem is in EXPSPACE.

Theorem 1: The strong satisfiability problem for specifications written in LTL is in the complexity class EXPSPACE.

3.2 Lower Bound

In this subsection, we show that the strong satisfiability problem is EXPSPACE-hard, by providing polynomial time reduction from the EXP-corridor tiling problem [13] to the strong satisfiability problem. The EXP-corridor tiling problem is EXPSPACE-complete.

Definition 4 (EXP-corridor tiling problem): The EXP-corridor tiling problem is as follows: For a given $(T, H, V, t_{init}, t_{final}, m)$, where T is a finite set of tile types, $H, V \subseteq T \times T$ are horizontal and vertical adjacency constraints, $t_{init}, t_{final} \in T$ are the initial and final tile types, and $m \in \mathbb{N}$, determine whether or not there exists a $k \in \mathbb{N}$ and an assignment function $f : [0, (2^m - 1)] \times [0, k] \rightarrow T$ such that the following conditions are satisfied:

1. $f(0, 0) = t_{init}$
2. $f(2^m - 1, k) = t_{final}$
3. for any $0 \leq j \leq k, 0 \leq i < 2^m - 1, (f(i, j), f(i+1, j)) \in H$ holds.
4. for any $0 \leq i \leq 2^m - 1, 0 \leq j < k, (f(i, j), f(i, j+1)) \in V$ holds.

As Fig. 2 shows, the tiling grid has $2^m \times \omega$ points. Intuitively, this leads to the following question: “For a given tiling grid, does there exist k such that a tile can be assigned to each point (i, j) for which $0 \leq i < 2^m$ and $0 \leq j \leq k$,

Fig. 2 The EXP-corridor tiling problem.

satisfying conditions 1–4?” Condition 1 is the condition for the initial tile, and states that the tile of type t_{init} is assigned to the leftmost, topmost point. Condition 2 is the condition for the final tile, and states that the tile of type t_{final} is assigned to the rightmost, bottommost point. Condition 3 is the condition for horizontal lines, and states that horizontally neighboring tiles satisfy the horizontal adjacency constraint H . Condition 4 is the analogous condition for vertical lines.

We provide polynomial time reduction from the EXP-corridor tiling problem to the complement of the strong satisfiability problem. That is, for the EXP-corridor tiling problem $(T, H, V, t_{init}, t_{final}, m)$, we construct a formula φ_{tiling} such that $\exists \tilde{a} \forall \tilde{b} (\langle \tilde{a}, \tilde{b} \rangle \models \neg \varphi_{tiling})$ holds if and only if the answer to the tiling problem $(T, H, V, t_{init}, t_{final}, m)$ is affirmative.

In this reduction, we relate “there exists a tiling assignment” in the tiling problem to “there exists an infinite sequence of sets of input events.” Furthermore, “the tiling assignment satisfies the conditions” is translated to “the corresponding infinite sequence of sets of input events does not satisfy φ_{tiling} for any infinite sequence of sets of output events.” That is, a tiling assignment is represented by an infinite sequence $\tilde{a}$ of sets of input events, and the conditions in the tiling problem are represented by $\forall \tilde{b} (\langle \tilde{a}, \tilde{b} \rangle \models \neg \varphi_{tiling})$, universally quantified on infinite sequences of sets of output events.

In the representation of conditions 1–3, we do not need meta-level universal quantification; i.e., conditions 1–3 are represented by pure LTL formulae over input events only. (This representation is similar to that of the reduction in [14]). To represent condition 4, we do need meta-level universal quantification. The representation of condition 4 is the key issue in our reduction.

(a) Input events

To relate an infinite sequence of sets of input events to a tiling assignment, we introduce the following input events.

- x_t for each $t \in T$: “the tile of type t is placed on the point (i, j) ” is translated to “the input events x_t occur at time $i + (2^m) \cdot j$.”
- end : “tiling is finished at the point (i, j) ” is related to

“ end occurs at time $i + (2^m) \cdot j$.”

- $c_0, \dots, c_{m-1}$: There are m bit counters that keep track of the amount of time. By checking these counters, we can identify a column of the tiling grid.

(b) Output events

We introduce the following output events.

- $y_0, \dots, y_{m-1}$: These are used to identify a column.

(c) The formula φ_{tiling}

The formula φ_{tiling} is the negation of the conjunction of the formulae (1)–(6) listed below. Here we use the following abbreviations:

$$\begin{aligned} \bar{c} = 0 &\equiv \bigwedge_{0 \leq i < m} \neg c_i \\ \bar{c} = 2^m - 1 &\equiv \bigwedge_{0 \leq i < m} c_i \\ \bar{c} = \bar{y} &\equiv \bigwedge_{0 \leq i < m} (c_i \leftrightarrow y_i) \end{aligned}$$

- The constraint for m bit counters $c_0, \dots, c_{m-1}$.

$$\left(\bigwedge_{0 \leq i < m} \neg c_i \right) \wedge \left(\bigwedge_{0 \leq i < m} \mathbf{G}((c_i \oplus \bigwedge_{0 \leq j < i} c_j) \leftrightarrow \mathbf{X}c_i) \right) \quad (1)$$

This represents the statement “the value of $\bar{c}$ is initially 0, and is incremented at each time step,” which means that the value of $\bar{c}$ represents the current column number.

- Single assignment of tile types.

$$\bigwedge_{t \in T} \mathbf{G}(x_t \rightarrow \bigwedge_{t' \neq t} \neg x_{t'}) \wedge \mathbf{G}(\neg end \rightarrow \bigvee_{t \in T} x_t) \quad (2)$$

This represents the statement “at most one tile type is assigned to each grid point, and if tiling is not finished, some tile type must be assigned.”

- The constraint for condition 1.

$$x_{t_{init}} \quad (3)$$

This represents the statement “the initial tile of type t_{init} is placed on the point $(0, 0)$.”

- The constraint for condition 2.

$$\neg end \mathbf{U} (\neg end \wedge \bar{c} = 2^m - 1 \wedge x_{t_{final}} \wedge \mathbf{X} \mathbf{G} end) \quad (4)$$

This represents the statement “the final tile of type t_{final} is placed on some point in column $2^m - 1$, and tiling is finished.”

- The constraint for condition 3.

$$\mathbf{G}(\bar{c} \neq 2^m - 1 \wedge \neg end \rightarrow \bigvee_{(t,t') \in H} (x_t \wedge \mathbf{X}x_{t'})) \quad (5)$$

This represents the statement “if tiling is not finished

and the current point is not in the $(2^m - 1)$ -th column (i.e., a point exists to the right of it), then the tile at the current point and the tile at the point to the right of it (which corresponds to the next time point) satisfy condition H ."

- The constraint for condition 4.

$$\begin{aligned} & \left(\bigwedge_{0 \leq i < m} \mathbf{G}(y_i \leftrightarrow \mathbf{X}y_i) \right) \rightarrow \\ & \mathbf{G}((\bar{c} = \bar{y} \wedge \mathbf{X}\mathbf{F}(\neg \text{end} \wedge \bar{c} = 0)) \rightarrow \\ & \bigvee_{(t,t') \in V} (x_t \wedge \mathbf{X}((\bar{c} \neq \bar{y})\mathbf{U}(\bar{c} = \bar{y} \wedge x_{t'}))) \end{aligned} \quad (6)$$

This represents the statement "if the value of $\bar{y}$ behaves as a constant, for any current point in the column indicated by $\bar{y}$, if tiling is not finished in the row just below the current row, the tile at the current point and the tile at the point beneath it (which corresponds to the time point after 2^m time units) satisfy condition V ." The point beneath the current point is characterized by the phrase "the first time point after the current point at which the value of the counter (the column number) again becomes $\bar{y}$." Because $y_0, y_1, \dots, y_{m-1}$ are output events, $\forall \bar{b}(\langle \bar{a}, \bar{b} \rangle \models (6))$ represents the statement "for any column, tiles in the column satisfy condition V ," which means "any tiles satisfy condition V ."

Theorem 2: The strong satisfiability problem for specifications written in LTL is EXPSPACE-hard.

Proof. As mentioned above, we can construct a formula φ_{tiling} such that the answer to the EXP-corridor tiling problem is affirmative if and only if the corresponding φ_{tiling} is not strongly satisfiable. The size of φ_{tiling} is polynomial in the size of the problem $(T, H, V, t_{\text{init}}, t_{\text{final}}, m)$, and φ_{tiling} can be constructed in polynomial time. Therefore, the EXP-corridor tiling problem is reducible to the complement of the strong satisfiability problem. Because the EXP-corridor tiling problem is EXPSPACE-complete, the complement of the strong satisfiability problem is EXPSPACE-hard, and the strong satisfiability problem is co-EXPSPACE-hard. Because EXPSPACE=co-EXPSPACE, the strong satisfiability problem is also EXPSPACE-hard. $\square$

4. Complexity for LTL Fragments with a Fixed Upper Bound on Temporal Depth

In this section, we consider the complexity of strong satisfiability problems for LTL fragments obtained by limiting the temporal depth (nesting depth of temporal operators).

The temporal depth $td(\varphi)$ of an LTL formula φ is defined by:

- $td(p) = 0$
- $td(\neg\psi) = td(\psi)$
- $td(\mathbf{X}\psi) = td(\psi) + 1$
- $td(\psi_1 \mathbf{U} \psi_2) = \max(td(\psi_1), td(\psi_2)) + 1$

We define LTL fragments by limiting the temporal depth:

For $i \in \mathbb{N}$, $\text{LTL}(i) = \{\varphi \mid td(\varphi) \leq i\}$.

We show that the strong satisfiability problem remains EXPSPACE-complete even for $\text{LTL}(2)$. We prove this by providing a reduction from the strong satisfiability problem for LTL to the strong satisfiability problem for $\text{LTL}(2)$.

We use the following lemmas for the reduction.

Lemma 1: Let Spec be a specification described in LTL over $X \cup Y$. Then for any $\sigma \in (2^{X \cup Y})^\omega$, $\varphi \in \text{sub}(\text{Spec})$, $c \notin X \cup Y$, the following holds:

$$\begin{aligned} \sigma \models \text{Spec} \\ \iff \exists \tilde{c} \in (2^{|c|})^\omega. \langle \sigma, \tilde{c} \rangle \models \text{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi), \end{aligned}$$

where $\text{sub}(\text{Spec})$ is the set of subformulae of Spec , and $\text{Spec}[c/\varphi]$ is the formula obtained by replacing all occurrences of φ in Spec with c .

Proof. $\exists \tilde{c} \in (2^{|c|})^\omega. \langle \sigma, \tilde{c} \rangle \models \text{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi) \iff \langle \sigma, \tilde{c} \rangle \models \text{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi)$, where $\tilde{c}$ is the sequence defined by

$$\tilde{c}[i] := \begin{cases} \emptyset & \text{if } \sigma[i \dots] \not\models \varphi, \\ \{c\} & \text{if } \sigma[i \dots] \models \varphi. \end{cases}$$

(This is because $\langle \sigma, \tilde{c} \rangle \models \mathbf{G}(c \leftrightarrow \varphi)$ holds.)

$$\iff \langle \sigma, \tilde{c} \rangle \models \text{Spec}$$

(This is because for all i , $\langle \sigma, \tilde{c} \rangle[i \dots] \models \varphi$ holds if and only if $\langle \sigma, \tilde{c} \rangle[i \dots] \models c$ holds.)

$$\iff \sigma \models \text{Spec}$$

(This is because c does not occur in Spec .) $\square$

Lemma 2: Let Spec be a specification described in LTL, let X be the set of input events of Spec , and let Y be the set of output events of Spec . For $\varphi \in \text{sub}(\text{Spec})$ and a new output event c , the specification Spec is strongly satisfiable if and only if the specification $\text{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi)$ is strongly satisfiable.

Proof. Spec is strongly satisfiable

$$\iff \forall \tilde{a} \exists \tilde{b}(\langle \tilde{a}, \tilde{b} \rangle \models \text{Spec})$$

$$\iff \forall \tilde{a} \exists \tilde{b} \exists \tilde{c} \in (2^{|c|})^\omega(\langle \tilde{a}, \tilde{b}, \tilde{c} \rangle \models \text{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi))$$

(by Lemma 1)

$$\iff \forall \tilde{a} \exists \tilde{b}' \in (2^{Y \cup \{c\}})^\omega(\langle \tilde{a}, \tilde{b}' \rangle \models \text{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi))$$

$$\iff \text{Spec}[c/\varphi] \wedge \mathbf{G}(c \leftrightarrow \varphi) \text{ is strongly satisfiable.} \quad \square$$

Lemma 2 suggests a transformation that decreases the temporal depth of a specification. For example, for the specification $\text{Spec} = x\mathbf{U}(\mathbf{X}\mathbf{F}\mathbf{G}y)$ such that $td(\text{Spec}) = 4$, we can obtain the following specification:

$$\text{Spec}_1 = x\mathbf{U}(\mathbf{X}\mathbf{c}\mathbf{F}\mathbf{G}y) \wedge \mathbf{G}(\mathbf{c}\mathbf{F}\mathbf{G}y \leftrightarrow \mathbf{F}\mathbf{G}y),$$

where $td(\text{Spec}_1) = 3$, and Spec is strongly satisfiable if and only if Spec_1 is strongly satisfiable. By applying the transformation repeatedly, we can obtain a specification Spec' , where $td(\text{Spec}') \leq 2$, and Spec is strongly satisfiable if and only if Spec' is strongly satisfiable. This implies that the strong satisfiability problem for $\text{LTL}(2)$ is EXPSPACE-complete.

Theorem 3: The strong satisfiability problem for reactive system specifications described in LTL(2) is EXPSPACE-complete.

Proof. We obtain a polynomial time reduction from the strong satisfiability problem for LTL to the strong satisfiability problem for LTL(2). That is, given a specification $Spec$ described in LTL over $X \cup Y$, we construct a specification $Spec'$ in LTL(2) such that $Spec$ is strongly satisfiable if and only if $Spec'$ is strongly satisfiable. The specification $Spec'$ is defined as follows:

- Set of input events: X .
- Set of output events: $Y \cup C$, where $C = \{c_\varphi \mid \varphi \in \text{sub}(Spec)\}$.
- $Spec' = c_{Spec} \wedge \bigwedge_{\varphi \in \text{sub}(Spec)} \mathbf{G}(c_\varphi \leftrightarrow f(\varphi))$ where $f(\varphi)$ is defined as follows:
 - φ if $\varphi \in X \cup Y$,
 - $\neg c_{\varphi'}$ if $\varphi = \neg\varphi'$,
 - $c_{\varphi'} \wedge c_{\varphi''}$ if $\varphi = \varphi' \wedge \varphi''$,
 - $\mathbf{X}c_{\varphi'}$ if $\varphi = \mathbf{X}\varphi'$,
 - $c_{\varphi'} \mathbf{U} c_{\varphi''}$ if $\varphi = \varphi' \mathbf{U} \varphi''$.

The specification $Spec'$ equals the specification obtained by applying the transformation of Lemma 2, which is related to all subformulae of $Spec$, in decreasing order of length. Thus, $Spec$ is strongly satisfiable if and only if $Spec'$ is strongly satisfiable. Because the temporal depth of $f(\varphi)$ is at most 1 for any $\varphi \in \text{sub}(Spec)$, the temporal depth of $Spec'$ is at most 2. The size of $Spec'$ is polynomial in the size of $Spec$, and $Spec'$ can be constructed in polynomial time.

As was proved in Sect. 3, the strong satisfiability problem for LTL is EXPSPACE-complete. Because this is reducible to the strong satisfiability problem for LTL(2), the strong satisfiability problem is EXPSPACE-hard even for LTL(2). Therefore, the strong satisfiability problem for LTL(2) is EXPSPACE-complete. $\square$

Corollary 1: Let $n \geq 2$. The strong satisfiability problem for reactive system specifications described in LTL(n) is EXPSPACE-complete.

5. Discussion

We discuss the complexity of the strong satisfiability problem in relation to that of the satisfiability problem and the realizability problem. The complexity of the satisfiability problem for specifications written in LTL is PSPACE-complete [15], and the complexity of the realizability problem for such specifications is 2EXPTIME-complete [16]. PSPACE is the complexity class of problems solvable in $O(p(n))$ space by a deterministic Turing machine, and 2EXPTIME is the complexity class of problems solvable in $O(2^{2^{p(n)}})$ time by a deterministic Turing machine. The relationship between these classes is as follows:

$$\text{PSPACE} \subseteq \text{EXPSPACE} \subseteq \text{2EXPTIME}$$

Therefore, the strong satisfiability problem is more difficult than the satisfiability problem, and is easier than or of equal difficulty to the realizability problem.

The satisfiability problems for real-time temporal logic TPTL [17], LTL with forgettable past (NLTL) [18], and LTL with freeze quantifier [14] are EXPSPACE-complete. The EXPSPACE-hardness of these problems is due to the fact that constraints between the times i and $i + 2^n$ can be described by formulae of polynomial size with respect to n . For example, the constraint described by the following LTL formula of exponential size can be described by a formula of polynomial size in these logics.

$$\mathbf{G}(\mathbf{X}\mathbf{F}(\neg \text{end} \wedge \bar{c} = 0) \rightarrow \bigvee_{(t,t') \in V} (x_t \wedge \overbrace{\mathbf{X}\mathbf{X} \dots \mathbf{X}}^{2^n \text{ times}} x_{t'}))$$

This corresponds to the constraint for the vertical condition of the EXP-corridor tiling problem in Sect. 3.2. The constraint cannot be described by formulae of polynomial size in pure LTL. In this paper, to prove the EXPSPACE-hardness of strong satisfiability, we showed that the above constraint can be represented by an LTL formula of polynomial size quantified by meta-level $\forall \tilde{b}$.

The complexities of satisfiability for various LTL fragments were presented in [19], and the complexities of realizability for LTL fragments were presented in [20], [21]. One of the results of [19], namely that the satisfiability problem remains PSPACE-complete even when only formulae with a temporal depth of at most 2 are allowed, is strongly related to our work. This result was proved via the same reduction technique used to prove our result. Lemma 2 of this paper demonstrates that the reduction technique can be generalized to the strong satisfiability problem.

6. Conclusion

We showed that the strong satisfiability problem for a specification in LTL is EXPSPACE-complete. This implies that the strong satisfiability problem is more difficult than the satisfiability problem, and is easier than or of equal difficulty to the realizability problem. Moreover, we showed that the strong satisfiability problem remains EXPSPACE-complete even when only formulae with a temporal depth of at most 2 are allowed.

In future work, we will investigate the complexity of stepwise satisfiability and strong stepwise satisfiability, which are properties of reactive system specifications introduced in [4]. Furthermore, we will discuss the complexity of the strong satisfiability problem for other LTL fragments. If we succeed in finding an LTL fragment for which specifications can be verified efficiently, verification of reactive system specifications will become more practical. Such LTL fragments were given in [20], [21] for realizability. We will find another fragment by taking strong satisfiability into account. The results presented in this paper will provide important insights for this future work.

Acknowledgments

This work was supported by a Grant-in-Aid for Scientific Research(C) (24500032).

References

- [1] D. Jackson, "Automating first-order relational logic," Proc. 8th ACM SIGSOFT international symposium on Foundations of software engineering: twenty-first century applications, pp.130–139, 2000.
- [2] M. Abadi, L. Lamport, and P. Wolper, "Realizable and unrealizable specifications of reactive systems," Proc. 16th International Colloquium on Automata, Languages, and Programming, LNCS, vol.372, pp.1–17, 1989.
- [3] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," Proc. 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp.179–190, 1989.
- [4] R. Mori and N. Yonezaki, "Several realizability concepts in reactive objects," Proc. Information Modeling and Knowledge Bases IV: Concepts, Methods and Systems, pp.407–424, 1993.
- [5] R. Mori and N. Yonezaki, "Derivation of the input conditional formula from a reactive system specification in temporal logic," Proc. Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS, vol.863, pp.567–582, 1994.
- [6] S. Hagihara and N. Yonezaki, "Completeness of verification methods for approaching to realizable reactive specifications," Proc. 1st Asian Working Conference on Verified Software AWCVS'06, UNU-IIST, vol.348, pp.242–257, 2006.
- [7] S. Hagihara, Y. Kitamura, M. Shimakawa, and N. Yonezaki, "Extracting environmental constraints to make reactive system specifications realizable," Proc. 16th Asia-Pacific Software Engineering Conference, pp.61–68, 2009.
- [8] N. Yoshiura, "Decision procedures for several properties of reactive system specifications," Proc. Software Security - Theories and Systems, LNCS, vol.3233, pp.154–173, 2004.
- [9] Z. Manna and A. Pnueli, "Axiomatic approach to total correctness of programs," Acta Informatica, vol.3, pp.243–263, 1974.
- [10] A. Pnueli, "The temporal semantics of concurrent programs," Theor. Comput. Sci., vol.13, pp.45–60, 1981.
- [11] H. Tauriainen, "On translating linear temporal logic into alternating and nondeterministic automata," Research Report A83, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland, 2003.
- [12] A.P. Sistla, M.Y. Vardi, and P. Wolper, "The complementation problem for büchi automata with applications to temporal logic," Theor. Comput. Sci., vol.49, no.2–3, pp.217–237, 1987.
- [13] P.V.E. Boas, "The convenience of tilings," Complexity, Logic, and Recursion Theory, Lecture Notes in Pure and Applied Mathematics, vol.187, pp.331–363, 1997.
- [14] S. Demri, R. Lazić, and D. Nowak, "On the freeze quantifier in constraint ltl: Decidability and complexity," Information and Computation, vol.205, no.1, pp.2–24, 2007.
- [15] A.P. Sistla and E.M. Clarke, "The complexity of propositional linear temporal logics," J. ACM, vol.32, no.3, pp.733–749, 1985.
- [16] R. Rosner, Modular Synthesis of Reactive Systmes, Ph.D. thesis, 1992.
- [17] R. Alur and T.A. Henzinger, "A really temporal logic," J. ACM, vol.41, no.1, pp.181–203, 1994.
- [18] F. Laroussinie and N. Markey, "Temporal logic with forgettable past," Proc. Logic in Computer Science, pp.383–392, 2002.
- [19] S. Demri and P. Schnoebelen, "The complexity of propositional linear temporal logics in simple cases," Information and Computation, vol.174, no.1, pp.84–103, 2002.
- [20] R. Alur and S. La Torre, "Deterministic generators and games for LTL fragments," ACM Trans. Comput. Logic, vol.5, no.1, pp.1–25,

2004.

- [21] N. Piterman and A. Pnueli, "Synthesis of reactive(1) designs," Proc. Verification, Model Checking, and Abstract Interpretation, LNCS, vol.3855, pp.364–380, 2006.

Masaya Shimakawa received the B.E. and M.E. degrees in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 2004 and 2006, respectively. His research interests include temporal logic, automata theory and formal verification.

Shigeki Hagihara received the B.E., M.E. and D.E. degrees in computer science from Tokyo Institute of Technology in 1993, 1995 and 2000, respectively. From 2000 to 2001, he was a research associate in Research for the Future Program, Japan Society for the Promotion of Science. Since 2001, he has been an assistant professor in Department of Computer Science, Tokyo Institute of Technology. His research interests include proof methods of non-classical logics such as temporal logics, epistemic logics, etc., and software verification, especially, specification analysis and model-checking for reactive systems, and security protocol analysis. He is a member of Japan Society for Software Science and Technology.

Naoki Yonezaki received B.E., M.E. and D.E. degrees from Tokyo Institute of Technology in 1972, 1974 and 1977, respectively. He has been a Professor of Tokyo Institute of Technology since 1991. He was also a Professor of Japan Advanced Institute of Science and Technology from 1991 to 1995. His research interests are formal approach to construction and analysis of complex systems, including algorithms, verification of software, verification of security and formal analysis of biological systems. Currently he is a dean of the School of Information Science and Engineering, and the Director General of Inter-departmental Organization for Informatics in Tokyo Institute of Technology. He is also the head of the Information Technology Specialist educational program (IT Bauhaus), and was the head of the joint education program for Translational Biomedical Informatics with Tokyo Medical and Dental University. He has been the editor in chief of Computer Software, the main Journal of Japan Society of Software Science and Technology (JSSST) from 2000 to 2004. He also served as a program committee chair of IWTS99 and IWPSE01. He was awarded a fellowship by JSSST in 2008.