Theoretical Computer Science 554 (2014) 229–253

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# A formal proof of the deadline driven scheduler in PPTL axiomatic system ☆

Nan Zhang [a], Zhenhua Duan [a,*], Cong Tian [a], Dingzhu Du [b]

[a] Institute of Computing Theory and Technology and ISN Lab, Xidian University, Xi'an, 710071, China
[b] University of Texas at Dallas, Richardson, TX 75080, USA

ARTICLE INFO

Available online 21 December 2013

*Keywords:*
Theorem proving
Projection temporal logic
Deadline driven scheduler
Real-time

ABSTRACT

This paper presents an approach for verifying the correctness of the feasibility theorem on the deadline driven scheduler (DDS) with the axiom system of Propositional Projection Temporal Logic (PPTL). To do so, the deadline driven scheduling algorithm is modeled by an MSVL (Modeling, Simulation and Verification Language) program and the feasibility theorem is formulated by PPTL formulas with two parts: a necessary part and a sufficient part. Then, several lemmas are abstracted and proved by means of the axiom system of PPTL. With the help of the lemmas, two parts of the theorem are deduced respectively. This case study convinces us that some real-time properties of systems can be formally verified by theorem proving using the axiom system of PPTL.



## 1. Introduction

Concurrent programs are notorious for hidden bugs. Testing and simulation are traditional methods for finding bugs in the programs, but these methods cannot provide full guarantee of correctness of programs since they can hardly cover all possible situations. On the other hand, formal verification techniques are valuable approaches for verifying the correctness and reliability of concurrent programs. Generally speaking, there are two typical approaches for formal verification: model checking [1,2] and theorem proving [3]. With model checking, the system under consideration is modeled as a model $M$ in terms of automata or transition systems etc. while the property to be verified is defined as a formula $P$ in a temporal logic. The property $P$ is verified over the model $M$ through exhaustive enumeration of all the states reachable by the model and the behaviors that traverse through them. A number of model checkers such as SPIN [4] and SMV [5] have been developed based on LTL [6,7] and CTL [1,8] and used with success. However, model checking suffers from state explosion problem and can be only used for checking finite state systems. In contrast, theorem proving can be used to verify finite and infinite state systems. To do this, the system is modeled as a formula $S$ with a theory, and the property is defined as a formula $P$ in a logic. Then, a proof procedure is employed to construct a proof for $S \rightarrow P$ in the axiom system. Several successful theorem provers such as PVS [9], ACL2 [10], and Coq [11,12] have also been developed. However, theorem proving techniques suffer from the following: (1) theorem provers are not automatic tools most of which rely on intelligence of human being; (2) in most of cases, model $S$ and property $P$ are written in different notations and the proof system is in another notation; this multi-notation problem makes the proof process complicated; (3) within the temporal logic community, LTL and CTL are popular, the often verified properties are safety and liveness since the two logics are not full regular. Hence, some

☆ This research is supported by the National Program on Key Basic Research Project of China (973 Program) Grant No. 2010CB328102, National Natural Science Foundation of China under Grant Nos. 61133001, 61202038, 61272117, 61272118, 61373043 and 61322202, the Fundamental Research Funds for the Central Universities Grant No. K5051303022.

* Corresponding author.
*E-mail addresses:* nanzhang@xidian.edu.cn (N. Zhang), zhhduan@mail.xidian.edu.cn (Z. Duan), ctian@xidian.edu.cn (C. Tian), dzdu@utdallas.edu (D. Du).

0304-3975/$ – see front matter 
http://dx.doi.org/10.1016/j.tcs.2013.12.014

properties such as Kleene closure properties and time duration properties cannot be defined with them [13]. To overcome these shortcomings, Projection Temporal Logic (PTL) [14–16] has been formalized based on Interval Temporal Logic (ITL) [17]. It extends ITL with a new projection (*prj*) operator, a previous ($\ominus$) operator, a cylinder computation construct based on projection and full regular sequence expressions [18], as well as infinite models. Further, a Propositional PTL (PPTL) has also been proposed and an axiom system for PPTL has been formalized [19]. As a result, the expressiveness of PPTL is full omega regular [13] and the axiom system is sound and complete [19]. Moreover, an executable subset of PTL called Modeling, Simulation and Verification Language (MSVL) has been developed [15,16]. MSVL is a temporal logic programming language consists of sequential, branching, and iteration statements as well as parallel, cylinder computation, framing, synchronous and asynchronous communication statements. A supporting tool MSV for MSVL allows us to model a system in terms of MSVL programs, to simulate a system by executing of a path of the model, and to verify properties of the system by means of a unified model checking [20] and theorem proving approaches. To do the theorem proving, a system can be modeled by an MSVL program $S$, and a property to be verified can be defined by a PPTL formula $P$. Since a finite-state program can be viewed as a finite propositional Kripke structure and it can be specified using propositional temporal logic, accordingly, a finite-state program $S$ can be treated as a PPTL formula [21]. Therefore, to prove whether the program $S$ satisfies the property $P$ is equivalent to proving $S \rightarrow P$ is a theorem in the PPTL axiom system. This paper presents a theorem proving approach for proving the correctness of the feasibility theorem on DDS based on MSVL programs and the PPTL axiom system.

Deadline Driven Scheduler (DDS) of Liu and Layland [22] was proposed in 1973. The algorithm is about scheduling a finite number of tasks on a single processor, assuming that (1) all tasks raise requests for processor time periodically; (2) each task requests a constant amount of processor time in its period; (3) the deadline of a request for a task is defined to be the time of the next request for the same task; (4) the tasks are independent of each other which means that the raising of requests of one task doesn't depend on the raising or completion of requests of other tasks. The DDS algorithm is devised to assign the task the highest priority whose deadline of the current request is the most urgency, and guarantee that at any instant, only one of the processes with the highest priority and an unfulfilled request can be selected to occupy or even preempt the processor.

In their paper, they also investigated a feasibility condition for DDS, which was a necessary and sufficient condition for the feasibility of DDS. The condition, called Theorem 1 given in Section 3, determined whether or not the set of tasks would meet their timing requirements. It is hard to prove the correctness of the theorem. Liu and Layland [22] only gave an informal proof demonstrated in two theorems and adopted a contradiction approach for the proof. Later on, Zheng and Zhou [23] presented a formal proof with the Duration Calculus (DC) proof system [24]. DC is an extension of real arithmetic and ITL [17]. It provides a possible way to introduce notions of real analysis into formal techniques for designing embedded real time systems. Since the processor may be preempted dynamically in DDS, DC provides a useful notation, state duration $\int Run_i$, to abstract the random preemption of the processor in a specified interval, where $Run_i$ is a function from time to boolean values $\{0, 1\}$ and $\int Run_i$ a function from intervals to real values, presenting the accumulated presence time of the state expression $Run_i$ in the interval. This is the first formal proof for DDS in the literature. However, it applied heavily the induction rules IR1 and IR2 which made the proof hard to follow. In 2001, Zhan [25] proposed another formal proof given likewise in terms of DC. The proof was to follow and to formalize the original one developed by Liu and Layland [22] which relied on many intuitive facts. Therefore it was more intuitive, while it was still formal. In 2008, Xu etc. [26] produced a new proof with DC, which not only improved Zhan's work in style but also considerably in contents.

All of the above proofs are based on the first order logic. To exemplify the flexibility of MSVL and illustrate the axiom system of PPTL can manage a nontrivial proof, we are motivated to prove the correctness of DDS with PPTL proof system in this paper. Our contribution is three-fold. First, we model the DDS algorithm in terms of a MSVL program. To the best of our knowledge, there does not exist any work in the literature modeling the DDS algorithm as an executable logical program so far. Second, the feasibility theorem on DDS is formalized by PPTL formulas in two parts: the necessary part and the sufficient part. Third, two parts of the theorem are respectively verified by means of theorem proving in the axiom system of PPTL given in [19], which convinces us the correctness of the feasibility theorem. To pave the way for the final conclusion, fifteen lemmas are proved in advance. These proofs rely on many intuitive facts so they are more intelligible. This case study also convinces us that some real-time properties of systems can be formally specified and verified in the framework of PPTL.

The rest of the paper is organized as follows. In the following section, we introduce the preliminaries for formal description and proof of deadline driven scheduler, including PTL, MSVL, PPTL and its proof system. In Section 3, the algorithm is formalized by an MSVL program. Then the proof of Liu and Layland's theorem is given in Section 4 including the proofs of the sufficiency and necessity. Finally, the conclusion is drawn in Section 5.

## 2. Preliminaries

### 2.1. Projection temporal logic

Our underlying logic is Projection Temporal Logic (PTL) [15,16]. In the following, we briefly introduce its syntax and semantics.