Complexity of Checking Strong Satisfiability of Reactive System Specifications

Masaya Shimakawa, Shigeki Hagihara, and Naoki Yonezaki

Department of Computer Science,
Graduate School of Information Science and Engineering,
Tokyo Institute of Technology.
2-12-1-W8-67 Ookayama,
Meguro-ku, Tokyo 152-8552, Japan
{masaya,hagihara,yonezaki}@fmx.cs.titech.ac.jp

Abstract. Many fatal accidents involving safety-critical reactive systems have occurred in unexpected situations, which were not considered during the design and test phases of the systems. To prevent these 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. It is known that the complexity of the realizability problem is 2EXPTIME-complete. On the other hand, 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.

Keywords: Reactive System, Verification of Specification, Complexity, Linear Temporal Logic.

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 fall into 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 the systems. It is therefore important to ensure that a specification does not possess this kind of flaw[6].

V.V. Das and P.M. El-Kafrawy (Eds.): SPIT 2012, LNICST 117, pp. 41–50, 2014.
© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2014

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 [1, 12]. In [12], A. Pnueli and R. Rosner showed that a reactive system can be synthesized from a realizable specification.

On the other hand, in [8, 9], we introduced the concept of strong satisfiability, which is a necessary condition for realizability. Many practical unrealizable specifications are also strongly unsatisfiable[9]. In [5], we presented a method for checking whether or not a specification satisfies strong satisfiability. We also proposed techniques for identifying the flaws in strongly unsatisfiable specifications in [4]. Another approach for checking strong satisfiability was introduced in [17].

However, there has been no discussion of the complexity of the strong satisfiability problem, which is an important consideration, since 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 is EXPSPACE-complete. Since it is known that 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.

The remainder of this paper is organized as follows. In Section2, we introduce the concepts of a reactive system, linear temporal logic(LTL) as a specification language, and strong satisfiability, which is a necessary condition for the realizability of a reactive system specification. In Section3, we show that the strong satisfiability problem for a specification written in LTL is EXPSPACE-complete. In Section4, we discuss the complexity of the strong satisfiability problem in relation to that of the satisfiability problem and the realizability problem. We present our conclusions in Section5.

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.

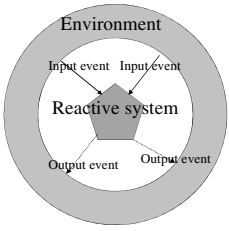

Fig. 1. A reactive system

Definition 1 (Reactive System). *A reactive system RS is a 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 reactive systems. Modal logics are widely used in computer science. Among these, temporal logics have often been applied to the analysis of reactive systems, following the application of such logics to program semantics by Z. Manna and A. Pnueli[7]. A propositional linear temporal logic (LTL)[11] 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.

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 at the next time,' while $f\mathbf{U}g$ means that '*f* always holds until *g* holds.' The notations $f \vee g$, $f \rightarrow g$, $f \leftrightarrow g$, $f \oplus g$, $f\mathbf{R}g$, $\mathbf{F}f$, and $\mathbf{G}f$ are abbreviations for $\neg(\neg f \wedge \neg g)$, $\neg(f \wedge \neg g)$, $\neg(f \wedge \neg g) \wedge \neg(\neg f \wedge g)$, $\neg(f \leftrightarrow g)$, $\neg(\neg f\mathbf{U}\neg g)$, and $(\neg\bot)\mathbf{U}f$, $\neg\mathbf{F}\neg f$ respectively, where $\bot$ is an atomic proposition representing 'falsity.'

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 $\sigma[i]$. When a formula *f* holds on the *i*-th set of a behavior σ, we write $\sigma, i \models f$, and inductively define this relation as follows:

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

We say that σ satisfies *f* and write $\sigma \models f$ if $\sigma, 0 \models f$. 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:*

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

where $\tilde{\imath}$ *is an infinite sequence of sets of input events; i.e.,* $\tilde{\imath} \in (2^X)^\omega$. $behave_{RS}(\tilde{\imath})$ is *the infinite behavior of* $\tilde{\imath}$ *caused by RS, defined as follows. If* $\tilde{\imath} = i_0 i_1 \ldots$,

$$behave_{RS}(\tilde{\imath}) = (i_0 \cup o_0)(i_1 \cup o_1) \ldots,$$

where o_i *is a set of output events caused by RS; i.e.,* $o_i = r(i_0 \ldots i_i)$, *and* $\cup$ *denotes the union of two sets.*

The following property was shown to be a necessary condition for realizability in [8].

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

$$\forall \tilde{\imath} \exists \tilde{o}(\langle \tilde{\imath}, \tilde{o} \rangle \models Spec),$$

where $\tilde{o}$ *is an infinite sequence of sets of output events; i.e.,* $\tilde{o} \in (2^Y)^\omega$. *If* $\tilde{\imath} = i_0 i_1 \ldots$ *and* $\tilde{o} = o_0 o_1 \ldots$, *then* $\langle \tilde{\imath}, \tilde{o} \rangle$ *is defined by* $\langle \tilde{\imath}, \tilde{o} \rangle = (i_0 \cup o_0)(i_1 \cup o_1) \ldots$.

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 control system for a door. 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 $\tilde{\imath} = \{x_1, x_2\}\{x_2\}\{x_2\}...$, $\exists\tilde{o}(\langle\tilde{\imath}, \tilde{o}\rangle \models Spec_1)$ does not hold. Hence $\forall\tilde{\imath}\exists\tilde{o}(\langle\tilde{\imath}, \tilde{o}\rangle \models Spec_1)$ 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 $O(2^{p(n)})$ amount of space by a deterministic Turing machine, where $p(n)$ is a polynomial function), 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 demonstrate a procedure for checking strong satisfiability which uses $O(2^{p(n)})$ amount of space. This procedure is a modified version of the technique introduced in [5].

A *non-deterministic Büchi automaton* is a tuple $A=\langle\Sigma, Q, q_0, \delta, F\rangle$, 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 A on an ω-word $\alpha=\alpha[0]\alpha[1]\ldots$ is an infinite sequence $\gamma = \gamma[0]\gamma[1]\ldots$ 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 A accepts α, if there is a run γ on α such that $In(\gamma)\cap F\neq\emptyset$ holds, where $In(\gamma)$ is the set of states that occur infinitely often in γ. The set of ω-words accepted by A is called the language accepted by A, and is denoted by $L(A)$.

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 $A = \langle 2^{X\cup Y}, Q, q_0, \delta, F\rangle$ such that $L(A) = \{\sigma \mid \sigma \models Spec\}$ holds.
2. Let $A' = \langle 2^{X}, Q, q_0, \delta', F\rangle$ be a non-deterministic Büchi automaton obtained by restricting A to only input events, where $\delta' = \{(q, i, q') \mid \exists o\ (q, i\cup o, q')\in \delta\}$. Note that $L(A') = \{\tilde{\imath} \mid \exists\tilde{o}\ \langle\tilde{\imath}, \tilde{o}\rangle\in L(A)\}$ holds due to the definition of δ'.

3. We check whether or not A' is universally acceptable (which means that $L(A') = (2^X)^\omega$). 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.

A can be constructed within $O(2^{|Spec|})$ amount of space, and the size of A is also $O(2^{|Spec|})$[16]. Since A' is obtained by projection, A' can be constructed within $O(|A|)$ amount of space, and the size of A' is O($|A|$). The universality problem for a Büchi automaton is in PSPACE[15], and Step 3 is accomplished within $O(p(|A'|))$ amount of space. Therefore, we can check strong satisfiability in $O(2^{|Spec|})$ amount of 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 section, we show that the strong satisfiability problem is EXPSPACE-hard, by providing polynomial time reduction from the EXP-corridor tiling problem[3] to the strong satisfiability problem. It is well known that 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 tiles,* $H, V \subseteq T \times T$ *are horizontal and vertical adjacency constraints,* $t_{init}, t_{final} \in T$ *are the initial and final tiles, and* $m \in N$*, determine whether or not there exists* $k \in N$*, and an assignment function* $f : [0,\ldots,(2^m - 1)] \times [0, k] \rightarrow T$*, such that the following conditions are satisfied:*

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

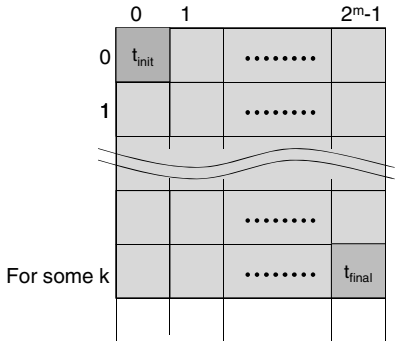

Fig. 2. The EXP-corridor tiling problem

As Fig. 2 shows, the tiling grid has $2^m \times \omega$ points. Intuitively, this problem asks: "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 $0 \leq j \leq k$, satisfying the conditions 1-4?" The condition 1 is the condition for the initial tile, and states that the initial tile t_{init} is assigned to the leftmost and topmost point. The condition 2 is the condition for the final tile, and states that the final tile t_{final} is assigned to the rightmost and bottommost point. The condition 3 is the condition for horizontal lines, and states that each tile and the tile to its right satisfy the horizontal adjacency constraint H. The condition 4 is the condition for vertical lines, and states that each tile and the tile beneath it satisfy the vertical adjacency constraint V.

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 $\exists\tilde{\imath}\forall\tilde{o}(\langle\tilde{\imath}, \tilde{o}\rangle \models \neg\varphi_{tiling})$ 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 related to "the corresponding infinite sequence of sets of input events does not satisfy φ_{tiling} for any infinite sequence of sets of output events."

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 t is placed on the point (i, j)" is related to "the input events x_t occur at the time $i + (2^m) \cdot j$."
- *end*: "tiling assignment concludes at the point (i, j)" is related to "*end* occurs at the time $i + (2^m) \cdot j$."
- $c_0, \ldots, c_{m-1}$: These are m bit counters that count the amount of time. By checking these counters, we can identify a column of the tiling grid.

Output events. We introduce the following output events.

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

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

$$\dot{c} = 0 \equiv \bigwedge_{0 \leq i < m} \neg c_i$$

$$\dot{c} = 2^m - 1 \equiv \bigwedge_{0 \leq i < m} c_i$$

$$\dot{c} = \dot{y} \equiv \bigwedge_{0 \leq i < m} (c_i \leftrightarrow y_i)$$

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

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

This represents the statement "the value of $\dot{c}$ is 0 initially, and is incremented on every pass," which means that "$\dot{c}$ is a counter."

– The relation between a tile and a point of the grid.

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

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

– The constraint for the condition 1.

$$x_{t_{init}} \tag{3}$$

This represents the statement "the initial tile t_{init} is placed on the point (0, 0)."

– The constraint for the condition 2.

$$\neg end \mathbf{U}(\neg end \wedge \dot{c} = 2^m - 1 \wedge x_{t_{final}} \wedge \mathbf{XG}end) \tag{4}$$

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

– The constraint for the condition 3.

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

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 to the right satisfy the condition H."

– The constraint for the condition 4.

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

This represents the statement "if the value of y is never changed, for any current point in the column indicated by y, if tiling is not finished at the point just below the current point, the tile at the current point and the tile beneath it satisfy the condition V." Here "the tile at the current point and the tile beneath it satisfy the condition V" is specified by "$(t, t') \in V$ such that t is placed on the current point and t' is placed on the point whose column follows that of the current point by y." Hence ($\forall \tilde{y}(\ldots \models$ (6)) represents the statement "for any column, tiles in the column satisfy the condition V," which means "any tiles satisfy the 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_{init}, t_{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. Since 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. Since EXPSPACE=co-EXPSPACE, the strong satisfiability problem is also EXPSPACE-hard.

4 Discussion

In this section, we discuss the complexity of the strong satisfiability problem in relation to that of the satisfiability problem and the realizability problem. It is well known that the complexity of the satisfiability problem for specifications written in LTL is PSPACE-complete[14], and the complexity of the realizability problem for specifications written in LTL is 2EXPTIME-complete[13]. PSPACE is the complexity class of problems solvable in $O(p(n))$ amount of space by a deterministic Turing machine, and 2EXPTIME is the complexity class of problems solvable in $O(2^{(2^{(p(n))})})$ amount of time by a deterministic Turing machine. The relationship between these classes is as follows:

$$\mathrm{PSPACE} \subsetneq \mathrm{EXPSPACE} \subseteq \mathrm{2EXPTIME}$$

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

5 Conclusion

In this paper, we showed that the strong satisfiability problem is EXPSPACE-complete. This indicates that the strong satisfiability problem is more difficult than the satisfiability problem, and is easier than or of equal difficulty to the realizability problem.

In future work, we will investigate the complexity of stepwise satisfiability and strong stepwise satisfiability, which are properties of reactive system specifications that were introduced in [8]. Furthermore, we will discuss the complexity of the strong satisfiability problem for subsystems of LTL that are syntactically restricted. If we succeed in finding a subsystem for which specifications can be verified efficiently, verification of reactive system specifications will become more practical. For realizability, subsystems of LTL were given in [2, 10]. We will find another subsystem by taking strong satisfiability into account. The results presented in this paper will provide important guidelines for this future work.

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

References

1. Abadi, M., Lamport, L., Wolper, P.: Realizable and unrealizable specifications of reactive systems. In: Ronchi Della Rocca, S., Ausiello, G., Dezani-Ciancaglini, M. (eds.) ICALP 1989. LNCS, vol. 372, pp. 1–17. Springer, Heidelberg (1989)
2. Alur, R., La Torre, S.: Deterministic generators and games for ltl fragments. ACM Trans. Comput. Logic 5(1), 1–25 (2004)
3. Boas, P.V.E.: The convenience of tilings. In: Complexity, Logic, and Recursion Theory, pp. 331–363. Marcel Dekker Inc. (1997)
4. Hagihara, S., Kitamura, Y., Shimakawa, M., Yonezaki, N.: Extracting environmental constraints to make reactive system specifications realizable. In: Proc. of the 16th Asia-Pacific Software Engineering Conference, pp. 61–68. IEEE (2009)
5. Hagihara, S., Yonezaki, N.: Completeness of verification methods for approaching to realizable reactive specifications. In: Proc. of 1st Asian Working Conference on Verified Software. UNU-IIST Technical Report, vol. 348, pp. 242–257 (2006)
6. Jackson, D.: Automating first-order relational logic. In: Proceedings of the 8th ACM SIGSOFT International Symposium on Foundations of Software Engineering: Twenty-First Century Applications, SIGSOFT 2000/FSE-8, pp. 130–139. ACM (2000)
7. Manna, Z., Pnueli, A.: Axiomatic approach to total correctness of programs. Acta Informatica 3(3), 243–263 (1974)
8. Mori, R., Yonezaki, N.: Several realizability concepts in reactive objects. In: Information Modeling and Knowledge Bases (1993)
9. Mori, R., Yonezaki, N.: Derivation of the input conditional formula from a reactive system specification in temporal logic. In: Langmaack, H., de Roever, W.-P., Vytopil, J. (eds.) FTRTFT 1994 and ProCoS 1994. LNCS, vol. 863, pp. 567–582. Springer, Heidelberg (1994)
10. Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. In: Emerson, E.A., Namjoshi, K.S. (eds.) VMCAI 2006. LNCS, vol. 3855, pp. 364–380. Springer, Heidelberg (2006)
11. Pnueli, A.: The temporal semantics of concurrent programs. Theoretical Computer Science 13, 45–60 (1981)
12. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 179–190 (1989)
13. Rosner, R.: Modular Synthesis of Reactive Systmes. Ph.D. thesis, Weizmann Institute of Science (1992)
14. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. J. ACM 32(3), 733–749 (1985)
15. Sistla, A.P., Vardi, M.Y., Wolper, P.: The complementation problem for Büchi automata with applications to temporal logic. Theor. Comput. Sci. 49(2-3), 217–237 (1987)
16. Tauriainen, H.: 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)
17. Yoshiura, N.: Decision procedures for several properties of reactive system specifications. In: Futatsugi, K., Mizoguchi, F., Yonezaki, N. (eds.) ISSS 2003. LNCS, vol. 3233, pp. 154–173. Springer, Heidelberg (2004)