Divergence Detection for CCSL Specification via Clock Causality Chain

Qingguo Xu[1,2]([✉]), Robert de Simone[3], and Julien DeAntoni[3]

[1] School of Computer Engineering and Science, Shanghai University,
No. 99, Shangda Road, Shanghai 200444, China
qgxu@mail.shu.edu.cn
[2] Shanghai Key Laboratory of Computer Software Testing and Evaluating,
Shanghai 201114, China
[3] Inria Sophia Antipolis Méditerranée AOSTE,
University of Nice Sophia Antipolis, I3S, 06902 Sophia Antipolis Cedex, France
Robert.de_Simone@inria.fr,
Julien.Deantoni@polytech.unice.fr

Abstract. The Clock Constraint Specification Language (CCSL), first introduced as a companion language for Modeling and Analysis of Real-Time and Embedded systems (MARTE), has now evolved beyond the time specification of MARTE, and has become a full-fledged domain specific modeling language widely used in many domains. A CCSL specification is a set of constraints, which symbolically represents a set of valid clock schedules, where a schedule represents the order of the actions in a system. This paper proposes an algorithm to detect the divergence behavior in the schedules that satisfy a given CCSL specification (*i.e.* it proposes to detect the presence of infinite but non periodic schedules in a CCSL specification). We investigate the divergence by constructing causality chains among the clocks resulting from the constraints of the specification. Depending on cycles in the causality chains, a bounded clock set built by our proposed algorithm can be used to decide whether the given specification is divergence-freedom or not. The approach is illustrated on one example for architecture-driven analysis.

Keywords: CCSL · Divergence · Clock causality chain · Bounded Clock Set · PVS

1 Introduction

The Unified Modeling Language (UML) Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) [1], adopted in November 2009, has introduced a *Time Model* [2] that extends the informal Simple Time of UML2. This time model is general enough to support different forms of time (discrete or dense,

chronometric or logical). Its so-called clocks allow enforcing as well as observing the occurrences of events and the behavior of annotated UML elements. The *Time Model* comes with a companion language named the Clock Constraint Specification Language (CCSL) [3] defined in the annex of the MARTE specification. Initially devised as a language for expressing constraints between clocks of a MARTE model, CCSL has evolved and has been developed independently of the UML. CCSL is now equipped with a formal semantics [3] and is supported by a software environment (TimeSquare [4]) that allows for the specification, solving, and visualization of clock constraints.

MARTE promises a general modeling framework to design and analyze systems. Lots of works have been published on the modeling capabilities offered by MARTE, much less on verification techniques supported. Inspired by the works about state-based semantics interpretation for the kernel CCSL operators [5], this paper focuses on the **divergence** (see Sect. 3.1) detection of some CCSL specifications. This issue was addressed by [6, 7] but their propositions were applying parallel composition of individual CCSL constraints, making the propositions unsuitable for industrial size systems. In this work, we significantly reduce the complexity of the divergence detection by constructing and analyzing clock causality chains. Additionally our algorithm can point out what constraint can be added to make the specification divergence-free. In order to acquire clock causality chains, we first highlighted some interesting properties about causal relation between clocks. Furthermore, we propose an algorithm to decide if a given CCSL is divergence-free or not, by constructing the proposed a "Bounded Clock Set" (BCS) based on clock delay expression as well as the causality relation between clocks.

Section 2 introduces a state-transition based semantics for CCSL. Section 3 shows how to detect the divergence and make sure the specification is divergence-freedom based on the notion of divergence of CCSL specifications. Also, an algorithm, which is used to build the bounded clock set for determining the convergence, is depicted in this section. Section 4 illustrates, by using an example from architecture-driven analysis, the use of our algorithm on a CCSL specification. It shows how to improve a divergent specification such that it becomes a convergent one by adding clock constraints hinted by the algorithm. Finally, Sect. 5 makes a comparison with related works and Sect. 6 concludes the contribution and outlines some future works.

2   Preliminaries

This section briefly introduces the logical time model [2] of MARTE and the Clock Constraint Specification Language (CCSL). A technical report [3] and it latest update [8] describes the syntax and the semantics of a kernel set of CCSL constraints. We describe in this paper only the constraints that are used for our discussion.

The notion of multiform logical time has first been used in the theory of synchronous languages [9] and its polychronous extensions. CCSL is a formal declarative language to specify polychronous clock specification. It provides a concrete syntax to make the clocks and clocks constraints first-class citizens of UML-like models. Clocks in CCSL are used to measure the number of occurrences of events in a system. Logical clocks replace physical times by a logical sequencing. A CCSL specification do not

need for clocks to be relative to a global physical time. They are by default independent of each other and what matter is the partial ordering of their ticks induces by the constraints between them.

A clock belongs to a clock set $\mathcal{C}$. During the execution of a system, an execution step is defined and at a given step, every clock in $\mathcal{C}$ can tick or not according to the constraints defined in the specification. A schedule captures what happens during one particular execution.

Definition 1 (Schedule): A *schedule* is defined as a function *Sched*: $\mathbb{N}_{>0} \rightarrow 2^{\mathcal{C}}$. ∎

Given an execution step $i \in \mathbb{N}_{>0}$, and a schedule $\sigma \in$ *Sched*, $\sigma(i)$ denotes the set of clocks that tick at step i.

For a given schedule, the configurations of the clocks tell us the advance of the clocks, relative to the others.

Definition 2 (Clock Configuration): For a given schedule σ, clock $c \in \mathcal{C}$ and a natural number $n \in \mathbb{N}$, the configuration χ_σ: $\mathcal{C} \times \mathbb{N} \rightarrow \mathbb{N}$ is defined recursively as:

$$\chi_\sigma(c, n) = \begin{cases} 0, & \text{if } n = 0 \\ \chi_\sigma(c, n-1), & \text{if } c \notin \sigma(n) \\ \chi_\sigma(c, n-1) + 1, & \text{if } c \in \sigma(n) \end{cases} \quad \text{(F.1)}$$

∎

For a clock $c \in$, and a step $n \in \mathbb{N}$, $\chi_\sigma(c, n)$ counts the number of times the clock c has ticked at step n for the given schedule σ.

CCSL is used to specify a set of valid schedules. There is usually more than one valid schedule that satisfies a given specification. If there is no satisfying schedule, then we say that the specification is ill-formed. The detail properties about the schedules against the given specification is investigated in [10]. Clocks can be finite of infinite. Since divergence problem only makes sense on infinite clocks, we do not care about constraints that make clock terminating (see [8] for details). Consequently, every clock in $\mathcal{C}$ are infinite (will never terminate), i.e., $\forall c \in \mathcal{C}$, $\chi_\sigma(c, n)$ is boundless with n increasing.

Definition 3 (CCSL Specification): A CCSL specification $\mathcal{SPEC}$ is a pair <$\mathcal{C}$, *CConstr*> , where $\mathcal{C}$ is a set of clocks, *CConstr* is a set of formulae (see Definition 5) used to specify the relations among the clocks in the set $\mathcal{C}$. ∎

Definition 4 (Clock Set) An element in the clock set $\mathcal{C}$ can be given by the specification writer explicitly (**explicit clock**), or by one of the following clock expressions (**implicit clock**):

$$Clock := a + b | a * b | a \backslash b | sup(a, b) | inf(a, b) | a\$n | SampledOn(a, b) | FilteredBy(a, u, v) \quad \text{(F.2)}$$

where $a, b \in \mathcal{C}$ are clocks, $u, v \in (0 + 1)^*$ are finite binary words, and $n \in \mathbb{N}$ is a natural number. ∎

Once we write one clock expression in the form of (F.2), a new clock is created and added into the clock set $\mathcal{C}$. For example, if we give the explicit clock set $\{a, b, c\}$, and clock expression set $\{d = a + b, e = c \$1\}$, then the considered clock set is $\mathcal{C} = \{a, b, c, d, e\}$. Note that there may not be any given name for the implicit clock of the clock expression (*e.g.* if it occurs in one clock relation) (see Definition 5). It should be noted that the new clock will be scheduled depending on the clock(s) that occur in that expression.

It is convenient to define a clock as an Abstract Data Type (ADT) [11] in Prototype Verification System (PVS) [12, 13] if we treat the clock expression as an element in a clock set instead of assigning them a new clock name.

The CCSL constraint defined over the clock set includes both the **explicit** clocks and **implicit** ones.

Definition 5 (CCSL Relation): For a given clock set $\mathcal{C}$, the corresponding clock relation set $\Phi(\mathcal{C})$ over $\mathcal{C}$ is defined recursively as:

$\psi \coloneqq a \prec b \mid a \preccurlyeq b \mid a \subset b \mid a \# b \mid \psi \wedge \psi$
where $a, b \in \mathcal{C}$. ∎

Every clock relation in the set $\Phi(\mathcal{C})$, is a primitive formula that relates a clock pair or their conjunction.

Definition 6 (CCSL Specification Satisfaction). For a given CCSL specification $\mathcal{SPEC} = <\mathcal{C}, CConstr>$, A schedule σ over $\mathcal{C}$ satisfies $\mathcal{SPEC}$, denoted $\sigma \vDash \mathcal{SPEC}$, if and only if for every formulae r in $CConstr$, σ evaluates to true according to the following definition postulated by the CCSL semantics:

$\sigma \vDash r$ *if and only if cases r's form of*

$$
\begin{array}{lll}
a \prec b : \forall n \in \mathbb{N}, & \chi_\sigma(a, n) = \chi_\sigma(b, n) \Rightarrow b \notin \sigma(n+1) & (\textbf{Precedence}) \\
a \preccurlyeq b : \forall n \in \mathbb{N}, & \chi_\sigma(a, n) \geq \chi_\sigma(b, n) & (\textbf{Causality}) \\
a \subset b : \forall n \in \mathbb{N}_{>0}, & a \in \sigma(n) \Rightarrow b \in \sigma(n) & (\textbf{Subclock}) \\
a \# b : \forall n \in \mathbb{N}_{>0}, & a \notin \sigma(n) \vee b \notin \sigma(n) & (\textbf{Exclusion}) \\
\psi_1 \wedge \psi_2 : \sigma \vDash \psi_1 \wedge \sigma \vDash \psi_2 & & (\textbf{Conjunction})
\end{array}
$$
$$(\text{F.3})$$

where $a, b \in \mathcal{C}$. ∎

It's straightforward to prove that both *Causality* and *Subclock* are pre-orders on $\mathcal{C}$, i.e., they are reflexive and transitive. For simplicity, we can write $a \preccurlyeq b \preccurlyeq c$ for $a \preccurlyeq b \wedge b \preccurlyeq c$, and so do other transitive clock relation. The transitivity property of the *Causality* relation is of importance in determining the boundedness of a specification (see Sect. 3.2). It is also straightforward to prove that *Exclusion* is neither reflexive nor transitive. The transitive property of *Precedence* is very tedious to prove from this form of definition, although it is obvious in other semantics model [8].

The *implicit clocks* defined using clock expressions (F.2), are constrained according to the parameters of the clock expression. In other words, a clock expression is a clock generator where the output clock ticks or not according to the input clock(s) state and other arguments, if any.

Definition 7 (Clock Expression Satisfaction). Whether an implicit clock (denoted c in the following) can tick or not in a schedule σ, is determined by the behaviors of the input clock(s) in σ,

$\sigma \models c$ *if and only if cases c is defined by*

$$
\begin{array}{ll}
a + b : \forall n \in \mathbb{N}_{>0}, c \in \sigma(n) \text{iff} a \in \sigma(n) \vee b \in \sigma(n) & (\textbf{Union}) \\
a * b : \forall n \in \mathbb{N}_{>0}, c \in \sigma(n) \text{iff} a \in \sigma(n) \wedge b \in \sigma(n) & (\textbf{Intersection}) \\
a \backslash b : \forall n \in \mathbb{N}_{>0}, c \in \sigma(n) \text{iff} a \in \sigma(n) \wedge b \notin \sigma(n) & (\textbf{Minus}) \\
sup(a, b) : \forall n \in \mathbb{N}, \chi_\sigma(c, n) = \min(\chi_\sigma(a, n), \chi_\sigma(b, n)) & (\textbf{Supremum}) \\
inf(a, b) : \forall n \in \mathbb{N}, \chi_\sigma(c, n) = \max(\chi_\sigma(a, n), \chi_\sigma(b, n)) & (\textbf{Infimum}) \\
a\$d : \forall n \in \mathbb{N}, \chi_\sigma(c, n) = \max(\chi_\sigma(a, n) - d, 0) & (\textbf{Delay})
\end{array}
\qquad \text{(F.4)}
$$

SampledOn(a, b): $\forall n \in \mathbb{N}_{>0}, c \in \sigma(n)$*iff*

$$(b \in \sigma(n) \wedge (\exists j \in [1..n], a \in \sigma(j) \wedge \forall m : [j..n-1], b \notin \sigma(m))) \quad (\textbf{SampledOn})$$

FilteredBy (a, u, v): $\forall n \in \mathbb{N}_{>0}, c \in \sigma(n)$*iff*

$$(a \in \sigma(n) \wedge (if k \leq |u| \text{then } u[k] = 1 \text{ else } v[(k - |u|) mod |v|] = 1), where\, k = \mathcal{X}_\sigma(a, n))$$
$$(\textbf{FilteredBy})$$

where $a, b \in \mathcal{C}, u, v \in (0 + 1)^*$, and $d \in \mathbb{N}_{>0}$. $|u|$ (resp. $|v|$) represents the length of binary word u (resp. v), $k = \chi_\sigma(a, n)$ is the number of tick of clock a from step 1 to n. ∎

By composing the relations and the expressions provided in Definitions 4 and 5, some user-defined clock relations can be further defined, as below:

$$a \sim b := a \prec b \prec a \$ 1 \quad (\textbf{Alternation}) \qquad \text{(F.5)}$$

$$a \prec_n b := a \prec b \prec a \$ n \quad (\textbf{Bounded precedence}) \qquad \text{(F.6)}$$

$$a \equiv b := a \preccurlyeq b \wedge b \preccurlyeq a \quad (\textbf{Coincidence}) \qquad \text{(F.7)}$$

Obviously, *Alternation* (F.5) is a special case of *bounded precedence* (F.6). *Alternation* (which is frequently used in the CCSL specifications), is a kind of bounded relation that is discussed in detail in Sect. 3.

From the definitions above, some proved propositions can be listed below. The reader who is interested the proof can get the details in the report [5].

Proposition 1 (Precedence Implies Causality). The Precedence is a stronger form of causality:

$\sigma \models a \prec b \Longrightarrow \sigma \models a \preccurlyeq b$ ∎

Proposition 2 (Subclock Implies Causality). When a is a *Subclock* of b, then b is faster than a:

$$\sigma \vDash a \subset b \Longrightarrow \sigma \vDash b \preccurlyeq a \ \blacksquare$$

Proposition 3 (Union and Causality). The union of two clocks is faster than both clocks:

$$\sigma \vDash u \coloneqq a + b \Longrightarrow \sigma \vDash u \preccurlyeq a \wedge \sigma \vDash u \preccurlyeq b \ \blacksquare$$

Proposition 4 (Intersection and Causality). The intersection of two clocks is slower than both clocks:

$$\sigma \vDash i \coloneqq a * b \Longrightarrow \sigma \vDash a \preccurlyeq i \wedge \sigma \vDash b \preccurlyeq i \ \blacksquare$$

Proposition 5 (Infimum and Causality). The infimum of two clocks is always faster than both clocks:

$$\sigma \vDash f \coloneqq inf(a, \ b) \Longrightarrow \sigma \vDash f \preccurlyeq a \wedge \sigma \vDash f \preccurlyeq b \ \blacksquare$$

Proposition 6 (Supremum and Causality). The supremum of two clocks is always slower than both clocks:

$$\sigma \vDash s \coloneqq sup(a, \ b) \Longrightarrow \sigma \vDash a \preccurlyeq s \wedge \sigma \vDash b \preccurlyeq s \ \blacksquare$$

Proposition 7 (Delay and Causality). The delayed clock is always slower than the base clock:

$$\sigma \vDash c \coloneqq a \ \$ \ d \Longrightarrow \sigma \vDash a \preccurlyeq c \ \blacksquare$$

Propositions 1 to 7 defines new implicit *Causality* relations from all the other relations (*Precedence* and *Subclock*) and expressions (*Union, Intersection, Infimum, Supremum* and *Delay*). The Definitions 1 to 7 have been formalized in PVS, and the Propositions 1 to 7 have been proved (in most cases by induction).

Based on the definitions of *SampledOn* and *FilteredBy* in (F.3), $c \coloneqq SampledOn(a, b)$ implies $a \preccurlyeq c$ and $c \subset a$, $c \coloneqq$ FilteredBy(a, u, v) implies $c \subset a$. Hence, some *Causality* relations between the new implicit clock and their input clock can indirectly be deduced from the clock expression *SampledOn* and *FilteredBy*. Therefore, we can also get *Causality* relations using *SampledOn* and *FilteredBy*, besides using Propositions 1 to 7.

3 Divergence Detection

3.1 Divergence and Bounded Relation

Let's consider a very simple CCSL specification $\mathcal{SPEC}_1 = <\{sending, ack\}, \{sending \preccurlyeq ack\}>$. It is **divergent** as we don't know what will happen at a certain execution step i in a schedule σ provided that σ satisfies $\mathcal{SPEC}_1$ from step 1 to step $i - 1$. That is to say, there are many possibilities of clock ticking (tick *sending*, tick *ack*, or both) to ensure the satisfaction of $\mathcal{SPEC}_1$ at step i. Furtherly, whether the clock *ack* ticks or not is uncontrollable and unpredicatable since there is nothing to trigger its firing. Therefore, some expected properties may eventually not be guaranteed, if it is implied by the implementation of *ack*'s tick.

Definition 8 (Divergent Specification). We say a CCSL specification $\mathcal{SPEC} = <\mathcal{C},$ $CConstr>$ is divergent, if an expected clock in $\mathcal{C}$ has the possibility never ticks, formally,

$\exists\sigma, (\sigma \vDash \mathcal{SPEC} \land \exists g, r \in, \forall i:\{k: \mathbb{N}_{>0}| r \in \sigma(k)\}, \exists j> i, g \notin \sigma(j))$. ∎

Here g (means *goal*) is the expected action that may be an acknowledge signal/operation of other clock r (means *request*). That is to say, after performing some necessary operation(s) by ticking the some clocks in $\mathcal{C}\backslash\{g\}$ containing clock r, there must be an expected final result (ticking clock g) appears in the admissible future, but unfortunately only God knows g will tick or not in the divergent specification $\mathcal{SPEC}$. Therefore, the divergence of $\mathcal{SPEC}_\infty$ is asserted by choosing $g = ack$, $r = sending$ and the given schedule σ such that $\forall i \in \mathbb{N}_{>0}, \sigma(i) = \{sending\}$.

On the contrary, another example $\mathcal{SPEC}_\in = <\{a, b\}, \{a \sim b\}>$ is not divergent because the constraint $a \sim b$ postulates that the schedule that satisfies $\mathcal{SPEC}_\in$ must execute by alternating a with b forever. In this case, we say the behavior and the corresponding CCSL specification are **convergent** if it is also free of some unexpected properties, such as deadlock and livelock and so on.

The divergent specification behavior is not predicable in the sufficient future. It is possible for some expected action(s) to be delayed infinitely in the future. This *bad* behavior is unexpected since some actions can never happen after a certain simulation step. Therefore, a specification that contains divergence is unsafe. How to detect **divergence** existence in a given CCSL specification is the main subject in this section. Additionally, we provide some suggestions to modify the CCSL specification so that it becomes divergence-free.

For a given CCSL specification $<\mathcal{C}, CConstr>$, if the difference between the speeds of two clocks $a, b \in \mathcal{C}$ is limited in an allowed boundary, we say the clock pair (a, b) has a **bounded relation**.

Definition 9 (Bounded Relation). For a given clock set $\mathcal{C}$, two clocks $a, b \in \mathcal{C}$, and a schedule σ over $\mathcal{C}$, a and b has the bounded relation with a given boundary $m \in \mathbb{N}$, denotes $|a, b| \le m$:

$\sigma \vDash |a, b| \le m$ ***iff*** $\forall i \in \mathbb{N}_{>0}, \exists j \in \mathbb{N}_{>0}, | \mathcal{X}_\sigma(a, i) - \mathcal{X}_\sigma(b, j) | \le m$
m (resp. $- m$) can be called *upper* (resp. *lower) bound*. ∎

When such an unbounded clock pair is found, we say that there is **divergence** in the specification. We say the specification free of divergence is a bounded specification.

Definition 10 (Bounded Specification). For a given CCSL specification $\mathcal{SPEC} = <\mathcal{C},$ $CConstr>$, $\forall a, b \in \mathcal{C}$, $\mathcal{SPEC}$ is **bounded** if and only if any clock pair has the **bounded relation**:

$\forall\sigma, \sigma \vDash \mathcal{SPEC} \Longrightarrow \exists m \in \mathbb{N}, \sigma \vDash |a, b| \le m$ ∎

A bounded specification is divergence-free because there are no possible boundless drifts between any clock pair (*i.e.* between the actions).

If we check every clock pairs among all clocks in $\mathcal{C}$ to decide whether a specification is divergence-free or not, there are $\binom{|\mathcal{C}|}{2} = |\mathcal{C}| \times (|\mathcal{C}| - 1)/2$ pairs need to be

checked. The number of checks then totals to $(|\mathcal{C}|^2)$. But in practice many checks can be safely neglected when the bounded relation is implied by the already checked one.

The **Bounded Relation** can directly be derived from the formula for most of the constraints. Let us show how to get the **Bounded Relation** between two clocks implied by the existing clock relations and expressions.

Bounded Relation is an equivalence relation over $\mathcal{C}$, i.e., it is reflexive, symmetric and transitive. Note that they do not necessarily have the same boundaries for different bounded relations.

Proposition 8 (Bounded Extension). The **Bounded Relation** is transitive and the transitive resulting boundary is the sum of the original ones.

$|a, b| \leq m_1 \wedge |b, c| \leq m_2 \Longrightarrow |b, c| \leq m_1 + m_2$ ∎

Proposition 8 can be proved by using classical properties of inequalities addition. The proof is obvious and omitted here.

Proposition 9 (Bounded Restriction). Bounded relation can be restricted w.r.t. *Causality*, and the resulting boundary is not greater than the original one.

$|a, b| \leq m \wedge \forall c \in \mathcal{C}, (a \preccurlyeq c \preccurlyeq b \vee b \preccurlyeq c \preccurlyeq a) \Longrightarrow |a, c| \leq m \wedge |b, c| \leq m$ ∎

*Proof of **Proposition 9**:*
Let $maxDrift\_ab(n) = \chi_\sigma(a, n) - \chi_\sigma(b, n)$,
$maxDrift\_ac(n) = \chi_\sigma(a, n) - \chi_\sigma(c, n)$,
$maxDrift\_bc(n) = \chi_\sigma(b, n) - \chi_\sigma(c, n)$, for all $n \in \mathbb{N}$,
from the Definition 9, *we have*

$|a, b| \leq m \Longrightarrow |maxdrift\_ab(n)| \leq m \Longrightarrow -m \leq maxdrift\_ab(n) \leq m$, for all $n \in \mathbb{N}$,
Assume $a \preccurlyeq c \preccurlyeq b$, from the definition 6, we have $\forall n \in \mathbb{N}, \chi_\sigma(a, n) \geq \chi_\sigma(c, n) \geq \chi_\sigma(b, n)$, then $maxDrift\_ac(n) \geq 0 \geq -m$.

$$from\ maxdrift\_ab(n) \leq m \tag{1}$$

$$maxdrift\_bc(n) \leq 0 \tag{2}$$

(1) + (2) *gets*
$maxDrift\_ac(n) = maxdrift\_ab(n) + maxdrift\_bc(n) \leq 0 + m = m$
therefore, $|maxDrift\_ac(n)| \leq m$.
similarly, from (2) we have $\forall n \in \mathbb{N}, maxDrift\_bc(n) \leq 0 \leq m$,

– *$maxDrift\_bc(n) \leq maxdrift\_ab(n) \leq m \Longrightarrow maxDrift\_bc(n) \geq -m$*

There exists the similar proof under the cases $b \preccurlyeq c \preccurlyeq a$. ∎
From Definition 7, we can see *Delay* implies the *Bounded Relation*:

Proposition 10 (Delay Implies Bounded). The delayed clock a and the corresponding base clock b has the **Bounded Relation** with a lower bound 0 and a upper bound d:
$\sigma \models a := b \$ d \Longrightarrow \sigma \models |a, b| \leq d$ ∎

Proof of **Proposition 10**: This is direct conclusion from Definition 7.

The primitive clock relations and the other clock expressions except *Delay* and *FilteredBy* don't imply bounded relation. But their composition may deduce.

Proposition 11 (Alternate Implies Bounded). The clock pair involves in *Alternation relation* has the **bounded relation** with the boundary 1:

$$\sigma \models a \sim b \implies \sigma \models |a, b| \leq 1 \qquad \blacksquare$$

Proof of **Proposition 11**:

$a \sim b \Leftrightarrow$ by **Alternation** definition (F.5)

$a \preccurlyeq b \preccurlyeq a \ \$ \ 1 \implies$ *by* Proposition 1 (*Precedence Implies Causality*)

$a \preccurlyeq b \preccurlyeq a \ \$ \ 1 \implies$ *by* Proposition 10 (*Delay Implies Bounded*)

$|a, a \ \$ \ 1| \leq 1 \implies$ *by* Proposition 9 (*Bounded Restriction*)

$|a, b| \leq 1 \wedge |a \ \$ \ 1, b| \leq 1 \implies$ *by proposition calculus*

$|a, b| \leq 1 \qquad \blacksquare$

Similarly, one can prove *Bounded Precedence* (F.6) is also a **Bounded Relation**. *FilteredBy* expression $c \coloneqq a \ \blacktriangledown(u.v^{\omega})$, which defines a clock new clock c as a **Subclock** of a according to two binary words u and v, implies a *Bounded Relation* between clock pair (c, a) if there exists at least one 1-bit in the periodical pattern v.

Proposition 12 (FilteredBy may Imply Bounded). The clock pair (a, c) involves in the expression $c \coloneqq a \ \blacktriangledown(u.v^{\omega})$ has the **Bounded Relation** with the boundary $|u| + p \times (|v| - 1)$ if and only if $\exists i \in [1..|v|]$ such that $v[i] = 1$. Where p is the number of periodical patterns that have passed from the initial configuration. $\qquad \blacksquare$

Proof of **Proposition 12**:

Suppose a schedule σ against by $c \coloneqq a \ \blacktriangledown(u.v^{\omega})$, For some schedule step m, suppose clock a has ticked $|u| + |v|$ times from start, then

$$\chi_{\sigma}(a, m) = |u| + |v|$$

*By **FilteredBy** definition (F.4), during the process, clock a ticks at step j if and only if both a ticks and $u.(v)^{\omega}[j] = 1$. Because $\exists i \in [1..|v|]$, $v[i] = 1$, we get:*

$$1 \leq \chi_{\sigma}(c, m) \leq |u| + |v|$$

With the passage of schedule σ, $\chi_{\sigma}(c, n)$ will increase at least one while $\chi_{\sigma}(a, n)$ increase every $|u| + |v|$ after passing a periodical pattern. Therefore, when the schedule σ reaches step n by the time p periodical patterns has passed:

$\chi_{\sigma}(a, n) = |u| + p \times |v|$ and $p \leq \chi_{\sigma}(c, n) \leq |u| + p \times |v| \implies |\chi_{\sigma}(a, n) - \chi_{\sigma}(c, n)| \leq |u| + p \times (|v| - 1)$ $\qquad \blacksquare$

3.2   Clock Causality Chain and Bounded Clock Set

In order to determine whether a CCSL specification is divergent or not, according to Definition 9, one must show *every* clock pair has *Bounded Relation*. Due to most clock constraints without the bound information, we try to capture the *Bounded Relation* from the *Causality* relation.

Definition 11 (Clock Causality Chain). For a given clock set $\mathcal{C}$, some clocks in $\mathcal{C}$ may form a Clock Causality Chain (CCC), which is a finite sequence $c_1, c_2, ..., c_n$ such that $\forall i \in [1..n-1], c_i \preccurlyeq c_{i+1}$ and $n \geq 2$. It is called Bounded Clock Causality Chain (BCCC) if $\exists m \in \mathbb{N}, |c_1, c_n| \leq m$, and m is called the chain's boundary. It is an Unbounded CCC (UCCC) otherwise. ∎

Proposition 13 (BCCC Boundedness). ABCCC $\rho = c_1, c_2, ..., c_n$ with a boundary m implies that any two clocks in ρ has the **Bounded Relation**: $\forall i, j \in [1..n], |c_i, c_j| \leq m$. ∎

Proposition 13 is easily proved by using Proposition 9 **(Bounded Restriction)** as well as the *Causality* transition.

Due to Proposition 13, Checking $\binom{n}{2}$ clock pairs bounded relation is replaced by checking only one clock pair and additionally sorting n clocks with respect to *Causality* relation. Moreover, the *Causality* relation is much easier to get than Bounded Relation as the former is implied by some clock constraints (see Propositions 1–7).

Obviously, a specification is divergent if we can get a UCCC.

Theorem 1 (UCCC Implies Divergence). A CCSL specification $\mathcal{SPEC} = <\mathcal{C}, CConstr>$ is divergent, if there exists a UCCC $\rho = c_1, c_2, ..., c_n$ ($\forall i \in [1..n], c_i \in \mathcal{C}$) induced by the *CConstr*. ∎

Proof of Theorem 1:

ρ *is unbounded* $\Longrightarrow \forall i \in [1..n-1], c_i \preccurlyeq c_{i+1} \Longrightarrow$ *That c_n doesn't tick forever in some schedule* σ *has nothing about asserting the true value of* $\sigma \models \mathcal{SPEC}$.

We can find such a schedule σ *against* $\mathcal{SPEC}$ *that c_1 tick at least once.* σ *is indeed the witness* ($g = c_n$ *and* $r = c_1$) *for uncovering the divergence of* $\mathcal{SPEC}$ *according to* Definition 8. ∎

Let us illustrate Theorem 1 *on a toy example,* $\mathcal{SPEC}_s = <\mathcal{C}, CConstr>$, *where* $\mathcal{C} = \{a, b, sup(a, b), c\}$ *and* $CConstr = \{a \sim c, sup(a, b) \preccurlyeq c\}$. *Let* $sup = sup(a, b)$, *we get*

$|a, c| \leq 1$, by Proposition 11

$a \prec b$ *and* $a \preccurlyeq b$, *by* **Alternation** *definition and* Proposition 1.

$a \preccurlyeq sup \preccurlyeq c$ *and* $b \preccurlyeq sup$, *by* Proposition 6.

The Causality Clock Graph (CCG) [5] in Fig. 1 shows the *Causality* relations among clocks. The *Causality* line from clock a to c can be safely removed because of causality transition.
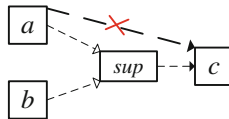

Fig. 1. CCG for $\mathcal{SPEC}_s$

The CCC a, sup, c is a BCCC, while the CCC $\rho = b, sup$ is a UCCC. The existence of ρ draws the conclusion $\mathcal{SPEC}_s$ is a divergent specification.

Via the analysis of $\mathcal{SPEC}_s$, Theorem 1 tell us that the existence of a UCCC in a CCSL specification witnesses its divergence. Unfortunately, this is a sufficient, but not necessary, condition for deciding specification's divergence. We say nothing about the convergence even if we get one or more BCCCs from a CCSL specification. The **Bounded Clock Set** can be used to determine a specification's convergence.

Definition 12 (Bounded Clock Set). For a given clock set $\mathcal{C}$, a clock set $\mathfrak{B} = \{c_\perp, c_1, \ldots, c_m, c^\top, c_{m+1}\ldots, c_{m+n}\}$, subset of $\mathcal{C}$, which contains at least 2 elements, is called a Bounded Clock Set (BCS), if all the following conditions hold:

  (i)   (**Lower-upper Bound**) $\exists d \in \mathbb{N}, |c_\perp, c^\top| \leq d,$
  (ii) (**Causality Bound**) $\forall i \in [1..m], c_\perp \preccurlyeq c_i \preccurlyeq c^\top,$
  (iii) (**Absence Unbounded Clock**) $\forall i \in [m+1..m+n], \exists c \in \mathfrak{B}_R$ *(see below)*, $d \in \mathbb{N}, |c, c_i| \leq d.$

The clock $c_\perp$ (resp. $c^\top$) is called the bottom (resp. top) clock. The subset $\mathfrak{B}_R = \{c_\perp, c_1, \ldots, c_m, c^\top\}$ of $\mathfrak{B}$ is called Causal Bounded Clock Set (CBCS). ■
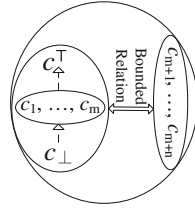

Fig. 2. The BCS structure

A BCS $\mathfrak{B} = \{c_\perp, c_1, \ldots, c_m, c^\top, c_{m+1}\ldots, c_{m+n}\}$, as shown in Fig. 2, includes two disjoint subsets:

- $\mathfrak{B}_R = \{c_\perp, c_1, \ldots, c_m, c^\top\}$ contains the fastest clock $c_\perp$ as the bottom, and the slowest clock $c^\top$ as the top, while all other ones' speed lies between $c_\perp$ and $c^\top$. Note that $m = 0$ in some cases.
- $\mathfrak{B}\backslash\mathfrak{B}_R = \{c_{m+1}\ldots, c_{m+n}\}$ We don't care about the speed of these clocks in relation to the one in $\mathfrak{B}_R$. But a common fact is that all of them must be bounded by speed of one of the clocks in $\mathfrak{B}_R$. Note that $n = 0$ in some cases, i.e., this set is an empty set.

The bottom (resp. top) clock is probably not a "proper" bottom (resp. top) clock because maybe there exists a clock in $\mathfrak{B}\backslash\mathfrak{B}_R$ which is faster (resp. slower) than it.

Theorem 2 (BCS Implies Divergence-Freedom). A CCSL specification $\mathcal{SPEC} = <\mathcal{C}, CConstr>$ is free of divergence, if there exists a BCS $\mathfrak{B} \supseteq \mathcal{C}$ implied by the *CConstr*. ■

Proof of Theorem 2:

Let $\mathfrak{B} = \{c_\perp, c_1, ..., c_m, c^\top, c_{m+1}..., c_{m+n}\}$, and $\mathfrak{B}_R = \{c_\perp, c_1, ..., c_m, c^\top\}$

By applying Proposition 9 ***(Bounded restriction)*** *to condition (i) and (ii) in* Definition 12, *we have*

$$\forall i \in [1,...,m], \; clock \, pairs \, (c_\perp, c_i) \; and \; (c_i, c^\top) \, are \, bounded. \qquad (\text{B\_1})$$

By Proposition 8 ***(Bounded extension),*** *via the $c_\perp$ or $c^\top$ as the middle clock b occurs, we have*

$$\forall i,j \in [1,...,m], \; clock \, pair \, (c_i, c_j) \; is \, bounded. \qquad (\text{B\_2})$$

Summarizing (B_1) *and* (B_2),

$$\forall c_i, c_j \in \mathfrak{B}_R, \; clock \, pair \, (c_i, c_j) \; is \, bounded. \qquad (\text{B\_3})$$

From condition (iii) in Definition 12, *$\forall i \in [m+1..m+n]$, $\exists c \in \mathfrak{B}_R$, clock pair ($c$, c_i) is bounded,*

By Proposition 8 *again via the c occurs in the last line as the middle clock b, we have*

$$\forall c_i \in \mathfrak{B}_R, c_j \in \mathfrak{B} \backslash \mathfrak{B}_R \, clock \, pair \, (c_i, c_j) \; is \, bounded. \qquad (\text{B\_4})$$

$$\forall c_i, c_j \in \mathfrak{B} \backslash \mathfrak{B}_R, \; clock \, pair \, (c_i, c_j) \; is \, bounded. \qquad (\text{B\_5})$$

Summarizing (B_3), (B_4) *and* (B_5), *we conclude*

$\forall c_i, c_j \in \mathfrak{B}$, clock pair ($c_i, c_j$) is bounded.

Because $\mathfrak{B} \supseteq \mathcal{C}$, $\forall c_i, c_j \in \mathcal{C}$, clock pair ($c_i, c_j$) is bounded.

By Definition 10 ***(Bounded Specification)***, *$\mathcal{SPEC}$ is divergence-freedom specification.* ∎

If we cannot derive a bounded relation from $\mathcal{C}$ obviously, introducing some external clock(s) that can form BCCC is allowed. Therefore, we use $\mathfrak{B} \supseteq \mathcal{C}$ rather than $\mathfrak{B} = \mathcal{C}$ in Theorem 2.
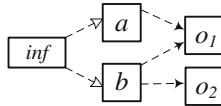

Fig. 3. Causality Clock Graph for $\mathcal{SPEC}_i$

Let us illustrate Theorem 2 on a simple example, $\mathcal{SPEC}_i = <\mathcal{C}, CConstr>$, where $\mathcal{C} = \{a, b, inf(a, b), o_1, o_2\}$ and $CConstr = \{ a \preccurlyeq o_1, inf(a, b) \sim o_1, b \sim o_2, b \preccurlyeq o_1\}$.

Let $inf = inf(a, b)$, via some given propositions above, we can get the explicit and implicit *Causality* relation list from *CConstr* (Fig. 3):

$inf \leqslant a \leqslant o_1,\ b \leqslant o_1,\ b \leqslant o_2$

By Proposition 11, we know clock pairs (inf, o_1) and (b, o_2) are bounded. Then we can construct a BCS $\mathfrak{B}_{speci} = \{c_\perp = inf,\ c_1 = a,\ c_2 = b,\ c^\top = o_1,\ c_3 = o_2\}$ shown in Fig. 4a with $m = 2$ and $n = 1$ in Definition 12. Because $\mathfrak{B}_{speci} \supseteq (\mathcal{SPEC}_i)$, we assert $\mathcal{SPEC}_i$ is free of divergence by Theorem 2.

Note that maybe there several other possibilities to assign the bottom or top clock. For example, $\mathfrak{B}_{speci\_alt} = \{c_\perp = inf,\ c^\top = b,\ c_1 = a,\ c_2 = o_1, c_3 = o_2\}$ shown in Fig. 4b is another same member set but assigned with different top clock. Here $m = 0$ and $n = 3$.
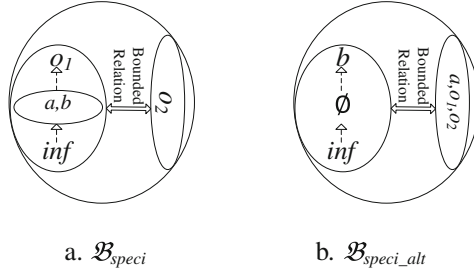

a. $\mathcal{B}_{speci}$ b. $\mathcal{B}_{speci\_alt}$

Fig. 4. Two represents for BCS of $\mathcal{SPEC}_i$

$\mathfrak{B}_{speci}$ is a "much better" BCS than $\mathfrak{B}_{speci\_alt}$ in the sense of determining the boundedness among clocks. We can easily assert $\mathfrak{B}_{speci}$ is a BCS satisfied by conditions (i), (ii) and (iii) in Definition 12. On the contrary, As to $\mathfrak{B}_{speci\_alt}$, both conditions (i) and (iii) are not straightforwardly asserted using the *CConstr($\mathcal{SPEC}_i$)* as well as the associated propositions list above. In fact, $\mathfrak{B}_{speci\_alt}$ is constructed dedicatedly to show its inconvenience when I know the fact convergence. In purpose of the efficiency for convergence assertion, Subsect. 3.3 will design an algorithm for constructing BCS.

All the proofs about these propositions and theorems in Sects. 3.1 and 3.2 are completed with the help of PVS. Hence, we can try to solve some problem without any doubt about its correctness. The following theorem, written in PVS specification, is a special case[1] of Theorem 2, can be used to assert divergence-free of CCSL specifications.

```
bounded_set_boundSPEC: theorem exists (m: nat):
    forall sgm: ((bot, top: Aclock): (sgm |= (bot <= top)) &
        maxDrift(sgm)(bot, top)(m) & (forall (c: clock): sgm |= (bot <= c & c <= top)))
    => forall (a, b: clock): maxDrift(sgm)(a, b)(m)
```

Note:

---

[1] condition (iii) in Definition 12 is not necessarily be considered.

(1) *Clock* is type to represent *explicit* clock set.
(2) *AClock* is a defined ADT includes both *Clock* and *implicit* clocks defined in Definition 4.
(3) "⊨" is interpreted via Definitions 6 and 7.
(4) *maxDrift* is *Bounded relation* in Definition 9.

3.3 Detection Algorithm

For a given specification $\mathcal{SPEC}$ = <$\mathcal{C}$, *CConstr*>, there are three simply **rules** to prevent $\mathcal{SPEC}$ from divergence:

Rule list for avoiding divergence

- **Rule 1** (**Redundant Clock Nonexistence**) There exists no a clock $c \in \mathcal{C}$ does not occur in any clock constraints in *CConstr*, clock c can tick arbitrarily otherwise.
- **Rule 2** (**Bounds Existence**) At least one expression in the form of $a \$ n$ or *FilteredBy* can be found in *CConstr*. None of clock pair is bounded otherwise. Note that the delay expression may be in the implicit form. For instance, it can be found only by expanding the alternation in $\mathcal{SPEC}_i$.
- **Rule 3** (**Absence of disordered clock**) A disorder clock $c \in \mathcal{C}$ is a clock which occurs only in the form of $c \# b$, cannot be found in other clock constraints. A disorder clock can tick arbitrarily except the requirement its exclusion with a certain clock b.

One violation of **Rules 1–3** causes the specification's divergence. The following parts will consider only the specification that follows **Rules 1–3**.

If we have no enough faith to assert the convergence CCSL specification, we can try to witness its divergence via discovering a UCCC by Theorem 1 because it is much easier to find an unbounded clock pair than to determine all the clock pairs are bounded.

When there is no obvious UCCC be found from a CCSL specification, we need to design an algorithm to try to construct a BCS, for the purpose of using Theorem 2 to guarantee specification's convergence.

For a given $\mathcal{SPEC} = <\mathcal{C}, CConstr>$, we first sort the clock based on the *Causality*-related clock constraints (includes *Causality* relation and those imply it) in *CConstr* with regard to *Causality*.

Algorithm 1   Divergence analysis

INPUT: SPEC=$<\mathcal{C}$, *CConstr*$>$,

OUTPUT: YES if SPEC is Divergence-freedom, NO otherwise.

$\mathfrak{B} = \emptyset$.

Ⅰ) Get a clock pair (a, b) in $\mathcal{C}$ s.t. $|a\text{-}b| \leq m$ for some $m \in \mathbb{N}$ and $a \preccurlyeq$ b, let $c_\perp = a$, $c^\top = b$

$\mathfrak{B} = \mathfrak{B} \cup \{c_\perp, c^\top\}$, $\mathcal{C} = \mathcal{C} \backslash \{c_\perp, c^\top\}$

If $\mathcal{C} = \emptyset$ return YES

If $\exists c \in \mathcal{C}$, $c^\top \preccurlyeq c$ and $(c^\top, c)$ is bounded

$\mathfrak{B} = \mathfrak{B} \cup \{c\}$, $\mathcal{C} = \mathcal{C} \backslash \{c\}$, $c^\top = c$

If $\mathcal{C} = \emptyset$ return YES

If $\exists c \in \mathcal{C}$, $c \preccurlyeq c_\perp$ and $(c_\perp, c)$ is bounded

$\mathfrak{B} = \mathfrak{B} \cup \{c\}$, $\mathcal{C} = \mathcal{C} \backslash \{c\}$, $c_\perp = c$

If $\mathcal{C} = \emptyset$ return YES

Ⅱ) For each clock $c \in \mathcal{C}$, such that $c_\perp \preccurlyeq c \preccurlyeq c^\top$,

$\mathfrak{B} = \mathfrak{B} \cup \{c\}$,$\mathcal{C} = \mathcal{C} \backslash \{c\}$

If $\mathcal{C} = \emptyset$ return YES

Ⅲ) For each clock $c \in \mathcal{C}$, such that $\exists b \in \mathfrak{B}$, pair (b, c) is bounded

$\mathfrak{B} = \mathfrak{B} \cup \{c\}$,$\mathcal{C} = \mathcal{C} \backslash \{c\}$

If $\mathcal{C} = \emptyset$ return YES

Return NO

Algorithm 1 includes three parts:

(I)   Get (and update if required) the bottom and top clock.
(II)   Get the clock set in which it is not faster than the bottom and not slower than the top.
(III) Analyze the boundedness of left clocks.

We can assert that $\mathfrak{B}$ constructed in Algorithm 1 must be a BCS by Propositions 8 and 9. So this algorithm's correctness is ensured by Theorem 2 since $\mathcal{C} = \emptyset \Longrightarrow \mathfrak{B} \supseteq \mathcal{C}$.

Algorithm 1 must terminate because of the finiteness of clock set. The complexity of Algorithm 1 is ($|\mathcal{C}|$). If most clocks can be dealt with in part I and II, the efficiency of algorithm is very high. Therefore, via Algorithm 1, determining boundedness among all clock pairs is converted into checking boundedness on much fewer clock pairs and additionally sorting clocks w.r.t. *Causality* relation.

4   Case Study

To illustrate the approach, we take an example inspired by [14], that was used for flow latency analysis on Architecture Analysis and Design Language (AADL) specifications. However, with CCSL we are conducting different kinds of analyses.
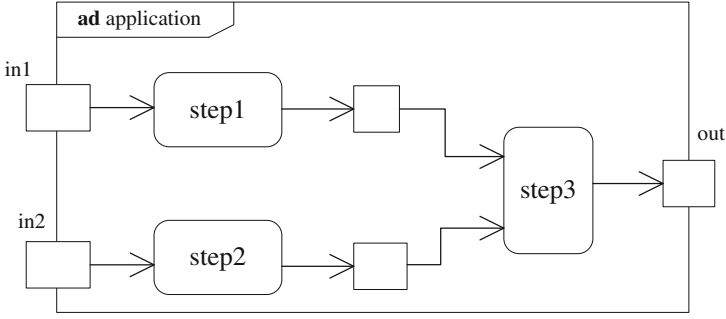
Fig. 5. Simple application

Figure 5 considers a simple application described as a UML activity. This application captures two inputs *in1* and *in2,* performs some calculations (*Step1*, *Step2* and *Step3*) and then produces a result *out*. This application has the possibility to compute *Step1* and *Step2* concurrently depending on the chosen execution platform. This application runs in a streaming-like fashion by continuously capturing new inputs and producing outputs.

To abstract this application as a CCSL specification, we assign one clock to each action. The clock has the exact same name as the associated action (e.g., *step1*). We also associate one clock with each input, this represents the capturing time of the inputs, and one clock with the production of the output (*out*). The successive instants of the clocks represent successive executions of the actions or input sensing time or output release time. The basic CCSL specification is $\mathcal{SPEC}_{\mathrm{simp}} = <\mathcal{C}, \mathrm{CConstr}>$, where $\mathcal{C} = \{in1, in2, step1, step2, step3, out\}$, *CConstr* includes the following clock constraints:

$$in1 \preccurlyeq step1 \bigwedge step1 \prec step3 \tag{F.8}$$

$$in2 \preccurlyeq step2 \bigwedge step2 \prec step3 \tag{F.9}$$

$$step3 \preccurlyeq out \tag{F.10}$$

(F.8) specifies that *step1* may begin as soon as an input *in1* is available. Executing *step3* also requires *step1* to have produced its output. (F.9) is similar for *in2* and *step2*. (F.10) states that an output can be produced as soon as *step3* has executed. Note that CCSL precedence is well adapted to capture infinite FIFOs denoted on the figure as object nodes. Such a specification is clearly not convergent because of its violation of *Rule 2* (**Bounds Existence**) in **Rule list for avoiding divergence**. After the sorting the clocks w.r.t. *Causality* relation, we get two CCCs:

$$\rho_1 : in \preccurlyeq step1 \preccurlyeq step3 \preccurlyeq out$$

$$\rho_2 : in \preccurlyeq step1 \preccurlyeq step3 \preccurlyeq out$$

It is also stated again that $\mathcal{SPEC}_{\text{simp}}$ is divergent by Theorem 1 as both ρ_1 and ρ_2 are unbounded. If one CCSL constraint like (F.11) is added into $CConstr(\mathcal{SPEC}_{\text{simp}})$ as a test like that in [5].

$$sup(in1, \ in2) \sim out \tag{F.11}$$

By Proposition 6, the following are two new UCCCs are acquired since the addition of (F.11):

$in1 \preccurlyeq sup(in1, \ in2)$

$in2 \preccurlyeq sup(in1, \ in2)$

Now we try to use Algorithm 1 *to check whether* $\mathcal{SPEC}_{\text{simp}}$ *is free of divergence or not. Let* $sup_{in12}/sup(in1, in2)$, by Expanding the Alternation definition, (F.11) becomes

$sup_{in12} \prec out \wedge out \prec (sup_{in12} \ \$ \ 1)$

Now $(\mathcal{SPEC}_{\text{simp}}) = \{in1, in2, step1, step2, step3, out, sup_{in12}, sup_{in12} \ \$ \ 1\}$ correspondingly, by part I of Algorithm 1, we get $\mathfrak{B} = \{c_\perp = sup_{in12}, c^\top = sup \ \$1\}$, then using part II of Algorithm 1, the clock *out* is added into $\mathfrak{B}$. Furtherly, because of ρ_1, ρ_2 and the fact sup_{in12} is the fastest clock that is slower than *in1* and *in2* by Definition 7, we can deduce $\forall c \in \{step1, step2, \ step3\}$, $c_\perp \preccurlyeq c \preccurlyeq c^\top$ by Propositions 1 to 7. Therefore, the clocks *step1, step2* and *step3* can also be added into $\mathfrak{B}$ via part II of Algorithm 1. Up to now, $\mathfrak{B} = \{c_\perp = sup_{in12}, step1, step2, step3, out, c^\top = sup_{in12} \ 1\}$, and none of other clock can be added into $\mathfrak{B}$ further. Therefore, $\mathcal{SPEC}_{\text{simp}}$ *is divergent because* $\mathfrak{B} \not\supseteq \mathcal{C}$ witnessed by *in1,in2* $\in \mathcal{C}$ but *in1,in2* $\notin \mathfrak{B}$. This is caused by that bounds on **Supremum** do not imply bounds on *in1(or in2)* and *out*, not to mention to the bound on *in1* and *in2*.

To become a bounded(or safe) system $\mathcal{SPEC}_{\text{simp\_safe}}$, we can for instance replace (F.11) by (F.12).

$$inf(in1, \ in2) \sim out \tag{F.12}$$

Let $in f_{in12}/inf(in1, \ in2)$, in fact, (F.12) equals

$in f_{in12} \prec out \wedge out \ (inf_{in12} \ \$ \ 1)$

Because of introducing new clock constraint (F.12), some new clocks (implicit clocks) are introduced as well, now $\mathcal{C}(\mathcal{SPEC}_{\text{simp\_safe}}) = \mathcal{C}(\mathcal{SPEC}_{\text{simp}}) \cup \{inf_{in12}, inf_{in12} \ \$ \ 1\}$.

Let's check its divergence-freedom again by Algorithm 1. By part I of Algorithm 1, we get $\mathfrak{B} = \{c_\perp = inf_{in12}, c^\top = inf_{in12} \ \$ \ 1\}$, then using part II of Algorithm 1, all the clocks in $\mathcal{C}$ can be added into $\mathfrak{B}$ because their speed are constrained between the slowest clock inf_{in12} and the fastest clock $inf_{in12} \ \$ \ 1$ *as revealed by the following CCCs deduced by the* Propositions 1, 5 and 7 *as well as* $\rho1$ and $\rho2$ above.

$inf_{in12} \preccurlyeq in1 \preccurlyeq step1 \preccurlyeq step3 \preccurlyeq out \preccurlyeq (inf_{in12} \ \$ \ 1)$

$inf_{in12} \preccurlyeq in2 \preccurlyeq step2 \preccurlyeq step3 \preccurlyeq out \preccurlyeq (inf_{in12} \ \$ \ 1)$

The resulting $\mathfrak{B} = \{c_\perp = inf_{in12}, in1, step1, in2, step2, step3, out, c^\top = inf_{in12} \ \$ \ 1\}$, as shown in Fig. 6, is obviously a superset of $\mathcal{C}(\mathcal{SPEC}_{\text{simp\_safe}})$. Hence, the CCSL specification $\mathcal{SPEC}_{\text{simp\_safe}}$ is free of divergence. Note that $\mathfrak{B}$ is also a CBCS because all of clock $c \in \mathfrak{B}$, $c_\perp \preccurlyeq c \preccurlyeq c^\top$.
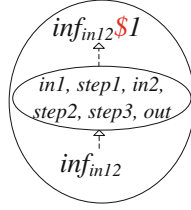
Fig. 6. BCS of simple application

With the help of *bounded_set_boundSPEC* in Sect. 3.2, we can complete the proof of theorem *simp_safe* for asserting the boundedness of $\mathcal{SPEC}_{\text{simp\_safe}}$ in PVS by the lemma *simp_bcs* as follows. In fact, $\mathcal{SPEC}_{\text{simp\_safe}}$ is a specification free of divergence.

---

simp_bcs: **lemma**   **let** *bot = in1 ∧ in2 , top =delay(bot,1)* **in**
    maxDrift(sgm)(bot,top)(1) & (forall (c: clock): sgm |= (bot <= c & c <= top)))
simp_safe: **theorem**   **forall** *(a,b:Clock) : maxDrift(sgm)(bot,top)(1)*

---

5   Related Work

Some techniques were provided as an effort to analyze CCSL specifications. [15] implemented the automatic analysis by translating CCSL into signal, for the purpose of generating executable specifications through discrete controller synthesis. However, this work did not consider the Infimum and Supremum operators that introduce unbounded counters and did not address the problem of deciding whether the specification is divergence-freedom or not. Exhaustive analysis of CCSL through a transformation into labeled transition systems has already been attempted in [16]. However, in those attempts, the CCSL operators were bounded because the underlying model-checkers cannot deal with infinite labeled transition systems.

In [6], the authors showed that even though the primitive constraints were unbounded, the composition of these primitive constraints could lead to a system where only a finite number of states were accessible. [7] defines a notion of safety for CCSL and establish a condition to decide whether a specification is safe on the transformed marked graph from CCSL.

All the above works share one common point: the specification analysis were done by some transformation and performed on the transformed target. The results were dependent on the correctness and efficiency of the mechanized transformation.

Our contribution is straightforward based on the clock *Causality* relation used to sort and the clock expression used to determine the clock pair's boundary. It is not necessary for the reader to have the other mathematic theory preliminaries except the basic set-theory.

6   Conclusion and Future Works

Based on the state-based semantics of a kernel subset of CCSL, We have presented a sufficient condition to discover the CCSL divergence existence, and an easily constructed Bounded Clock Set (BCS) for deciding the convergence of CCSL. An algorithm is proposed to actually build BCS of a given specification with the help of sorted the bounded clock chain with respect to causality relation and the clock delay expression used to decide clock pair's boundary. Therefore, determining boundedness among all clock pairs is converted into checking boundedness on much fewer clock pairs and additionally sorting clocks with respect to causality relation. Finally, a simple application's convergence is investigated. We first discover its divergence by the existence of unbounded clock causality chain. Consequently, a BCS is built to ensure the new specification is convergent by adding suitable constraint.

As a future work, we plan to extend and prove the extensive application of clock causality chain further. For instance, the potential causality conflict between clocks may be found if the specification is not deadlock-free or ill-formed, the periodicity in a schedule may be revealed in the chain in some style, etc.

References

1. OMG, UML Profile for MARTE v1.0. Object Management Group (2009). formal/02-11-2009
2. André, C., Mallet, F., de Simone, R.: Modeling time(s). In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 559–573. Springer, Heidelberg (2007)
3. André, C.: Syntax and Semantics of the Clock Constraint Specification Language (CCSL), p. 37. Inria I3S Sophia Antipolis (2009)
4. DeAntoni, J., Mallet, F.: TimeSquare: treat your models with logical time. In: Furia, C.A., Nanz, S. (eds.) TOOLS 2012. LNCS, vol. 7304, pp. 34–41. Springer, Heidelberg (2012)
5. Mallet, F., Millo, J.-V., Romenska, Y.: State-based representation of CCSL operators (2013)
6. Mallet, F., Millo, J.-V.: Boundness issues in CCSL specifications. In: Groves, L., Sun, J. (eds.) ICFEM 2013. LNCS, vol. 8144, pp. 20–35. Springer, Heidelberg (2013)
7. Mallet, F., Millo, J.-V., De Simone, R.: Safe CCSL specifications and marked graphs. In: MEMOCODE - 11th IEEE/ACM International Conference on Formal Methods and Models for Codesign. IEEE CS (2013)
8. Deantoni, J., André, C., Gascon, R.: CCSL denotational semantics, p. 29. Inria I3S Sophia Antipolis (2014)
9. Benveniste, A., et al.: The synchronous languages 12 years later. Proc. IEEE **91**(1), 64–83 (2003)
10. Ling, Y., et al.: Schedulability analysis with CCSL specifications. In: 2013 20th Asia-Pacific Software Engineering Conference (APSEC) (2013)
11. Owre, S., Shankar, N.: Abstract Datatypes in PVS. Computer Science Laboratory, SRI International, Menlo Park (1993)
12. Owre, S., Rushby, J.M., Shankar, N.: PVS: A prototype verification system. In: Kapur, D. (ed.) CADE 2011. LNCS (LNAI), vol. 607, pp. 748–752. Springer, Heidelberg (1992)

13. Computer Science Laboratory, SRI International. PVS: Specification and Verification System. http://pvs.csl.sri.com/
14. Feiler, P.H., Hansson, J.: Flow latency analysis with the architecture analysis and design language (AADL), CMU (2007)
15. Mallet, F., Andre, C.: UML/MARTE CCSL, Signal and Petri nets (2008)
16. Gascon, R., Mallet, F., DeAntoni, J.: Logical time and temporal logics: comparing UML MARTE/CCSL and PSL. In: 2011 Eighteenth International Symposium on Temporal Representation and Reasoning (TIME) (2011)