Information Sciences 377 (2017) 51–74

Contents lists available at ScienceDirect

Information Sciences

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

Model checking of linear-time properties in multi-valued systems[☆]

Yongming Li[a,*], Manfred Droste[b], Lihui Lei[a]

[a] College of Computer Science, Shaanxi Normal University, Xi'an, 710062, China
[b] Institute of Computer Science, Leipzig University, D-04109 Leipzig, Germany

ARTICLE INFO

Article history:
Received 27 March 2016
Revised 14 October 2016
Accepted 19 October 2016
Available online 19 October 2016

Keywords:
Model checking
Multi-valued transition system
Invariant
Safety
Liveness
Lattice-valued finite automaton

ABSTRACT

In this paper, we study the model-checking problem of linear-time properties in multi-valued systems. Safety properties, invariant properties, liveness properties, persistence and dual-persistence properties in multi-valued logic systems are introduced. Some algorithms related to the above multi-valued linear-time properties are discussed. The verification of multi-valued regular safety properties and multi-valued ω-regular properties using lattice-valued automata are thoroughly studied. Since the law of non-contradiction (i.e., $a \wedge \neg a = 0$) and the law of excluded-middle (i.e., $a \vee \neg a = 1$) do not hold in multi-valued logic, the linear-time properties introduced in this paper have new forms compared to those in classical logic. Compared to those classical model-checking methods, our methods to multi-valued model checking are accordingly more direct: We give an algorithm for showing $TS \models P$ for a model TS and a linear-time property P, which proceeds by directly checking the inclusion $Traces(TS) \subseteq P$ instead of $Traces(TS) \cap \neg P = \emptyset$. A new form of multi-valued model checking with membership degree is also introduced. In particular, we show that multi-valued model checking can be reduced to classical model checking. The related verification algorithms are also presented. Some illustrative examples and a case study are also provided.

1. Introduction

In the last four decades, computer scientists have systematically developed theories of correctness and safety as well as methodologies, techniques and even automatic tools for correctness and safety verification of computer systems; see for example [1,33–34,43,47]. Of which, model checking has become established as one of the most effective automated techniques for analyzing correctness of software and hardware designs. A model checker checks a finite-state system against a correctness property expressed in a propositional temporal logic such as LTL (Linear Temporal Logic) or CTL (Computational Tree Logic). These logics can express safety (e.g., "No two processes can be in the critical section at the same time") and liveness (e.g., "Every job sent to the printer will eventually print") properties. Model checking has been effectively applied to reasoning about correctness of hardware, communication protocols, software requirements, etc. Many industrial model checkers have been developed, including SPIN [25], SMV [44].

☆ This work is supported by National Science Foundation of China (grant No: 11271237,61228305,11671244) and the Higher School Doctoral Subject Foundation of Ministry of Education of China (grant No:200807180005).

* Corresponding Author. Fax: 86 29 85310118.
E-mail addresses: liyongm@snnu.edu.cn (Y. Li), droste@informatik.uni-leipzig.de (M. Droste), leilihui@snnu.edu.cn (L. Lei).

http://dx.doi.org/10.1016/j.ins.2016.10.030
0020-0255/© 2016 Elsevier Inc. All rights reserved.

Despite their variety, existing model checkers are typically limited to reasoning in classical logic. However, there are a number of problems for which classical logic is insufficient. One of these is reasoning under uncertainty. This can occur either when complete information is not known or cannot be obtained (e.g., during 'requirements' analysis), or when this information has been removed (abstraction). Classical model checkers typically deal with uncertainty by creating extra states, one for each value of the unknown variable and each feasible combination of values of known variables. However, this approach adds significant extra complexity to the analysis. Classical reasoning is also insufficient for models that contain inconsistencies. Models may be inconsistent because they combine conflicting points of view, or because they contain components developed by different people. Conventional reasoning systems cannot cope with inconsistency because the presence of a single contradiction results in trivialization – anything follows from $A \wedge \neg A$. Hence, faced with an inconsistent description and the need to perform automated reasoning, we must either discard information until consistency is achieved again, or adopt a nonclassical logic. Multi-valued logic (mv-logic, in short) provides a solution to both reasoning under uncertainty and under inconsistency. For example, we can use unknown and no agreement as logic values. In fact, model checkers based on three-valued and four-valued logics have already been studied. For example, [8] (c.f., [46]) used a three-valued logic for interpreting results of model-checking with abstract interpretation, whereas [24] used four-valued logics for reasoning about abstractions of detailed gate or switch-level designs of circuits. For reasoning about dynamic properties of systems, we need to extend existing modal logics to the multi-valued case. Fitting [20] explores two different approaches for doing this: the first extends the interpretation of atomic formulae in each world to be multi-valued; the second also allows multi-valued accessibility relations between worlds. The latter approach is more general, and can readily be applied to the temporal logics used in model checking [12]. We use different multi-valued logics to support different types of analysis. For example, to model information from multiple sources, we may wish to keep track of the origin of each piece of information, or just the majority vote, etc. Thus, rather than restricting ourselves to any particular multi-valued logic, our approach is to extend classical symbolic model checking to arbitrary multi-valued logics, as long as conjunction, disjunction and negation of the logical values are well defined. M. Chechik and her colleagues have published a series of papers along this line, see [8–10,12,13].

Our purpose is to develop automata-based model-checking techniques in the multi-valued setting. More precisely, the major design decision of this paper is as follows:

A lattice-valued automaton is adopted as the model of the systems. This is reasonable since classical automata (or equivalent transition systems) are common system models in classical model checking. Linear-time properties of multi-valued systems are checked in this paper. They are defined to be infinite sequences of sets of atomic propositions, as in the classical case, with truth-values in a given lattice. The key idea of the automata-based approach to model checking is that we can use an auxiliary automaton to recognize the properties to be checked, and then combine it with the system to be checked so that the problem of checking the safety or ω-properties of the system is reduced to checking some simpler (invariance or persistence) properties of the larger system composed by the systems under checking and the auxiliary automaton. A difference between the classical case and the multi-valued case deserves a careful explanation. Since the law of non-contradiction (i.e., $a \wedge \neg a = 0$) and the law of excluded middle (i.e., $a \vee \neg a = 1$) do not hold in multi-valued logic, the present forms of many classical properties in multi-valued logic must have some new forms, and some distinct constructions need to be given in multi-valued logic.

As said in Ref. [2], the equivalences and preorders between transition systems that "correspond" to linear temporal logic are based on trace inclusion and equality, whereas for branching temporal logic such relations are based on simulation and bisimulation relations. That is to say, the model checking of a transition system TS which represents the model of a system satisfying a linear temporal formula φ, i.e., $TS \models \varphi$ is equivalent to checking the inclusion relation $Traces(TS) \subseteq P$, where $Traces(TS)$ is the trace function of the transition system TS and P is the temporal property representing the formula φ. In classical logic, we know that $a \leq b$ if and only if $a \wedge \neg b = 0$ holds. Therefore, $TS \models \varphi$ if and only if $Traces(TS) \cap \neg P = \emptyset$. Then, instead of checking $TS \models \varphi$ directly using the inclusion relation $Traces(TS) \subseteq P$, it is equivalent to checking the emptiness of the language $L(\mathcal{A}) \cap L(\mathcal{A}_{\neg\varphi})$ indirectly, where $\mathcal{A}$ is a Büchi automaton representing the trace function of the transition system TS (i.e., $L(\mathcal{A}) = Traces(TS)$), and $\mathcal{A}_{\neg\varphi}$ is a Büchi automaton related to temporal property $\neg\varphi$ (i.e., $L(\mathcal{A}_{\neg\varphi}) = \neg P$).

In contrast, in mv-logic, $a \leq b$ is in general not equivalent to the condition $a \wedge \neg b = 0$, so the classical method to solve model checking of linear-time properties does not universally apply to the multi-valued model checking. The available methods of multi-valued model checking ([9]) still used the classical method with some minor correction. That is, instead of checking of $TS \models P$ for a multi-valued linear time property P using the inclusion of the trace function $Traces(TS) \subseteq P$, the available method only checked the membership degree of the language $L(\mathcal{A}) \cap L(\mathcal{A}_{\neg P})$, where $\mathcal{A}_{\neg P}$ is a multi-valued Büchi automaton such that $L(\mathcal{A}_{\neg P}) = \neg P$. As we know, these two methods are not equivalent in mv-logic. Then, some new methods to apply multi-valued model checking of linear-time properties based on trace inclusion relations need to be developed.

We provide new results along this line. In fact, we shall give a method of multi-valued model checking of linear-time property directly using the inclusion of the trace function of TS into a linear-time property P. In propositional logic, we know that we can use the implication connective $\rightarrow$ to represent the inclusion relation. In fact, in classical logic, we know that the implication connective can be represented by disjunction and negation connectives, that is, $a \rightarrow b = \neg a \vee b$. In this case, we know that $a \leq b$ if and only if $\neg a \vee b = 1$, if and only if $a \wedge \neg b = 0$, if and only if $a \rightarrow b = 1$. Then a natural problem arises: how to define the implication connective in multi-valued logic? By the above analysis, it is not appropriate to use the implication connective defined in the form $a \rightarrow b = \neg a \vee b$ to represent the inclusion relation in multi-valued logic. In order to use the implication connective to reflect the inclusion relation in mv-logic, we shall use implication connective