Towards SMT-Based LTL Model Checking of Clock Constraint Specification Language for Real-Time and Embedded Systems *

Min Zhang Yunhui Ying †

Shanghai Key Lab of Trustworthy Computing, ECNU, Shanghai, China
MoE International Joint Lab of Trustworthy Software, ECNU, Shanghai, China
zhangmin@sei.ecnu.edu.cn, ying_yunhui@163.com

Abstract

The Clock Constraint Specification Language (CCSL) is a formal language companion to MARTE (shorthand for Modeling and Analysis of Real-Time and Embedded systems), a UML profile used to facilitate the design and analysis of real-time and embedded systems. CCSL is proposed to specify constraints on the occurrences of events in systems. However, the language lacks efficient verification support to formally analyze temporal properties, which are important properties to real-time and embedded systems. In this paper, we propose an SMT-based approach to model checking of the temporal properties specified in Linear Temporal Logic (LTL) for CCSL by transforming CCSL constraints and LTL formulas into SMT formulas. We implement a prototype tool for the proposed approach and use the state-of-the-art tool Z3 as its underlying SMT solver. We model two practical real-time and embedded systems, i.e., a traffic light controller and a power window system in CCSL, and model check LTL properties of them using the proposed approach. Experimental results demonstrate the effectiveness and efficiency of our approach.

CCS Concepts • **Computer systems organization → Real-time system specification**; • **Software and its engineering → Model checking**

Keywords Model checking, SMT, Z3, CCSL, LTL

1. Introduction

Events play an important role in real-time and embedded systems for the communication and synchronization of tasks. Events are closely related. Constraints among them need to be carefully defined to avoid any conflict, deadlock and inconsistency. To describe such constraints, the Clock Constraint Specification Language (CCSL) [1] is originally proposed as a companion language of MARTE [17, 19], a UML profile for Modeling and Analysis of Real-Time and Embedded systems. The role that CCSL plays in MARTE is to formalize constraints on events. It focuses on constraints independently from any other assumptions on the global behavior such as periodic tasks, harmonic executions and preemptive communications, which allows quick prototyping of models of target systems [16] and verification of the conformance of implementations with the specification [2]. Events in systems are represented as logical clocks in CCSL. Logical clocks give a flexible abstraction to compare and order the occurrence of events. Systems are modeled in a incremental way by the conjunction of constraints. CCSL has evolved and been developed independently of MARTE. Applications of CCSL to some industrial systems such as vehicle systems [7, 14] and Cyber-Physical Systems [15] demonstrate its effectiveness to the design of real-time and embedded systems.

As an emerging modeling language, CCSL is attracting more and more researchers' interest in investigating effective and efficient approaches to the formal analysis of CCSL constraints, such as schedulability and deadlock detection, by either verification or simulation approaches [2, 12]. A recent survey [16] describes many research efforts in this direction. Approaches are proposed to encode CCSL constraints using existing formalisms such as transition systems [16], Promela [20], Büchi automata [21], timed automata [18], Rewriting Logic [22]. However, these existing encoding approaches have one or more drawbacks as listed below:

- Some CCSL constraints cannot be represented with finite transition systems, and hence only a subset of CCSL constraints are considered for transformation such as [2, 18, 20, 21].
- Some encoding approaches such as [16, 18, 20, 21] need nontrivial intermediate transformation which is either prone to unexpected complexity in transformation.
- State-based encoding approaches may lead to drastic increase of state space and become inefficient for verification [22].

In this paper, we propose an SMT-based (Satisfiability Modulo Theories [5]) approach to LTL (Linear Temporal Logic) model checking of CCSL. In the approach, both CCSL constraints and LTL properties are encoded as SMT formulas. Model checking of LTL properties is transformed into the satisfiability problem of generated SMT formulas, which can be solved by existing SMT solvers, e.g. Z3 [11] or CVC4 [4]. Compared with aforementioned approaches, the SMT-based approach has the following advantages:

1. It takes advantage of the efficiency of the state-of-the-art SMT solvers Z3 in solving the satisfiability problem to achieve efficient model checking of CCSL;
2. The transformation from CCSL constraints into SMT formulas isin a modular way and is applicable to all kinds of CCSL constraints.

* This material is based upon work supported by National Natural Science Foundation of China (NSFC) project: No. 61502171.

† Corresponding author.

LCTES'17, June 21–22, 2017, Barcelona, Spain
© 2017 ACM. 978-1-4503-5030-3/17/06...$15.00
http://dx.doi.org/10.1145/3078633.3081035

Table 1. Syntax and semantics of CCSL

$\delta \models c_1\ [b] \prec c_2$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.\chi(c_2,n) - \chi(c_1,n) = b \implies c_2 \notin \delta(n)$	(Precedence)
$\delta \models c_1 \preccurlyeq c_2$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.\chi(c_1,n) \geq \chi(c_2,n)$	(Causality)
$\delta \models c_1 \subseteq c_2$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.c_1 \in \delta(n) \Rightarrow c_2 \in \delta(n)$	(Subclock)
$\delta \models c_1 \# c_2$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.c_1 \notin \delta(n) \vee c_2 \notin \delta(n)$	(Exclusion)
$\delta \models c_1 \triangleq c_2 + c_3$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.(c_1 \in \delta(n) \iff c_2 \in \delta(n) \vee c_3 \in \delta(n))$	(Union)
$\delta \models c_1 \triangleq c_2 * c_3$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.(c_1 \in \delta(n) \iff c_2 \in \delta(n) \wedge c_3 \in \delta(n))$	(Intersection)
$\delta \models c_1 \triangleq c_2 \wedge c_3$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.\chi(c_1,n) = max(\chi(c_2,n),\chi(c_3,n))$	(Infimum)
$\delta \models c_1 \triangleq c_2 \vee c_3$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.\chi(c_1,n) = min(\chi(c_2,n),\chi(c_3,n))$	(Supremum)
$\delta \models c_1 \triangleq c_2\ \$\ d$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.\chi(c_1,n) = max(\chi(c_2,n) - d, 0)$	(Delay)
$\delta \models c_1 \triangleq c_2\ \$\ d\ on\ c_3$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.c_1 \in \delta(n) \iff (c_3 \in \delta(n) \wedge \exists m \in \mathbb{N}^+.(c_2 \in \delta(m) \wedge \chi(c_3,n) - \chi(c_3,m) = d))$	(DelayFor)
$\delta \models c_1 \triangleq c_2 \propto p$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.c_1 \in \delta(n) \iff (c_2 \in \delta(n) \wedge \exists m \in \mathbb{N}^+.\chi(c_2,n) = m \times p - 1)$	(Periodicity)
$\delta \models c_1 \triangleq c_2 \lightning c_3$	$\Longleftrightarrow$	$\forall n \in \mathbb{N}^+.c_1 \in \delta(n) \iff (c_3 \in \delta(n) \wedge (\exists m \in \mathbb{N}^+.c_3 \in \delta(m) \wedge \chi(c_3,n) - \chi(c_3,m) = 1 \wedge \chi(c_2,n) - \chi(c_2,m) \geq 1))$	(SampledOn)

One limitation of the SMT-based approach is that SMT solvers may not always return a result for some generated SMT formulas because the satisfiability problem in the logic in which the generated SMT formulas are defined is undecidable. One solution is to set a bound and do bounded model checking. Bounded model checking is useful to verify the behaviors of systems within a bounded number of steps. To overcome the limitation on boundedness, we identify a special class of clock behaviors of CCSL which are periodic in that clocks tick periodically after some step. Then, we propose sufficient conditions under which LTL properties can be verified with no limitation on boundedness for such periodic behaviors using bounded model checking.

In summary, the contributions of this work are as follows:

1. An approach is proposed to achieve model checking of LTL properties of CCSL constraints by transforming all the CCSL constraints and LTL properties into SMT formulas;
2. Sufficient conditions are proposed for bounded model checking of LTL properties of CCSL constraints with respect to periodic behaviors of CCSL;
3. Two practical case studies are conducted. Experimental results demonstrate the effectiveness and efficiency of our approach.

Organization of the paper: Section 2 introduces the CCSL language and discusses its schedulability problem. Section 3 and 4 present approaches to transforming CCSL constraints and LTL formulas into SMT formulas, respectively. Section 5 describes two case studies and shows the experimental results. Section 6 mentions some related work, and Section 7 concludes the paper.

2. CCSL and its Schedulability Problem

In this section, we present the syntax and semantics of CCSL and explain its schedulability problem.

2.1 Syntax and Semantics of CCSL

Before introducing CCSL, we first give the definition of logical clock, schedule and history.

Definition 1 (Logical clock). *A logical clock c is an infinite sequence of ticks $(c^i)_{i\in\mathbb{N}^+}$ with each c^i being* tick *or* idle.

We use $\mathbb{N}^+$ to denote the set of all the non-zero natural numbers. If the value of c^i is *tick*, it means that the event associated to c occurs at step i, and does not occur otherwise.

Definition 2 (Schedule). *Given a set C of clocks, a schedule of C is a total function $\delta : \mathbb{N}^+ \to 2^C$ such that for all i in $\mathbb{N}^+$, $\delta(i) = \{c | c \in C \wedge c^i = tick\}$ and $\delta(i) \neq \emptyset$.*

Intuitively, $\delta(i)$ is the subset of all the clocks in C that tick at step i. The condition $\delta(i) \neq \emptyset$ says that it is not allowed that no clocks tick at a step. Such a step is called an *empty step*, which is trivial in that whether they exist in a schedule or not does not affect the satisfiability of the schedule with logical relations among the clocks. Thus, we exclude empty steps from schedules for succinctness.

Definition 3 (History). *Given a schedule $\delta : \mathbb{N}^+ \to 2^C$ on a set C of clocks, a history of δ is a function $\chi_\delta : C \times \mathbb{N}^+ \to \mathbb{N}$ such that for each clock $c \in C$ and natural number $i \in \mathbb{N}^+$:*

$$\chi_\delta(c,i) = \begin{cases} 0 & \text{if } i = 1 \\ \chi_\delta(c,i-1) & \text{if } i > 1 \wedge c \notin \delta(i-1) \\ \chi_\delta(c,i-1) + 1 & \text{if } i > 1 \wedge c \in \delta(i-1) \end{cases}$$

Obviously, $\chi_\delta(c,i)$ records the number of the ticks that clock c has ticked immediately before it reaches step i. [1] For simplicity, we also write χ for χ_δ if it is clear from the context.

In the work [16], clock relations are divided into two classes, i.e., clock constraints and clock definitions. There are four primitive constraints called Precedence, Causality, Subclock and Exclusion. Four binary constraint operators are defined for them, respectively. There are eight kinds of clock definitions called Union, Intersection, Infimum, Supremum, Delay, DelayFor, SampledOn and Periodicity. For the sake of simplicity, we call both clock constraints and clock definitions CCSL *constraints* or *constraints*.

Table 1 shows the syntax and semantics of the twelve CCSL constraints. The semantics of CCSL constraints are defined in terms of the satisfaction of schedules with the constraints. We use $\delta \models \phi$ to denote that schedule δ satisfies constraint ϕ, and use $\delta \models \Phi$ to denote the satisfaction of δ with a set Φ of constraints, i.e., $\delta \models \Phi$ if and only if $\forall \phi \in \Phi.\delta \models \phi$. Below we simply explain the intuitive meaning of each constraint.

- Precedence: $c_1\ [b] \prec c_2$ means that clock c_1 is at least b steps faster than c_2, where $b \in \mathbb{N}$, i.e., at each step i if c_1 has ticked b times more than c_2 (if $b > 0$) or the times they have ticked are the same (if $b = 0$) up to the step, c_2 should not tick at step i. In particular when $b = 0$, the constraint is written as $c_1 \prec c_2$.

[1] Note that the tick of c at step i is not counted in $\chi_\delta(c,i)$.

- Causality: $c_1 \preccurlyeq c_2$ means that clock c_1 is not slower than c_2.
- Subclock: $c_1 \subseteq c_2$ means that at each step the tick of c_1 results in the tick of of c_2.
- Exclusion: $c_1 \ \# \ c_2$ means that at each step c_1, c_2 cannot tick simultaneously.
- Union: $c_1 \triangleq c_2 + c_3$ defines a clock c_1 such that at each step c_1 ticks iff either c_2 or c_3 ticks.
- Intersection: $c_1 \triangleq c_2 * c_3$ defines a clock c_1 such that at each step c_1 ticks iff both c_2 and c_3 ticks.
- Infimum: $c_1 \triangleq c_2 \wedge c_3$ defines a clock c_1 which is the slowest clock but faster than both c_2 and c_3.
- Supremum: $c_1 \triangleq c_2 \vee c_3$ defines a clock c_1 which is the fastest clock but slower than both c_2 and c_3.
- Delay: $c_1 \triangleq c_2 \ \$ \ d$ defines a clock c_1 which is delayed by c_2 with d times c_2 occurs.
- DelayFor: $c_1 \triangleq c_2 \ \$ \ d \ on \ c_3$ defines a sampling clock c_1 with c_2 being sampled every d times c_3 occurs. When c_2 and c_3 are the same, there is $c_1 \triangleq c_2 \ \$ \ d \ on \ c_2$ which equals to $c_1 \triangleq c_2 \ \$ \ d$. That is, Delay is a special case of DelayFor.
- PeriodicOn: $c_1 \triangleq c_2 \propto p$ defines a clock c_1 which periodically ticks together with c_2 every p times c_2 occurs.
- SampledOn: $c_1 \triangleq c_2 \lightning c_3$ defines a sampling clock c_1 which is the fastest but slower than c_2 by synchronizing c_2 on clock c_3.

Interested readers are referred to the work [1, 16] for further details about the physical meanings of the CCSL constraints.

Example 1. *We consider a simple example of specifying alternative blinking between the green and red light using* CCSL*. We assume that green light starts first. Such requirements can be formalized by the following* CCSL *constraints:*

$$green \prec red \tag{1}$$

$$tmp \triangleq green \ \$ \ 1 \tag{2}$$

$$red \prec tmp \tag{3}$$

We declare two clocks $green$ and red to represent the green and red lights respectively. Ticking of clock $green$ (resp. red) represents the event that the green (resp. red) light is turned on, and idling represents being turned off. Clock *tmp* is an intermediate (or virtual) clock that does not represent any real event, but is only used to help specify the constraints on clocks. Although it is possible to compose a complex constraint to avoid intermediate clocks, e.g., $red \prec (green \ \$ \ 1)$ for (2) and (3), using intermediate clock makes each constraint simpler and more readable.

For the above three constraints in the example, there exists one and only one schedule denoted by δ_{E1}, which is defined as follows:

$$\delta_{E1}(i) = \begin{cases} \{green\} & \text{if } i \text{ is odd,} \\ \{red\} & \text{if } i \text{ is even and } i \neq 0. \end{cases} \tag{4}$$

Note that we do not consider intermediate clocks in the definition of schedules. In the work [16], the three constraints are encoded into a finite-state automata, and there exists one and only one infinite path from the initial state. The path is considered the unique schedule that satisfies the three constraints.

2.2 The Schedulability Problem of CCSL

Given a set of CCSL constraints, one basic question to answer is whether there exists a schedule satisfying all the constraints. It is called the schedulability problem of CCSL constraints. To our knowledge, there is no procedure proposed to check the existence of schedules for an arbitrarily given set of CCSL constraints. Two reasons together make the problem challenging. One reason is that schedules are infinite, and the other is that some constraints such as Precedence and DelayFor are *stateful* in that the behavior of the clocks in such constraints at one step depends on the history of these clocks.

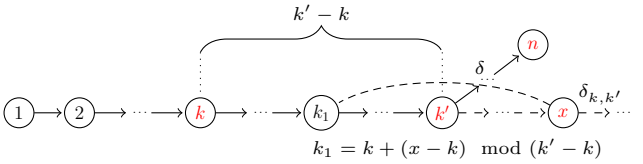


Figure 1. Construction of a periodic schedule $\delta_{k,k'}$ from an n-bounded schedule δ.

A simple solution to the schedulability problem of CCSL constraints is to set bound to schedules. However, bounded schedules are usually too restrictive in practice for real-time and embedded systems. Among infinite schedules there is a special class which we call *periodic schedules* in that by such schedules clocks tick periodically after some step. Periodic schedules are practically important to real-time and embedded systems for the periodicity features of such systems.

Definition 4 (Periodic schedule)**.** *A schedule δ is called* periodic *if there exist two natural numbers k and p such that for each k' in $\mathbb{N}^+$ such that $k' \geq k$, there is $\delta(k' + p) = \delta(k')$.*

Definition 4 says that after step k, all clocks repeat infinitely their behavior as they do from step k to step $k + p - 1$. p is called a period of δ.

Checking the existence of periodic schedules is still challenging for the same reasons as aforementioned. One alternative solution is to *construct* a periodic schedule by extending a bounded schedule where there are two steps at which the set of ticking clocks are the same. It is a decidable problem of finding such schedules when a bound is provided because the number of potential schedules within a bound is finite.

Definition 5. *Given a non-zero natural number n, a schedule δ is called an n-bounded schedule for a set Φ of* CCSL *constraints if δ satisfies Φ from step 1 to n.*

Definition 6 (Extending bounded schedules to periodic schedules)**.** *Given an n-bounded schedule δ and two natural numbers k, k' such that $k < k' \leq n$ and $\delta(k) = \delta(k')$, let $\delta_{k,k'}$ be a periodic schedule defined as follows:*

$$\delta_{k,k'}(x) = \begin{cases} \delta(x) & \text{if } x \leq k' \\ \delta(k + (x - k) \mod (k' - k)) & \text{if } x > k' \end{cases},$$

Figure 1 depicts the construction of a periodic schedule $\delta_{k,k'}$ from a bounded schedule δ whose bound is n. Each node represents one step, and a path represents a schedule.

A bounded schedule is called a periodic bounded schedule if it can be extended to a periodic one by Definition 6, and is called a non-periodic bounded schedule otherwise. Given a bound n, the problem of checking the existence of an n-bounded schedule δ for a set Φ of CCSL constraints such that there exist two steps k and k' within n steps and $\delta(k) = \delta(k')$ is decidable, which can be straightforwardly derived from the pigeonhole principle. However, the periodic schedule $\delta_{k,k'}$ extended from δ may not satisfy Φ. Figure 2 shows a periodic schedule that is extended from a bounded one with $k = 2$ and $k' = 8$ of the causality constraint $c_1 \preccurlyeq c_2$ but does not satisfy the constraint. The schedule with bound 8 satisfies

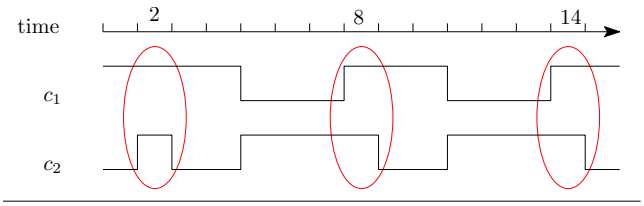


Figure 2. A periodic schedule that is extended from a bounded one of $c_1 \preccurlyeq c_2$ but does not satisfy $c_1 \preccurlyeq c_2$

$c_1 \preccurlyeq c_2$. Because the clocks that tick at step 2 and 8 are the same, i.e., c_1 and c_2, we can extend the bounded schedule by Definition 6. However, the periodic schedule does not satisfy $c_1 \preccurlyeq c_2$ at step 14 because $\chi(c_1, 14) = 7$ while $\chi(c_2, 14) = 8$.

In our earlier work [22], sufficient conditions are proposed to guarantee that extended periodic schedules still satisfy the constraints that their corresponding bounded schedules satisfy. That is, when a bounded schedule δ is found for a given set Φ of CCSL constraints and δ satisfies the conditions, if there exist two steps k and k' such that $\delta(k) = \delta(k')$, then $\delta_{k,k'}$ is a periodic schedule of Φ. However, because constraints DelayFor and SampledOn are not considered in that work, the conditions become insufficient after the two constraints are taken into consideration.

Figure 3 shows a concrete example where a periodic schedule that is extended from a bounded one does not satisfy a DelayFor constraint. Let δ be the schedule with a bound 8, as depicted in the figure. According to the definition of DelayFor, δ is a schedule of the constraint $c_1 \triangleq c_2 \ \$ \ 4 \ on \ c_3$. Because $\delta(2) = \delta(8)$, we can extend δ to a periodic one, i.e., $\delta_{2,8}$ by Definition 6. However, $\delta_{2,8}$ does not satisfy $c_1 \triangleq c_2 \ \$ \ 4 \ on \ c_3$ because c_1 must tick at step 9 according to the constraint while $c_1 \notin \delta_{2,8}(9)$, and c_1 must not tick at step 12 while $c_1 \in \delta_{2,8}(12)$. The reason for the violation is that the tick of c_2 which determines the tick of c_1 in the first period from step 2 to 8 occurs at step 1, which is outside the same period where c_1 ticks.

Next, we propose two sufficient conditions for DelayFor and SampledOn constraints, respectively.

Theorem 1. *Let δ be a k'-bounded schedule of a clock constraint $\phi = c_1 \triangleq c_2 \ \$ \ d \ on \ c_3$ and there exists a natural number k such that $k < k'$ and $\delta(k) = \delta(k')$. The periodic schedule $\delta_{k,k'}$ satisfies ϕ if the following two conditions hold:*

1. $\forall x \in \{k, \ldots, k'-1\}.(c_3 \in \delta(x) \wedge \exists m \in \mathbb{N}^+.c_2 \in \delta(m) \wedge \chi(c_3, x) - \chi(c_3, m) = d) \implies m \geq k$;
2. $\forall x \in \{1, \ldots, k'-1\}.c_2 \in \delta(x) \implies \exists m \in \mathbb{N}^+.c_1 \in \delta(m) \wedge x \leq m < k'$.

Intuitively, condition 1 means that if c_1 ticks in the first period, its corresponding base clock c_2 must have ticked also in the same period before c_1 ticks, and condition 2 says that each tick of c_2 up to step k' must have a corresponding delayed tick of c_1 up to step k'. By condition 2, it forces that all the delayed ticks of c_1 that are caused by the ticks of c_2 must occur before iterating the schedule from step k to $k'-1$. By the two conditions, c_1 ticks in a period if and only if c_2 ticks in the same period.

Proof of Theorem 1. By definition of $\delta_{k,k'}$, it suffices to prove that for all $i > k'$, $c_1 \in \delta_{k,k'}(i) \iff c_3 \in \delta_{k,k'}(i) \wedge \exists m' \in \mathbb{N}^+.(c_2 \in \delta_{k,k'}(m') \wedge \chi(c_3, i) - \chi(c_3, m') = d)$. Because $i > k'$, $\delta_{k,k'}(i) = \delta(k + (i-k) \mod (k'-k))$. For simplicity, we use j to stand for $k + (i-k) \mod (k'-k)$ in the proof.

- From left to right: $c_1 \in \delta_{k,k'}(i)$ implies that $c_1 \in \delta(j)$. Because δ is a k'-bounded schedule of ϕ and $j < k'$, $\delta; j \models \phi$. Thus, $c_3 \in \delta(j)$ and there exists $m \in \mathbb{N}^+$ such that $c_2 \in \delta(m)$ and $\chi(c_3, j) - \chi(c_3, m) = d$. By condition 1, $k \leq m < j$. Let $m' = m + \left\lfloor \frac{i-k}{k'-k} \right\rfloor * (k'-k)$. By definition of $\delta_{k,k'}$, $\delta_{k,k'}(m') = \delta(m)$. Thus, $c_2 \in \delta_{k,k'}(m')$. Because for all $j' \in \{1, \ldots, i-m'-1\}$ there is $\delta_{k,k'}(m'+j) = \delta(m+j)$ and $\chi(c_3, i) - \chi(c_3, m') = \chi(c_3, j) - \chi(c_3, m) = d$. This direction is proved.
- From right to left: $c_3 \in \delta_{k,k'}(i)$ implies that $c_3 \in \delta(j)$. With condition 1 and 2, it is straightforward to prove that $m' \geq k + \left\lfloor \frac{i-k}{k'-k} \right\rfloor * (k'-k)$ by contradiction. Let $m = m' - \left\lfloor \frac{i-k}{k'-k} \right\rfloor * (k'-k)$, i.e., $m = k + (m'-k) \mod (k'-k)$. By definition of $\delta_{k,k'}$, $\delta_{k,k'}(m') = \delta(m)$ and also for all j in $\{1, \ldots, i-m'-1\}$ there is $\delta_{k,k'}(m'+j) = \delta(m+j)$. Thus, $\chi(c_3, j) - \chi(c_3, m) = \chi(c_3, i) - \chi(c_3, m') = d$, which implies that $c_1 \in \delta(j)$ and hence $c_1 \in \delta_{k,k'}(i)$. This direction is proved.

The theorem is proved. □

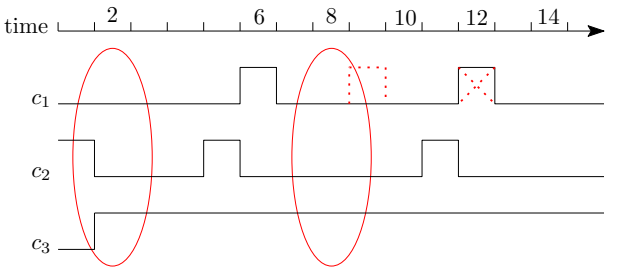


Figure 3. A periodic schedule that is extended from a bounded one of $c_1 \triangleq c_2 \ \$ \ 4 \ on \ c_3$ but does not satisfy $c_1 \triangleq c_2 \ \$ \ 4 \ on \ c_3$

Theorem 2. *Let δ be a k'-bounded schedule of a clock constraint $\phi = c_1 \triangleq c_2 \not\downarrow c_3$ and there exists a natural number k such that $k < k'$ and $\delta(k) = \delta(k')$. The periodic schedule $\delta_{k,k'}$ satisfies ϕ if the following two conditions hold:*

1. $\forall x \in \{k, \ldots, k'-1\}.(c_3 \in \delta(x) \wedge \exists m \in \mathbb{N}^+.\chi(c_3, x) - \chi(c_3, m) = 1 \wedge \chi(c_2, x) - \chi(c_2, m) \geq 1) \implies m \geq k$
2. $\forall x \in \{k, \ldots, k'-1\}.(c_2 \in \delta(x) \implies \exists y, z \in \{x+1, k'-1\}.c_3 \in \delta(y) \wedge c_3 \in \delta(z))$.

Condition 1 in Theorem 2 is similar to the first condition in Theorem 1. It means that if c_1 ticks in a periodic then its sampling clock c_3 must also occur in the same periodic. The second condition means that if c_2 occurs in a period c_3 must tick twice in the same period but after c_2 ticks. By this condition, it forces that if c_3 ticks

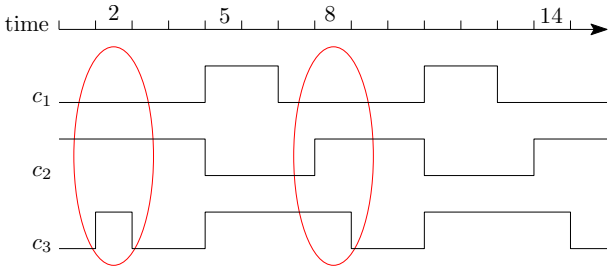


Figure 4. A bounded schedule of the constraint $c_1 \triangleq c_2 \not\downarrow c_3$ that satisfies the two conditions in Theorem 2

Table 2. Encoding CCSL constraints into SMT formulas

ϕ	$[\![\phi]\!]$
$c_1\ [b] \prec c_2$	$\forall n \in \mathbb{N}^+.h_{c_2}(n) - h_{c_1}(n) = b \implies \neg t_{c_2}(n)$
$c_1 \preccurlyeq c_2$	$\forall n \in \mathbb{N}^+.h_{c_1}(n) \geq h_{c_2}(n)$
$c_1 \subseteq c_2$	$\forall n \in \mathbb{N}^+.t_{c_1}(n) \implies t_{c_2}(n)$
$c_1 \ \# \ c_2$	$\forall n \in \mathbb{N}^+.\neg t_{c_1}(n) \vee \neg t_{c_2}(n)$
$c_1 \triangleq c_2 + c_3$	$\forall n \in \mathbb{N}^+.(t_{c_1}(n) \iff t_{c_2}(n) \vee t_{c_3}(n))$
$c_1 \triangleq c_2 * c_3$	$\forall n \in \mathbb{N}^+.(t_{c_1}(n) \iff t_{c_2}(n) \wedge t_{c_3}(n))$
$c_1 \triangleq c_2 \wedge c_3$	$\forall n \in \mathbb{N}^+.h_{c_1}(n) = max(h_{c_2}(n), h_{c_3}(n))$
$c_1 \triangleq c_2 \vee c_3$	$\forall n \in \mathbb{N}^+.h_{c_1}(n) = min(h_{c_2}(n), h_{c_3}(n))$
$c_1 \triangleq c_2 \ \$ \ d$	$\forall n \in \mathbb{N}^+.h_{c_1}(n) = max(h_{c_2}(n) - d, 0)$
$c_1 \triangleq c_2 \ \$ \ d$	$\forall n \in \mathbb{N}^+.(t_{c_1}(n) \iff (t_{c_3}(n) \wedge \exists m \in \mathbb{N}^+.(t_{c_2}(m) \wedge h_{c_3}(n) - h_{c_3}(m) = d)))$
$c_1 \triangleq c_2 \propto p$	$\forall n \in \mathbb{N}^+.(t_{c_1}(n) \iff (t_{c_2}(n) \wedge \exists m \in \mathbb{N}^+.h_{c_2}(n) = m \times p - 1))$
$c_1 \triangleq c_2 \ \lightning \ c_3$	$\forall n \in \mathbb{N}^+.(t_{c_1}(n) \iff (t_{c_3}(n) \wedge (\exists m \in \mathbb{N}^+.t_{c_3}(m) \wedge h_{c_3}(n) - h_{c_3}(m) = 1 \wedge h_{c_2}(n) - h_{c_2}(m) \geq 1)))$

at some step i in a period and there exists a step m' such that $\chi(c_3, i) - \chi(c_3, m') = 1$ and $\chi(c_2, i) - \chi(c_2, m') \geq 1$ then m' must be a step in the same period before i.

Figure 4 shows an example on a bounded schedule of the constraint $c_1 \triangleq c_2 \ \lightning \ c_3$ and the schedule satisfies the two conditions with $k = 2$ and $k' = 8$. Because of condition 1, there is no an m in $\mathbb{N}^+$ such that $m \geq 2$ and $\chi(c_3, 2) - \chi(c_3, m) = 1$, and hence $c_1 \notin \delta(2)$ although $c_3 \in \delta(2)$. Because of condition 2, there does not either exist an m in $\mathbb{N}^+$ such that $\chi(c_3, 8) - \chi(c_3, m) = 1$ and $\chi(c_2, 8) - \chi(c_2, m) \geq 1$, and hence $c_1 \notin \delta(8)$. At step 5, there exists $m = 2$ such that $\chi(c_3, 5) - \chi(c_3, 2) = 1$ and $\chi(c_2, 5) - \chi(c_2, 2) = 3 \geq 1$. Therefore $c_1 \in \delta(5)$. By repeating the period from step 2 to 7 at step 8, we obtain a periodic schedule, and obviously the periodic schedule satisfies the constraint.

The proof of Theorem 2 is similar to the one of Theorem 1, and we omit it in the paper due to the limitation of space.

3. Encoding CCSL Constraints into SMT

CCSL constraints can be naturally encoded into SMT formulas by interpreting each clock c as a predicate $t_c : \mathbb{N}^+ \to Bool$, where $Bool$ is the set of Boolean values. For a step $i (i \in \mathbb{N}^+)$, $t_c(i)$ being true means that clock c ticks at i, and idles otherwise.

We introduce an auxiliary function $h_c : \mathbb{N}^+ \to \mathbb{N}$ for each clock such that for each $i \in \mathbb{N}^+$:

$$h_c(i) = \begin{cases} 0 & \text{if } i = 1 \\ h_c(i-1) & \text{if } i > 1 \wedge \neg t_c(i-1) \\ h_c(i-1) + 1 & \text{if } i > 1 \wedge t_c(i-1) \end{cases} \quad \text{(F1)}$$

Intuitively, $h_c(i)$ represents the number of ticks of c immediately before it reaches step i.

Given a set Φ of CCSL constraints on a set C of clocks, let T_C be the set of uninterpreted predicates for the clocks in C, i.e., $T_C = \{t_c | c \in C\}$, and H_C be the set of auxiliary functions for the clocks in C. Finding a schedule for Φ is equivalent to finding a solution (or definition) for each predicate in T_C.

Table 2 shows the transformation from CCSL constraints into SMT formulas in First-Order Logic (FOL). The transformation is straightforward by replacing each $c \in \delta(n)$ (resp. $c \notin \delta(n)$) with $t_c(n)$ (resp. $\neg t_c(n)$) and $\chi(c, n)$ with $h_c(n)$.

The uninterpreted predicates in T_C and the functions in H_C also need to satisfy an extra condition besides Formula F1 and the formulas that are transformed from given CCSL constraints. According to Definition 2, at each step there must be at least one clock ticking. Thus, the predicates in T_C must satisfy the following formula for the condition:

$$\forall n \in \mathbb{N}^+. \bigvee_{c \in C} t_c(n) \quad \text{(F2)}$$

We use $[\![\Phi]\!]$ to denote the conjunction of Formula F1, F2 and the formulas that are transformed from Φ, i.e.,

$$[\![\Phi]\!] = F1 \wedge F2 \wedge (\bigwedge_{\phi \in \Phi} [\![\phi]\!]).$$

The schedulability problem of Φ is transformed into the satisfiability problem of the formula $[\![\Phi]\!]$. According to the SMT-LIB standard [5], $[\![\Phi]\!]$ belongs to the logic of of UFLIA (formulas with Uninterpreted Functions and Linear Integer Arithmetic), while the satisfiability problem of the formulas in UFLIA is undecidable. Provided that there is a bound b ($b \in \mathbb{N}$) to the universally quantified variable n in the formula, the quantifiers in $[\![\Phi]\!]$ can be eliminated and the satisfiability problem becomes decidable. We use $[\![\Phi]\!]_b$ to denote the formula $[\![\Phi]\!]$ with the range of n being $\{1, \ldots, b\}$. Although bounded satisfiability does not guarantee the satisfiability of $[\![\Phi]\!]$, there are still useful applications, such as invalidity proving and execution trace analysis. Invalidity proving means to prove the invalidity of CCSL constraints, and execution trace analysis means to check if an execution trace (a bounded schedule) satisfies given CCSL constraints. Details of such applications can be found in our earlier work [23].

The existence of a bounded solution to $[\![\Phi]\!]$ implies the satisfiability of $[\![\Phi]\!]$ if the corresponding bounded schedule can be extended to a periodic one by Definition 6. To find such a bounded solution, we transform the sufficient conditions that are proposed in the work [23] and those in Theorem 1 and 2 for the constraint DelayFor and SampledOn into SMT formulas. Since the transformation is straightforward as CCSL constraints are transformed, we only take the transformation of the conditions in Theorem 1 as an example. The two conditions are transformed into the following two formulas correspondingly:

$$\forall x \in \{k, \ldots, k' - 1\}.t_{c_3}(x) \wedge \exists m \in \{1, \ldots, x - 1\}. \\ (t_{c_2}(m) \wedge h_{c_3}(x) - h_{c_3}(m) = d) \implies m \geq k \quad \text{(F3)}$$

$$\forall x \in \{1, \ldots, k' - 1\}.t_{c_2}(x) \implies \exists m \in \{x, \ldots, k' - 1\}.t_{c_1}(m) \quad \text{(F4)}$$

In the above formulas, k and k' are two constant natural numbers. Quantifiers in the two formulas can be eliminated, and hence the satisfiability problem of the generated SMT formulas is still de-

cidable after adding the above formulas as conjuncts to $[\![\Phi]\!]_b$. We denote the new conjunction by ${}^{k'}_{k}[\![\Phi]\!]_b$. Once a solution to ${}^{k'}_{k}[\![\Phi]\!]_b$ is found, the predicates in T_C are defined and k, k' are instantiated by some concrete natural numbers. We obtain a periodic schedule for Φ with k and k' being the starting steps of the first and second periods in the schedule.

4. Encoding LTL Formulas for CCSL into SMT

LTL is an extension of propositional logic for specifying temporal properties of reactive systems. In LTL, there are logical connectives, i.e., $\neg, \wedge, \vee, \implies$, and temporal connectives, i.e., X (next), F (future), G (globally), U (until). Let AP be a set of atomic properties. The syntax of LTL formulas is defined by:

$$\psi ::= \top \mid \bot \mid a \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \psi \implies \psi \\ X\psi \mid F\psi \mid G\psi \mid \psi U \psi,$$

where $a \in AP$, $\top$ is true and $\bot$ is false. Other standard temporal connectives such as W (weak-until) and R (release) can be defined as abbreviations of other existing connectives: $\psi W \psi = \psi U \psi \vee G\psi$ and $\psi R \psi = \neg(\neg\psi U \neg\psi)$.

We suppose that the atomic propositions in AP are clocks and an atomic proposition is true at some step i if and only if its corresponding clock ticks at i. Then, an LTL formula constructed with the atomic propositions in AP and some connectives above represents a temporal properties of the behaviors of clocks. In the rest of this paper, an LTL formula only refers to those whose atomic properties are clocks by default without extra explanation.

Below we give the semantics of LTL whose AP is a set of clocks with respect to CCSL schedules.

Definition 7 (Semantics of LTL w.r.t. infinite schedules)**.** *The satisfiability of an LTL formula ψ w.r.t. an infinite schedule δ, denoted by $\delta \models \psi$, is defined as follows:*

$$\begin{aligned}
&\delta \models c && \iff c \in \delta(1)\\
&\delta \models \neg\psi && \iff \delta \not\models \psi\\
&\delta \models \psi_1 \wedge \psi_2 && \iff \delta \models \psi_1 \wedge \delta \models \psi_2\\
&\delta \models \psi_1 \vee \psi_2 && \iff \delta \models \psi_1 \vee \delta \models \psi_2\\
&\delta \models \psi_1 \implies \psi_2 && \iff \delta \models \psi_1 \implies \delta \models \psi_2\\
&\delta \models X\psi && \iff \delta^2 \models \psi\\
&\delta \models F\psi && \iff \exists i \geq 1.(\delta^i \models \psi)\\
&\delta \models G\psi && \iff \forall i \geq 1.(\delta^i \models \psi)\\
&\delta \models \psi_1 U \psi_2 && \iff \exists i \geq 1.\forall j < i.(\delta^i \models \psi_2 \wedge \delta^j \models \psi_1)
\end{aligned}$$

where, $\delta^i : \mathbb{N}^+ \to Bool$ such that $\delta^i(j) = \delta(j + i - 1)$ for all j in $\mathbb{N}^+$.

Given a schedule δ and an LTL formula ψ, the satisfiability of ψ w.r.t. δ is undecidable because of the existence of quantifiers in the definition of $\delta \models \psi$. Thus, we consider *bounded model checking* of LTL properties, i.e., to model check the satisfiability of an LTL property w.r.t. bounded or periodic schedules. Namely, we set an upper bound n in schedules and check if all the bounded schedules within bound n satisfy a given LTL formula.

We then give the semantics of LTL w.r.t. bounded schedules. We need to consider two cases of bounded schedules, i.e., periodic ones and non-periodic ones, because the semantics of LTL w.r.t. them are different.

Definition 8 (Semantics of LTL w.r.t. non-periodic bounded schedules)**.** *The satisfiability of an LTL formula ψ w.r.t. δ from step i to n, denoted by $\delta \models^i_n \psi$, is defined as follows:*

$$\begin{aligned}
&\delta \models^i_n c && \iff c \in \delta(i)\\
&\delta \models^i_n \neg\psi && \iff \delta \not\models^i_n \psi\\
&\delta \models^i_n \psi_1 \wedge \psi_2 && \iff (\delta \models^i_n)\psi_1 \wedge (\delta \models^i_n \psi_2)\\
&\delta \models^i_n \psi_1 \vee \psi_2 && \iff (\delta \models^i_n \psi_1) \vee (\delta \models^i_n \psi_2)\\
&\delta \models^i_n \psi_1 \implies \psi_2 && \iff (\delta \models^i_n \psi_1) \implies (\delta \models^i_n \psi_2)\\
&\delta \models^i_n X\psi && \iff (i < n) \wedge (\delta \models^{i+1}_n \psi)\\
&\delta \models^i_n F\psi && \iff \exists j \in \{i, \cdots, n\}.(\delta \models^j_n \psi)\\
&\delta \models^i_n G\psi && \iff \bot\\
&\delta \models^i_n \psi_1 U \psi_2 && \iff \exists j \in \{i, \ldots, n\}.\forall k \in \{i, \ldots, j-1\}.\\
& && \quad (\delta \models^j_n \psi_2 \wedge \delta \models^k_n \psi_1)
\end{aligned}$$

An n-bounded schedule δ satisfies an LTL formula ψ if and only if $\delta \models^1_n$ is true. Let $[\![\psi]\!]^i_n$ denote an SMT formula such that for all periodic bounded schedule δ, $\delta \models^i_n \psi$ holds if and only if δ satisfies $[\![\psi]\!]^i_n$. Table 3 shows the definition of $[\![\psi]\!]^i_n$. One can see the clear correspondence between the semantics in Definition 8 and the formula in Table 3.

Table 3. Encoding LTL formulas into SMT formulas w.r.t. non-periodic bounded schedules

ψ	$[\![\psi]\!]^i_n$
c	$t_c(i)$
$\neg\psi$	$\neg[\![\psi]\!]^i_n$
$\psi_1 \wedge \psi_2$	$[\![\psi_1]\!]^i_n \wedge [\![\psi_2]\!]^i_n$
$\psi_1 \vee \psi_2$	$[\![\psi_1]\!]^i_n \vee [\![\psi_2]\!]^i_n$
$\psi_1 \implies \psi_2$	$[\![\psi_1]\!]^i_n \implies [\![\psi_2]\!]^i_n$
$X\psi$	$[\![\psi]\!]^{i+1}_n$ if $i < n$ $\bot$ otherwise
$F\psi$	$\exists j \in \{i, \ldots, n\}.[\![\psi]\!]^j_n$
$G\psi$	$\bot$
$\psi_1 U \psi_2$	$\exists j \in \{i, \ldots, n\} \forall k \in \{i, \ldots, j-1\}.([\![\psi_2]\!]^j_n \wedge [\![\psi_1]\!]^k_n)$

Definition 9 (Semantics of LTL w.r.t. periodic bounded schedules)**.** *Let δ be a periodic bounded schedule and l and k be two steps such that $\delta_{l,k}$ is an extended infinite periodic schedule. The satisfiability of an LTL formula ψ w.r.t. δ from step i to k, denoted by ${}_l\delta \models^i_k \psi$, is defined as follows:*

$$\begin{aligned}
&{}_l\delta \models^i_k \psi && \iff \delta \models^i_k \psi, \text{ if } \psi \text{ is in the form of } c, \neg\psi_1,\\
& && \quad \psi_1 \wedge \psi_2, \psi_1 \vee \psi_2 \text{ or } \psi_1 \implies \psi_2.\\
&{}_l\delta \models^i_k X\psi && \iff (i < k \implies \delta \models^{i+1}_k \psi) \wedge\\
& && \quad (i = k \implies \delta \models^{l+1}_k \psi)\\
&{}_l\delta \models^i_k G\psi && \iff (i < l) \implies \forall j \in \{i, \ldots, k\}.\delta \models^j_k \psi \wedge\\
& && \quad (l \leq i \leq k).\forall j \in \{l, \ldots, k\}.\delta \models^j_k \psi\\
&{}_l\delta \models^i_k F\psi && \iff (i < l) \implies \exists j \in \{i, \ldots, k\}.\delta \models^j_k \psi \wedge\\
& && \quad (l \leq i \leq k).\exists j \in \{l, \ldots, k\}.\delta \models^j_k \psi\\
&{}_l\delta \models^i_k \psi_1 U \psi_2 && \iff (\exists j \in \{i, \ldots, k\} \forall j' \in \{i, \ldots, j-1\}.\\
& && \quad {}_l\delta \models^j_k \psi_2 \wedge {}_l\delta \models^{j'}_k \psi_1) \vee \exists j \in \{l, \ldots, k\}\\
& && \quad \forall j' \in \{l, \ldots, j-1\} \forall j'' \in \{i, \ldots, k\}\\
& && \quad {}_l\delta \models^j_k \psi_2 \wedge {}_l\delta \models^{j'}_k \psi_1 \wedge {}_l\delta \models^{j''}_k \psi_1.
\end{aligned}$$

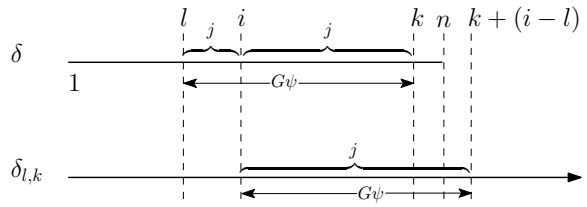


Figure 5. One case of defining the semantics of $G\psi$ w.r.t. periodic schedules in terms of bounded ones

Definition 9 gives the semantics of LTL w.r.t. an n-bounded periodic schedule δ. Provided that δ is a periodic one, there must exist two natural numbers l, k such that $l < k \leq n$ and $\delta_{l,k}$ is an infinite periodic schedule according to Definition 6. The difference between Definition 9 from Definition 8 is the definition of temporal connectives. For instance, to check $X\psi$ at step k, it suffices to check ψ at step $l+1$ for the periodicity of δ. Similarly, to check $G\psi$ at step i, it is equivalent to check whether ψ holds at all the steps from i to k when $i < k$, and to check from l to k when $l \leq i \leq k$. Figure 5 depicts the latter case for $G\psi$ when $i < k$. The definition for F and U are similar. Apparently, $\delta \,{}_l\models^1_k \psi$ is equivalent to $\delta_{i,k} \models \psi$. That is, to model check LTL properties of periodic schedules, it is equivalent to model check the first k steps where k is the step where the iteration of period is started.

Let ${}_l[\![\psi]\!]^i_k$ denote an SMT formula such that for all schedule δ, $\delta \,{}_l\models^i_k \psi$ if and only if δ satisfies ${}_l[\![\psi]\!]^i_k$. Table 4 shows the definition of ${}_l[\![\psi]\!]^i_k$ for each case of ψ. The correspondence between $\delta \,{}_l\models^i_k \psi$ and their SMT formulas is straightforward. We omit detailed explanations due to limited space.

Using the above encoding approaches we obtain an SMT formula $[\![\psi]\!]^1_n$ of ψ for model checking of non-periodic bounded schedules, and ${}_l[\![\psi]\!]^1_k$ for model checking of periodic bounded schedules. Given a set Φ of CCSL constraints, to bounded model check if all non-periodic (resp. periodic) schedules of Φ satisfy ψ within a bound n is equivalent to check if $[\![\Phi]\!]_n \wedge \neg[\![\psi]\!]^1_n$ (resp. ${}^k_l[\![\Phi]\!]_n \wedge \neg\,{}_l[\![\psi]\!]^1_k$) is satisfiable or not. If the formula is unsatisfiable, it means that the property represented by ψ is satisfied by all schedules of Ψ up to step n (resp. by all periodic schedules of Φ that are found up to step n). Otherwise, a solution is found as a counterexample which shows the violation of a schedule against the property.

Table 4. Encoding LTL formulas into SMT formulas w.r.t. periodic bounded schedules

ψ	${}_l[\![\psi]\!]^i_k$
c	$t_c(i)$
$\neg\psi$	$\neg{}_l[\![\psi]\!]^i_k$
$\psi_1 \wedge \psi_2$	${}_l[\![\psi_1]\!]^i_k \wedge {}_l[\![\psi_2]\!]^i_k$
$\psi_1 \vee \psi_2$	${}_l[\![\psi_1]\!]^i_k \vee {}_l[\![\psi_2]\!]^i_k$
$\psi_1 \Longrightarrow \psi_2$	${}_l[\![\psi_1]\!]^i_k \Longrightarrow {}_l[\![\psi_2]\!]^i_k$
$X\psi$	${}_l[\![\psi]\!]^{i+1}_k$ if $i < k$
	${}_l[\![\psi]\!]^{l+1}_k$ otherwise
$F\psi$	$\exists j \in \{i,\ldots,k\}.{}_l[\![\psi]\!]^j_k$ if $i < l$
	$\exists j \in \{l,\ldots,k\}.{}_l[\![\psi]\!]^j_k$ otherwise
$G\psi$	$\forall j \in \{i,\ldots,k\}.{}_l[\![\psi]\!]^j_k$ if $i < l$
	$\forall j \in \{l,\ldots,k\}.{}_l[\![\psi]\!]^j_k$ otherwise
$\psi_1 U \psi_2$	$\exists j \in \{i,\ldots,k\} \forall n \in \{i,\ldots,j-1\}.({}_l[\![\psi_2]\!]^j_k \wedge$ ${}_l[\![\psi_1]\!]^n_k) \vee \exists j \in \{l,\ldots,k\} \forall m \in \{l,\ldots,j-1\} \forall n \in \{i,\ldots,k\}.({}_l[\![\psi_2]\!]^j_k \wedge {}_l[\![\psi_1]\!]^m_k \wedge {}_l[\![\psi_1]\!]^n_k)$

Table 5. The Checking results with different bounds

Bound (n)	Time(sec)	Memory(MB)
100	1.032	9.5
200	8.8989	33.3
300	12.289	45.9
400	22.777	73.5

Example 2 (Model checking of alternative blink of lights in Example 1)**.** *A desired property of the constraints in Example 1 is that green light and red light blink alternatively. We formalize the property as an LTL formula* $\psi_{blink} \triangleq G((green \Longrightarrow X\, red) \wedge (red \Longrightarrow X\, green))$. *Let* Φ_{E1} *be the set of the three constraints in Example 1. We check the satisfiability of the formula* ${}^n_l[\![\Phi_{E1}]\!]_n \wedge \neg\,{}_l[\![\psi]\!]^1_n$ *with a fixed bound* n *using Z3. The returned result is* `unsat`*, meaning that the formula is not satisfiable. The property is verified.*

Table 5 shows the cost time and memory with different bound. All the experiments are conducted on a Windows10 desktop with an Intel Core CPU (i5-5200U, 2.2GHz) and 4GB memory.

5. Case Studies

In this section, we present two case studies to evaluate the effectiveness and efficiency of the proposed approach.

5.1 Case I: Traffic Light Controller

The first case is about the traffic light controller [9], installed in intersections of North-South (N-S) highway and East-West (E-W) farm road, as shown in Figure 6. The N-S road is the main road, and is given the green light unless a sensor on the E-W street is activated. When activation occurs and the N-S light is green for sufficient time, the light changes to give way to the E-W traffic. The design also takes into account emergency vehicles which can activate an emergency sensor. When the emergency sensor is activated, both the N-S and E-W lights are turned red.

State transitions of the two controllers are modeled as two finite state machines (FSM), which are specified in CCSL as clock constraints. In CCSL, we can use the concept of schedule and build constraints to describe FSM. i.e., the states and transition conditions of FSM can be mapped to CCSL logical clocks. For instance, N-S light in state *GREEN* is mapped to a logical clock named

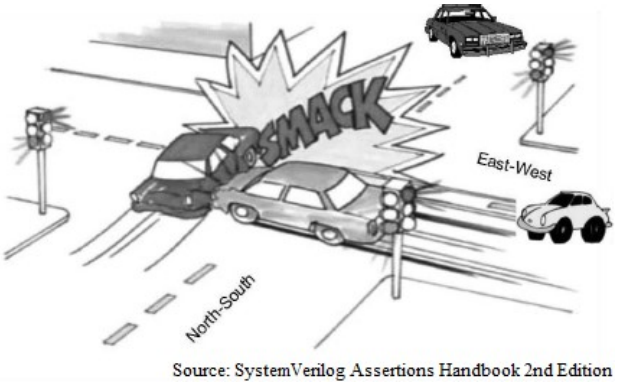
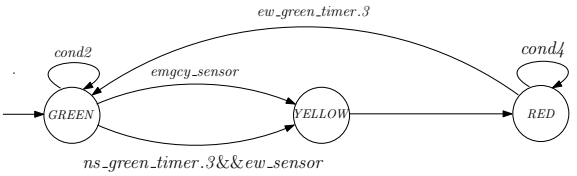
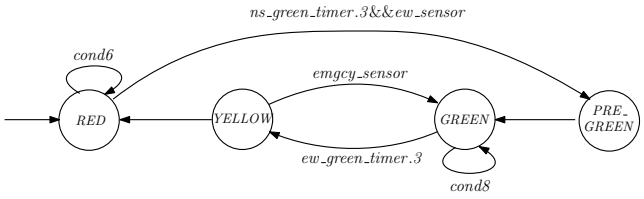


Figure 6. North-South and East-West Intersection.



1. $ns\_light.GREEN \ \# \ ns\_light.YELLOW$
2. $ns\_light.GREEN \ \# \ ns\_light.RED$
3. $ns\_light.YELLOW \ \# \ ns\_light.RED$
4. $cond1 \triangleq ns\_green\_timer.3 * ew\_sensor + emgcy\_sensor$
5. $cond1 \ \# \ cond2, clk \triangleq cond1 + cond2$
6. $cond1 * ns\_light.GREEN \ \$ \ 1 \ on \ clk \subseteq ns\_light.YELLOW$
7. $cond2 * ns\_light.GREEN \ \$ \ 1 \ on \ clk \subseteq ns\_light.GREEN$
8. $ns\_light.YELLOW \ \$ \ 1 \ on \ clk \subseteq ns\_light.RED$
9. $ew\_green\_timer.3 \ \# \ cond4, clk \triangleq ew\_green\_timer.3 + cond4$
10. $ew\_green\_timer.3 * ns\_light.RED \ \$ \ 1 \ on \ clk \subseteq ns\_light.GREEN$
11. $cond4 * ns\_light.RED \ \$ \ 1 \ on \ clk \subseteq ns\_light.RED$

(a) FSM and CCSL constraints of N-S controller



1. $ew\_light.GREEN \ \# \ ew\_light.YELLOW$
2. $ew\_light.GREEN \ \# \ ew\_light.RED$
3. $ew\_light.GREEN \ \# \ ew\_light.PRE\_GREEN$
4. $ew\_light.YELLOW \ \# \ ew\_light.RED$
5. $ew\_light.YELLOW \ \# \ ew\_light.PRE\_GREEN$
6. $ew\_light.RED \ \# \ ew\_light.PRE\_GREEN$
7. $cond5 \triangleq ns\_green\_timer.3 * ew\_sensor$
8. $cond5 \ \# \ cond6, clk = cond5 + cond6$
9. $cond5 * ew\_light.RED \ \$ \ 1 \ on \ clk \subseteq ew\_light.PRE\_GREEN$
10. $cond6 * ew\_light.RED \ \$ \ 1 \ on \ clk \subseteq ew\_light.RED$
11. $ew\_light.PRE\_GREEN \ \$ \ 1 \ on \ clk \subseteq ew\_light.GREEN$
12. $cond7 \triangleq ew\_green\_timer.3 + emgcy\_sensor$
13. $cond7 \ \# \ cond8, clk = cond7 + cond8$
14. $cond7 * ew\_light.GREEN \ \$ \ 1 \ on \ clk \subseteq ew\_light.YELLOW$
15. $cond8 * ew\_light.GREEN \ \$ \ 1 \ on \ clk \subseteq ew\_light.GREEN$
16. $ew\_light.YELLOW \ \$ \ 1 \ on \ clk \subseteq ew\_light.RED$

(b) FSM and CCSL constraints of E-W controller

Figure 7. Finite state machines and their corresponding encoding into CCSL constraints of N-S and E-W controllers

$ns\_light.GREEN$. The sensor of East-West road in the state trigger is mapped to ew_sensor. The emergency sensor after being triggered is mapped to $emgcy\_sensor$. Figure 7(a) and 7(b) depict the two FSMs of the N-S and E-W controllers and their corresponding CCSL constraints, respectively. We use complex constraints such as the forth and sixth ones in Figure 7(a) to make the constraints succinct. They can be represented by primitive constraints

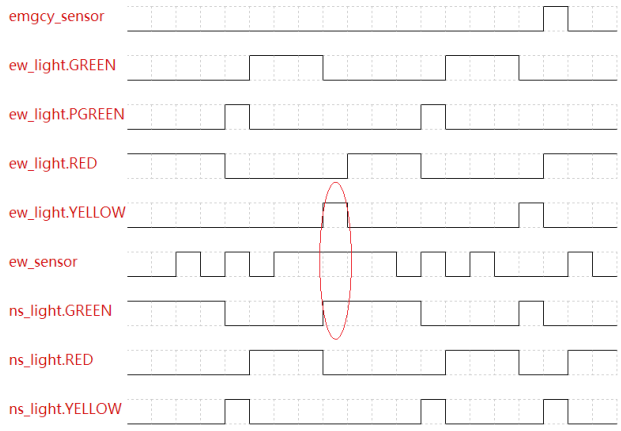


Figure 8. A counterexample of Property II within bound 20

with some intermediate clocks. It is also worth mentioning that we declare a *root* clock clk which ticks at each step. The purpose of the root clock is to express physical meaning of logical clocks by measuring their duration time between two immediate ticks. The constraint on clk is defined as a formula $\forall n \in \mathbb{N}^+.clk \in \delta(n)$. Two time counters *ns_green_timer* and *ew_green_timer* are used to count the duration of the lights when they are green. For instance, N-S light in the state *GREEN* during more than three lengths is mapped to $ns\_green\_timer.3$. More precisely, for each d in $\mathbb{N}^+$, $ns\_green\_timer.d$ is defined as follow:

$$\forall n \in \mathbb{N}^+.ns\_green\_timer.d \in \delta(n) \iff$$
$$\chi(ns\_light.GREEN, n+1) =$$
$$\chi(ns\_light.GREEN, n+1-d) + d$$

We extract the requirements of the system from the textual description in [9], and identify three temporal properties that the system should satisfy. We formalize the three properties as LTL formulas and model check them using the approach proposed in Section 4.

Property I: The North-South Highway traffic light and the East-West farm traffic lights must not be green simultaneously at any moment. The property can be formalized as the following LTL formula:

$$G(\neg(ns\_light.GREEN \wedge ew\_light.GREEN)).$$

Property II: To avoid accidents, the highway light must not be green when the farm road light is yellow. The property is formalized as the following LTL formula:

$$G(ns\_light.GREEN \implies \neg\, ew\_light.YELLOW).$$

Property III: Whenever the highway light is green, it must not become red in the next clock cycle. The following LTL formula is defined to specify the property:

$$G(ns\_light.GREEN \implies X(\neg\, ns\_light.RED)).$$

To model check the three properties, we transform the CCSL constraints shown in Figure 7(a) and 7(b) together with the negation of the LTL formula of each property into SMT formulas and use Z3 to check the satisfiability of the generated SMT formulas. We repeat model checking within different bounds. Z3 always returns `unsat` with Property I and III, meaning that the two properties are satisfied by those periodic schedules of the designed system.

For Property II, Z3 returns a solution when the bound is set 20, meaning that the generated SMT formulas are satisfiable and there exists a schedule that does not satisfy the property. The solution

Table 6. Model checking results in Case I (Time: sec)

Bound	Property I		Property II		Property III	
	Result	Time	Result	Time	Result	Time
10	✓	0.813	✓	0.661	✓	0.323
20	✓	2.270	✗	1.935	✓	0.968
40	✓	10.742	✗	7.794	✓	3.902
60	✓	46.711	✗	23.214	✓	11.146
80	✓	78.598	✗	31.011	✓	22.968

is an witness to the violation against the property by the designed system and is called a counterexample of the property. Figure 8 depicts the counterexample. The reason for the occurrence of the example is that the conditions that N-S light turns green and E-W light turns yellow are the same, i.e., after E-W light is green for three seconds. The result coincides with the faulty in the design, as pointed out in [9].

Table 6 shows the model checking results and the time cost by model checking the three properties with different bounds. When the bound is not greater than 80, verification is done in minutes.

5.2 Case II: Power Window System

The second case is about a power window system introduced in [7]. The system is composed of a switch, a controller and an electro-mechanical subsystem, as shown in Figure 9. The controller is in charge of interpreting both the actions of the user on the switch and the signal from the electro-mechanical subsystem to drive the motor to move the window. When the switch is pulled, a signal sUp is sent to the controller. On receiving the signal, the controller gets a signal $wEnd$ from the subsystem to check if the window is fully closed. If the window is not fully closed, the controller sends a signal wUp to the motor to raise up the window. The system also supports one touch mode, i.e., automatic raising of the window after a brief pull of the control switch.

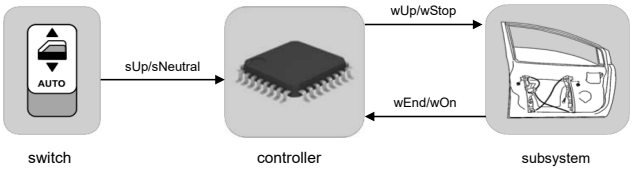


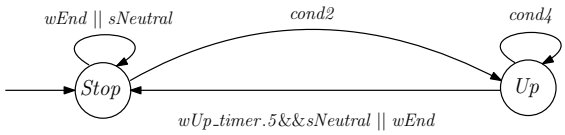
Figure 9. View of the Power Window Model

The behavior of this controller is formalized as a finite state machine as depicted by the upper part of Figure 10. To formalize one touch mode in FSM, we define a flag $wUp\_timer.5$ to represent that the window has been moved up for more than five units of time. If that is the case and $sNeutral$ is true, the window stops. The CCSL constraints used to specify the FSM are given in the lower part of Figure 10.

The one touch mode feature implies a timed behavior. That is, the motor is activated to automatically raise or lower the window after the control switch has been pulled during a given delay.

Table 7. Model checking results in Case II

Bound	100	200	500	1000
Result	✓	✓	✓	✓
Time(sec)	29.096	167.680	936.804	4685.006



1. $wUp \# wStop, sUp \# sNeutral, clk \triangleq sUp + sNeutral$
2. $wEnd \# wOn, clk \triangleq wEnd + wOn$
3. $cond1 \triangleq wEnd + sNeutral$
4. $cond1 \# cond2, clk \triangleq cond1 + cond2$
5. $cond1 * wStop \$ 1 \ on \ clk \subseteq wStop$
6. $cond2 * wStop \$ 1 \ on \ clk \subseteq wUp$
7. $cond3 \triangleq wEnd + wUp\_timer.5 * sNeutral$
8. $cond3 \# cond4, clk \triangleq cond3 + cond4$
9. $cond3 * wUp \$ 1 \ on \ clk \subseteq wStop$
10. $cond4 * wUp \$ 1 \ on \ clk \subseteq wUp$

Figure 10. The FSM and its corresponding CCSL constraints

Another requirement for the system is that the controller should stop moving the window after the window is closed completely. The requirement is formalized as an LTL formula as follows:

$$G(wEnd \Longrightarrow X \ wStop).$$

Because the formula contains global operator G, we transform the property into an SMT formula with respect to bounded periodic schedules as defined in Table 4. We then model check the generated formula together with those that are transformed from the constraints shown in Figure 10. Z3 always returns `unsat` within different bounds, meaning that the requirement is satisfied by the periodic schedules that are found within 1000 steps.

Table 7 shows the model checking result and the time cost by model checking the property. It is shown that with the increase of bound, more time is needed for model checking. When the bound is 1000, it takes approximately one hour and a quarter to finish. However, for model checking of periodic schedules we usually do not need a large bound like 1000 because in general a periodic schedule starts its first period soon after the first step.

6. Related Work

Schedulability is a basic and central property of CCSL constraints. However, to our knowledge the schedulability problem of CCSL is still an open problem, and no decision procedure has been proposed for the problem, let alone further formal analysis of other properties which depends on the schedulability of CCSL constraints. In existing approaches to the formal analysis of CCSL, either only subsets of CCSL constraints are considered such as the work [2, 18, 20, 21], or semi-algorithms are proposed with no guarantee for termination [16]. Frédéric et. al also mentioned encoding CCSL into a subclass of counter automata and performing reachability analysis using acceleration techniques [3]. The idea seems promising, but no concrete approach has been proposed.

Like existing approaches, although it neither fully solves the schedulability problem by transforming CCSL constraints into SMT formulas, two features of the SMT-based approach make it still useful in practice and complementary to existing ones. One feature is its support to model checking of temporal properties of CCSL. Temporal properties are of importance in practice to measure if a design satisfies requirements. The other feature is exploiting advantage of the efficiency of state-of-the-art SMT solvers in model checking of CCSL constraints. We did not compare the

efficiency of our approach with other approaches because of the lack of benchmarks. Nevertheless, it is well known that SMT-based approaches have been proved effective to tackle the notorious state-explosion problem in model checking in many works, e.g. [6]. Our experimental results also demonstrate that the properties in our two case studies can be model-checked in minutes. One piece of our future work is to compare the efficiency of the SMT-based approach with those of existing ones and evaluate the scalability of the approach by doing more complicated case studies.

There are existing works on model checking LTL properties using SAT solvers [13]. In particular, our approach to encoding LTL formulas into SMT formulas is inspired by the work [24] and [8]. In their encoding approaches, the underlying model is finite-state machine which is encoded into propositional formula. They also consider two cases of transforming LTL formulas into propositional formulas. One is for those paths which do not have loops, and the other is for those paths which contain loops. The two cases correspond to our encoding approaches for the case of non-periodic schedules and periodic schedules respectively. The difference is that in their approach LTL formulas are transformed into propositional formulas which are decidable, while in our approach they are transformed into SMT formulas in UFLIA logic.

7. Conclusion and Future Work

We have proposed an SMT-based approach to model checking LTL properties of CCSL constraints by transforming both CCSL constraints and LTL formulas into SMT formulas. Due to the undecidability of the satisfiability problem of generated SMT formulas in UFLIA logic, bounds are needed to do model checking. To achieve model checking of LTL properties with no limitation on boundedness of schedules, we propose a new encoding approach from LTL formulas into SMT formulas to model check LTL properties of periodic schedules in terms of model checking bounded periodic schedules. We developed a prototype tool based on the proposed approach, and conducted studies on two classical cases that are used as examples in the design of real-time and embedded systems. Experimental results demonstrate the usefulness and effectiveness of the proposed approach.

There are some other potential applications of the SMT-based approach to formal analysis of CCSL constraints. One is to *prove* but not to check properties of CCSL constraints in that no bound is required. Although without a bound SMT solvers may not terminate or terminate due to timeout or running out of memory, in some cases SMT solvers may return result to prove of falsify a property. This can be considered as *theorem proving*, an complementary verification approach to model checking. For this purpose, we may choose another SMT solver CVC4 in our further work because CVC4 has a better performance over Z3 for the satisfiability problem of the formulas in UFLIA logic [10].

Another potential application of the SMT-based approach is deadlock detection, i.e., to detect the existence of a schedule which prevents all clocks from ticking after reaching some step. We only need to encode deadlock into an SMT formula based on the encoding of CCSL constraints into SMT formulas. To unify all those applications, an integrated tool is desired to be developed, which is powered by existing efficient SMT solvers using the SMT-based approach that is proposed in this paper to support various formal analysis of CCSL constraints.

References

[1] C. André. *Syntax and semantics of the clock constraint specification language (CCSL)*. PhD thesis, INRIA, 2009.

[2] C. André and F. Mallet. Specification and verification of time requirements with CCSL and Esterel. In *Proceedings of the LCTES 2009*, volume 44 of *ACM Sigplan Notices*, pages 167–176. ACM, 2009.

[3] S. Bardin, A. Finkel, and J. Leroux. FASTer acceleration of counter automata in practice. In *Proceedings of the 10th TACAS*, volume 2988 of *LNCS*, pages 576–590. Springer, 2004.

[4] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli. CVC4. In *Proceedings of the 23rd CAV*, volume 6806 of *LNCS*, pages 171–177. Springer, 2011.

[5] C. Barrett, P. Fontaine, and C. Tinelli. The SMT-LIB standard: Version 2.5. *SMT-LIB. org*, 2016.

[6] A. Biere, A. Cimatti, E. M. Clarke, et al. Bounded model checking. *Advances in computers*, 58:117–148, 2003.

[7] F. Boulanger, A. Dogui, C. Hardebolle, C. Jacquet, D. Marcadet, and I. Prodan. Semantic adaptation using CCSL clock constraints. In *Proceedings of MODELS 2011*, volume 7167 of *LNCS*, pages 104–118. Springer, 2011.

[8] A. Cimatti, M. Pistore, M. Roveri, and R. Sebastiani. Improving the encoding of LTL model checking into SAT. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 196–207. Springer, 2002.

[9] B. Cohen, S. Venkataramanan, and A. Kumari. *SystemVerilog Assertions Handbook:–for Formal and Dynamic Verification*. VhdlCohen Publishing, 2005.

[10] D. R. Cok, D. Déharbe, and T. Weber. The 2014 SMT competition. *Journal on Satisfiability, Boolean Modeling and Computation*, 9:207–242, 2016.

[11] L. M. de Moura and N. Bjørner. Z3: an efficient SMT solver. In *Proceedings of the 14th TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[12] J. DeAntoni and F. Mallet. Timesquare: Treat your models with logical time. In *Proceedings of TOOLS 2012*, volume 7304 of *LNCS*, pages 34–41. Springer, 2012.

[13] C. P. Gomes, H. Kautz, A. Sabharwal, and B. Selman. Satisfiability solvers. *Foundations of Artificial Intelligence*, 3:89–134, 2008.

[14] E.-Y. Kang and P.-Y. Schobbens. Schedulability analysis support for automotive systems: from requirement to implementation. In *Proceedings of the 29th SAC*, pages 1080–1085. ACM, 2014.

[15] F. Mallet. MARTE/CCSL for modeling Cyber-Physical Systems. In *Proceedings of the Formal Modeling and Verification of Cyber-Physical Systems*, pages 26–49. Springer, 2015.

[16] F. Mallet and R. de Simone. Correctness issues on MARTE/ CCSL constraints. *Sci. Comp. Prog.*, 106:78–92, 2015.

[17] OMG. UML profile for MARTE: modeling and analysis of real-time embedded systems, 2015.

[18] J. Suryadevara, C. C. Seceleanu, F. Mallet, et al. Verifying MARTE/CCSL mode behaviors using UPPAAL. In *Proceedings of the 11th SEFM*, volume 8137 of *LNCS*, pages 1–15. Springer, 2013.

[19] J. Vidal, F. De Lamotte, G. Gogniat, P. Soulard, and J.-P. Diguet. A co-design approach for embedded system modeling and code generation with UML and MARTE. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 226–231. European Design and Automation Association, 2009.

[20] L. Yin, F. Mallet, and J. Liu. Verification of MARTE/CCSL time requirements in Promela/SPIN. In *Proceedings of the 16th ICECCS*, pages 65–74. IEEE CS, 2011.

[21] H. Yu, J. Talpin, L. Besnard, T. Gautier, H. Marchand, and P. L. Guernic. Polychronous controller synthesis from MARTE/CCSL timing specifications. In *Proceedings of the 9th IEEE/ACM MEMOCODE*, pages 21–30. IEEE, 2011.

[22] M. Zhang and F. Mallet. An executable semantics of clock constraint specification language and its applications. In *Proceedings of the 4th FTSCS*, volume 596 of *CCIS*, pages 37–51. Springer, 2015.

[23] M. Zhang, F. Mallet, and H. Zhu. An SMT-based approach to the formal analysis of MARTE/CCSL. In *Proceedings of the 18th ICFEM*, volume 10009 of *LNCS*, pages 433–449. Springer, 2016.

[24] W. Zhang. SAT-based verification of LTL formulas. In *Proceedings of FMICS 2006*, volume 4346 of *LNCS*, pages 277–292. Springer, 2006.