On the Complexity of Verification of Time-Sensitive Distributed Systems

Max Kanovich[1,6] Tajana Ban Kirigin[2(✉)] Vivek Nigam[3,4] Andre Scedrov[5] and Carolyn Talcott[7]

[1] University College, London, UK, m.kanovich@ucl.ac.uk
[2] Department of Mathematics University of Rijeka, Rijeka, Croatia, bank@uniri.hr
[3] Federal University of Paraíba, João Pessoa, Brazil, vivek@ci.ufpb.br
[4] Munich Research Center, Huawei, Munich, Germany
[5] University of Pennsylvania, Philadelphia, USA, scedrov@math.upenn.edu
[6] HSE University, Computer Science Dept., Moscow, Russia
[7] SRI International, Menlo Park, USA, clt@csl.sri.com

Abstract. This paper develops a Multiset Rewriting language with explicit time for the specification and analysis of Time-Sensitive Distributed Systems (TSDS). Goals are often specified using explicit time constraints. A good trace is an infinite trace in which the goals are satisfied perpetually despite possible interference from the environment. In our previous work [14], we discussed two desirable properties of TSDSes, *realizability* (there exists a good trace) and *survivability* (where, in addition, all admissible traces are good). Here we consider two additional properties, *recoverability* (all compliant traces do not reach points-of-no-return) and *reliability* (the system can always continue functioning using a good trace). Following [14], we focus on a class of systems called *Progressing Timed Systems* (PTS), where intuitively only a finite number of actions can be carried out in a bounded time period. We prove that for this class of systems the properties of recoverability and reliability coincide and are PSPACE-complete. Moreover, if we impose a bound on time (as in bounded model-checking), we show that for PTS the reliability property is in the Δ_2^p class of the polynomial hierarchy, a subclass of PSPACE. We also show that the bounded survivability is both NP-hard and coNP-hard.

Keywords: Multiset Rewriting · Time-Sensitive Distributed Systems · Complexity

Dedicated to Joshua Guttman with gratitude for his inspiration and friendly and insightful discussions.

1 Introduction

In our previous work [14], we considered the verification of Time-Sensitive Distributed Systems (TSDS) motivated by applications with autonomous drones performing surveillance of an area. The drones must always collectively have recent pictures, *i.e.*, at most M time units old, of certain strategic locations. In attempting to achieve this goal, the

drones consume energy and must return to the base station to recharge their batteries. In addition, the environment may interfere as there may be winds that move the drone in a certain direction, or other flying objects may block a drone's path.

In [14] we considered two verification properties, realizability and survivability. Here we introduce two more properties, reliability and recoverability. Let us explain all four properties in a little more detail. The *realizability* problem consists of checking, whether under the given time constraints, the specified system can achieve the assigned goal, *e.g.*, always collect recent pictures of the sensitive locations. In many settings, the drones themselves or the environment may behave non-deterministically. For example, if a drone wants to reach a point in the northeast, it may initially move either north or east, both being equally likely. Similarly, there could be wind at a particular location, causing any drone under the influence of the wind to move in the direction of the wind. A stronger property, *survivability*, accounts for such nondeterminism and tests whether the specified system can achieve the assigned goal for all possible outcomes (of drone actions and environmental influences). The properties of realizability and survivability represent the two extremes w.r.t. requirements placed on a system. A system that is realizable can achieve the designed goal in some way. A system that satisfies survivability will always achieve the goal, under all circumstances. In some cases, realizability may not be satisfactory, while in others, survivability may be too costly or unattainable. For such systems, intermediate solutions are of interest.

To model such intermediate requirements in system design, in this paper we introduce additional properties, namely *reliability* and *recoverability*. In order to ensure system goals, drones should always be able to function. In particular, drones should always be able to come back to recharge, both in terms of distance and energy. In other words, drones should never go too far and reach so-called *points-of-no-return* where it may no longer be possible to safely return to home base. Engineers should strive to program drones to avoid reaching points-of-no-return. This property is referred to as *recoverability*. A system satisfies *reliability* if the system is always able to successfully continue its expected performance, *i.e.*, the system never gets stuck. For example, drones should always be able to ensure the system goals, regardless of the disturbances they have experienced in the environment. At any point in time, after the drones have successfully monitored sensitive locations for a certain period of time, they should be able to find a way to continue with their good performance. For example, considering possible technical failures and maintenance of the drones, it may be necessary for engineers to call in additional drones to collectively provide up-to-date images of the entire area of interest.

Following [14], we focus on a class of systems called *Progressing Timed Systems* (PTS), which are specified as timed multiset rewriting theories. In a PTS, only a finite number of actions can be carried out in a bounded time interval. In addition to formalizing the properties, we show that the following relations hold for PTS:

$$S_{urvivability} \implies R_{eliability} \iff R_{ecoverability} \implies R_{ealizability}\,.$$

In their spirit, these properties seem similar to safety and liveness properties [1] or a combination of these properties. However, it is not straightforward to classify them in these terms. Namely, the properties we consider, formally defined in Section 3.3, contain

an alternation of quantifiers, which makes it more challenging to formally represent them as a combination of safety and liveness properties [1].

In our previous work [19, 18, 13, 17], we proposed a timed Multiset Rewriting (MSR) framework for specifying compliance properties similar to quantitative safety properties [1, 6] and investigated the complexity of a number of decision problems. These properties were defined over sets of finite traces, *i.e.*, executions of a finite number of actions. The above properties, on the other hand, are defined over *infinite traces*. The transition to properties over infinite traces leads to many challenges, as one can easily fall into undecidable fragments of verification problems. The main challenge is to identify the syntactic conditions on specifications so that the verification problems fall into a decidable fragment and, at the same time, that interesting examples can be specified.

The remainder of the paper is organized as follows:

- Following [14], in Section 2 we discuss *Progressing Timed Systems*.
- In Section 3 we define concepts for specifying the relevant quantitative temporal properties of timed systems used to define the properties of *realizability*, *reliability, recoverability* and *survivability*.
- In Section 4 we then formally compare the expressiveness of these properties.
- Section 5 investigates the complexity of verification problems that involve the above properties. While these problems are undecidable in general [19], we show that they are PSPACE-complete for PTSes. We also show that, when we bound time (as in bounded-model checking), realizability of PTSes is NP-complete, and that survivability and reliability are in the Δ_2^p class of the polynomial hierarchy (P^{NP}) [27]. We also obtain the NP and co-NP lower bound for the n-time bounded survivabilty problem.
- Section 6 provides a discussion on related and future work.

Relation to our previous work This paper considerably extends the conference paper [14]. All the material involving properties of reliability and recoverability is new, including the investigation of the relations among all four properties from Section 4, the complexity results relating to reliability from Section 5, and the lower bound complexity results for n-time bounded survivability are new. Due to space constraints, many proofs and detailed considerations are placed in the technical report [16].

2 Multiset Rewriting Systems

We briefly review timed multiset rewriting with discrete time of [19].

Assume a finite first-order typed alphabet, Σ, with variables, constants, function and predicate symbols. Terms and formulas are constructed as usual (see [10]) by applying symbols of correct type (or sort). We assume that the alphabet contains the constant $z : Nat$ denoting zero and the function $s : Nat \to Nat$ denoting the successor function. Whenever it is clear from the context, we write n for $s^n(z)$ and $(n + m)$ for $s^n(s^m(z))$. In addition, we allow an unbounded number of fresh values [5, 9] to be involved.

If P is a predicate of type $\tau_1 \times \tau_2 \times \cdots \times \tau_n \to o$, where o is the type for propositions, and $u_1, \ldots, u_n$ are terms of types $\tau_1, \ldots, \tau_n$, respectively, then $P(u_1, \ldots, u_n)$ is a *fact*. A fact is *ground* if it contains no variables.

In order to specify timed systems, we attach a timestamp to each fact. *Timestamped facts* are of the form $F@t$, where F is a fact and $t \in \mathbb{N}$ is a natural number called

timestamp. For simplicity, we often just say facts instead of timestamped facts. Also, when we want to emphasize the difference between a fact F and a timestamped fact $F@t$, we say that F is an *untimed fact*.

Note that timestamps are *not* constructed using the successor function. Rather, timestamps can take any natural number value. To obtain the complexity results, we use a symbolic representation of the problems and abstractions that can handle unbounded time values. For more insight see discussion after Definition 2.

There is a special predicate symbol $Time$ with arity zero that is used to represent global time. A *configuration* is a finite multiset of ground timestamped facts, $\mathcal{S} = \{\ Time@t,\ F_1@t_1, \ldots,\ F_n@t_n\ \}$ with a single occurrence of a $Time$ fact.

Given a configuration $\mathcal{S}$ containing $Time@t$, a fact $F@t_F$ in $\mathcal{S}$ is a *future fact* if its timestamp is greater than the global time t, *i.e.*, if $t_F > t$. Similarly, a fact $F@t_F$ in $\mathcal{S}$ is a *past fact* if $t_F < t$, and a fact $F@t_F$ in $\mathcal{S}$ is a *present fact* if $t_F = t$.

Configurations are to be interpreted as states of the system, *e.g.*, configuration

$$\mathcal{S}_1 = \{\, Time@4,\ Dr(d1, 1, 2, 10)@4,\ Dr(d2, 5, 5, 8)@4,\ P(p1, 1, 1)@3,\ P(p2, 4, 6)@1\,\}$$

denotes a scenario with two drones located at positions $(1, 2)$ and $(5, 5)$, with 10 and 8 energy units, and with two points to be monitored at positions $(1, 1)$ and $(4, 6)$. The former was last photographed at time 3 and the latter at time 1. The global time is 4.

Using variables, including time variables, we are able to represent configurations of particular form. For example, configuration

$$Time@(T + D),\ Dr(X, 5, 6, Y)@(T + D),\ P(p2, 4, 6)@T$$

specifies that some drone X with Y energy units is currently at the position $(5, 6)$ and that the point of interest at position $(4, 6)$ was last photographed D time units ago. This holds for any configuration containing the above facts for some instantiation of the variables T, D, X and Y.

Configurations are modified by rewrite rules which can be interpreted as actions of the system. There is only one rule, $Tick$, which represents how global time advances:

$$Time@T \longrightarrow Time@(T + 1), \tag{1}$$

where T is a time variable denoting the global time. With an application of a $Tick$ rule, a configuration, $\{\ Time@t,\ F_1@t_1, \ldots,\ F_n@t_n\ \}$, representing the state of a system at time t, is replaced with the configuration $\{\ Time@(t + 1),\ F_1@t_1, \ldots,\ F_n@t_n\ \}$ representing the system at time $t + 1$.

The remaining rules are *instantaneous*, since they do not modify global time, but may modify the remaining facts of configurations (those different from $Time$). Instantaneous rules have the form:

$$\begin{aligned} &Time@T,\ W_1@T_1, \ldots,\ W_p@T_p,\ F_1@T'_1, \ldots,\ F_n@T'_n \ \mid\ \mathcal{C} \longrightarrow \\ &\exists \vec{X}.\,[\,Time@T,\ W_1@T_1, \ldots,\ W_p@T_p,\ Q_1@(T + d_1), \ldots,\ Q_m@(T + d_m)\,] \end{aligned} \tag{2}$$

where $d_1, \ldots, d_m$ are natural numbers, $W_1@T_1, \ldots, W_p@T_p, F_1@T'_1, \ldots, F_n@T'_n$ are timestamped facts, possibly containing variables, and $\mathcal{C}$ is the guard of the rule which is a set of constraints involving the time variables that appear as timestamps of facts in the pre-condition of the rule, *i.e.*, the variables $T, T_1, \ldots, T_p, T'_1, \ldots, T'_n$. The facts W_i, F_j and Q_k are all different from the fact $Time$ and $\vec{X}$ are variables that do not appear in $W_1, \ldots, W_p, F_1, \ldots, F_n$.

Constraints may be of the form $T > T' \pm d$ or $T = T' \pm d$, where T and T' are time variables, and $d \in \mathbb{N}$ is a natural number. Here and throughout the rest of the paper, the symbol $\pm$ stands for either $+$ or $-$, *i.e.*, constraints may involve addition or subtraction. We use $T' \geq T' \pm d$ to denote the disjunction of $T > T' \pm d$ and $T = T' \pm d$. All variables in the guard of a rule are assumed to appear in the rule's pre-condition.

Finally, the variables $\vec{X}$ that are existentially quantified in a rule (Eq. 2) are to be replaced by fresh values, also called *nonces* in the protocol security literature [5, 9]. As in our previous work [12], we use nonces whenever unique identification is required, for example for drone identification.

A rule $\mathcal{W} \mid \mathcal{C} \longrightarrow \exists \vec{X}.\, \mathcal{W}'$ can be applied to a configuration $\mathcal{S}$ if there is a ground substitution σ such that $\mathcal{W}\sigma \subseteq \mathcal{S}$ and that $\mathcal{C}\sigma$ is true. The resulting configuration is $\big((\mathcal{S} \setminus \mathcal{W}) \cup \mathcal{W}'\big)\sigma$, where variables $\vec{X}$ are fresh. More precisely, given a rule r, an instance of a rule is obtained by substituting constants for all variables appearing in the pre- and post-condition of the rule. This substitution applies to variables appearing in terms inside facts, to variables representing fresh values, and to time variables used to specify timestamps of facts. An instance of an instantaneous rule can only be applied if all the constraints in its guard are satisfied.

Following [9] we say that a timestamped fact $F@T$ is *consumed* by a rule r if this fact occurs more times on the left side than on the right side of the rule r . A timestamped fact $F@T$ is *created* by some rule r if that fact occurs more times on the right side than on the left side of the rule r. Hence, facts $F_1@T'_1, \ldots, F_n@T'_n$ are consumed by rule (Eq. 2) while facts $Q_1@(T + d_1), \ldots, Q_m@(T + d_m)$ are created by this rule. Note that a fact F can appear in a rule with different timestamps, but for the above notions we count instances of the same timestamped fact $F@T$. In a rule, we usually color red the consumed facts and blue the created facts.

Remark 1. Using constraints we are able to formalize time-sensitive properties and problems that involve explicit time requirements. The set of constraints may, however, be empty, *i.e.*, rules may have no constraints attached.

We write $\mathcal{S} \longrightarrow_r \mathcal{S}'$ for the one-step relation where the configuration $\mathcal{S}$ is rewritten into $\mathcal{S}'$ using an instance of rule r. For a set of rules $\mathcal{R}$, we define $\mathcal{S} \longrightarrow^*_{\mathcal{R}} \mathcal{S}'$ to be the transitive reflexive closure of the one-step relation on all rules in $\mathcal{R}$. We omit the subscript $\mathcal{R}$, when it is clear from the context, and simply write $\mathcal{S} \longrightarrow^* \mathcal{S}'$.

Note that due to the nature of multiset rewriting, there are various aspects of non-determinism in the model. For example, different actions and even different instantiations of the same rule may apply to the same configuration $\mathcal{S}$, leading to different resulting configurations $\mathcal{S}'$.

Definition 1 (Timed MSR System). *A* timed MSR system $\mathcal{T}$ *is a set of rules containing only instantaneous rules (Eq. 2) and the* $Tick$ *rule (Eq. 1).*

A trace of a timed MSR system is constructed by a sequence of its rules. In this paper, we consider both finite and infinite traces. A *finite trace* of a timed MSR system $\mathcal{T}$ starting from an initial configuration $\mathcal{S}_0$ is a sequence

$\mathcal{S}_0 \longrightarrow \mathcal{S}_1 \longrightarrow \mathcal{S}_2 \longrightarrow \cdots \longrightarrow \mathcal{S}_n$

and an *infinite trace* of $\mathcal{T}$ starting from an initial configuration $\mathcal{S}_0$ is a sequence

$$\mathcal{S}_0 \longrightarrow \mathcal{S}_1 \longrightarrow \mathcal{S}_2 \longrightarrow \cdots \longrightarrow \mathcal{S}_n \longrightarrow \mathcal{S}_{n+1} \longrightarrow \cdots$$

where for all $i \geq 0$, $\mathcal{S}_i \longrightarrow_{r_i} \mathcal{S}_{i+1}$ for some $r_i \in \mathcal{T}$. When a configuration $\mathcal{S}$ apperas in a trace P we write $\mathcal{S} \in P$.

We will pay particular attention to periods of time represented by traces. Since time advances by one unit of time per $Tick$ rule, a finite (infinite) number of $Tick$ rules in a trace represents a finite (infinite) time period. One can easily imagine traces containing a finite number of $Tick$ rules and an infinite number of instantaneous rules. Such traces would represent an infinite number of actions performed in a finite time interval. In this paper we are not interested in such traces and focus on so called *infinite time traces*.

Definition 2 (Infinite Time Trace). *A trace P of a timed MSR $\mathcal{T}$ is an* infinite time trace *if the time tends to infinity in P,* i.e.*, $(\forall n \in \mathbb{N})\ (\exists\, \mathcal{S} \in P)$ such that $Time@T \in \mathcal{S}$ and $T > n$.*

Since in any trace, the global time ticks in single time units, any infinite time trace is an infinite trace, and it contains an infinite number of $Tick$ rules.

We have shown in our previous work [20, 12, 19, 13, 17] that reachability problems for MSR are undecidable if no further restrictions are imposed, already when considering only finite traces. In order to obtain decidability, among other restrictions, we assume a bound on the size of facts. The size of a timestamped fact $P@T$, written $|P@T|$, is the total number of symbols appearing in P, not considering the timestamp. For instance, $|P(s(z), f(a, X), a)@12| = 7$. Without this bound, interesting decision problems can be shown undecidable by encoding the Post correspondence problem [9]. For our complexity results, it is also important to assume that the system is *balanced* [20, 19]. A timed MSR system $\mathcal{T}$ is *balanced* if for all instantaneous rules $r \in \mathcal{T}$, r creates the same number of facts as it consumes, *i.e.*, the instantaneous rules are of the form:

$$Time@T,\ \mathcal{W},\ F_1@T'_1, \ldots, F_n@T'_n \ \mid\ \mathcal{C} \longrightarrow \\ \exists \vec{X}.\ [\, Time@T,\ \mathcal{W},\ Q_1@(T + d_1), \ldots, Q_n@(T + d_n)\,]\ .$$

By consuming and creating facts, rewrite rules can increase and decrease the number of facts in configurations throughout a trace. However, in balanced MSR systems, the number of facts in configurations is constant throughout a trace.

2.1 Progressing Timed Systems

Following [14], we discuss a particular class of timed MSR systems, called *progressing timed MSR systems* (PTSes), in which only a finite number of actions can be carried out in a bounded time interval. This is a natural condition for many systems, similar similar to the *finite-variability assumption* used in the temporal logic and timed automata literature.

Definition 3 (Progressing Timed System). *A timed MSR system $\mathcal{T}$ is a* progressing timed MSR system (PTS) *if $\mathcal{T}$ is balanced and for all instantaneous rules $r \in \mathcal{T}$:*

i) Rule r creates at least one *fact with timestamp greater than the global time,* i.e.*, in (Eq. 2), $d_i \geq 1$ for at least one $i \in \{1, \ldots, n\}$;*

ii) Rule r consumes only *facts with timestamps in the past or at the current time,* i.e.*, in (Eq. 2), the set of constraints $\mathcal{C}$ contains the set*

$$\mathcal{C}_r = \{\, T \geq T'_i \mid F_i@T'_i,\ 1 \leq i \leq n \,\}\,.$$

For the sake of readability, from this point on we assume that for all rules r the set of their constraints implicitly contains the set $\mathcal{C}_r$, as shown in Definition 3, and do not always write $\mathcal{C}_r$ explicitly in our specifications.

The following rule, which denotes the action of a drone taking a photo of a point of interest, is an example of a rule in a PTS:

$$Time@T,\ P(I,X,Y)@T',\ Dr(Id,X,Y,E+1)@T \mid \{\, T' < T \,\} \longrightarrow Time@T,\ P(I,X,Y)@T,\ Dr(Id,X,Y,E)@(T+1)$$

The constraint $T' < T$ is used to prevent drones from repeatedly photographing the same point of interest at the same time to save energy.

The following result [14] establishes a bound on the number of instances of instantaneous rules appearing between two consecutive instances of $Tick$ rules in a trace of a PTS. This bound is then used to formalize the intuition that PTSes always move things forward.

Proposition 1. *Let $\mathcal{T}$ be a PTS, $\mathcal{S}_0$ an initial configuration and m the number of facts in $\mathcal{S}_0$. For all traces $\mathcal{P}$ of $\mathcal{T}$ starting from $\mathcal{S}_0$, let*

$$\mathcal{S}_i \longrightarrow_{Tick} \mathcal{S}_{i+1} \longrightarrow \cdots \longrightarrow \mathcal{S}_j \longrightarrow_{Tick} \mathcal{S}_{j+1}$$

be any subtrace of $\mathcal{P}$ with exactly two instances of the $Tick$ rule, one at the beginning and the other at the end. Then $j - i < m$. [14]

Proof. Let $\mathcal{P}$ be an arbitrary trace in $\mathcal{T}$ and

$$\mathcal{S}_i \longrightarrow_{Tick} \mathcal{S}_{i+1} \longrightarrow \cdots \longrightarrow \mathcal{S}_j \longrightarrow_{Tick} \mathcal{S}_{j+1}$$

an arbitrary subtrace of $\mathcal{P}$ with exactly two instances of the $Tick$ rule. All the rules between $Tick$ rules in the above subtrace are instantaneous.

Since $\mathcal{T}$ is a PTS, the application of any instantaneous rule creates at least one future fact and consumes at least one present or past fact. In other words, an application of an instantaneous rule reduces the total number of past and present facts in the configuration.

Since the system $\mathcal{T}$ is balanced, all the above configurations $\mathcal{S}_i, \ldots, \mathcal{S}_j$ have the same number of facts, m. Recall also that the fact $Time$ does not change whe the instantaneous rules are applied. Thus, since there are at most $m-1$ present or past facts different from $Time$ in any $\mathcal{S}_k$, $i < k \leq j$, a series of at most $m-1$ instantaneous rules can be applied between two $Tick$ rules. □

As per the above statement, in a PTS an unbounded number of instantaneous rules cannot be applied in a bounded interval of time. Hence, infinite traces in PTSes represent infinite time periods. In particular, there are no Zeno-type phenomena in traces of PTSes.

Proposition 2. *Let $\mathcal{T}$ be a PTS. All infinite traces of $\mathcal{T}$ are infinite time traces,* i.e.*, traces where time tends to infinity. [14]*

Proof. Assume that in some infinite trace $\mathcal{P}$ of a PTS $\mathcal{T}$ the current time does not exceed some value M. Then, as time advances by a single time unit, there are at most M time

ticks in $\mathcal{P}$. According to Proposition 1, there are at most $m-1$ instantaneous rules between any $Tick$ rule and the next $Tick$ rule in $\mathcal{P}$. Consequently, in total, there are at most $(M+1)\cdot(m-1)+M$ rules in $\mathcal{P}$, *i.e.*, $\mathcal{P}$ is a finite trace. Contradiction. □

Finally, notice that the PTS model has many carefully developed syntactic conditions, *e.g.*, balanced condition, the form of time constraints, the form of instantaneous rules (Eq. 2). As we have previously shown [19], relaxing any of these conditions leads to undecidability of important verification problems over finite traces. Clearly, these conditions are also needed for infinite traces. The additional challenge in allowing infinite traces is to represent arbitrarily large time periods. Our definition of PTS is a simple and elegant way to enforce this. Moreover, as we show in [14] and the technical report [15], it is still possible to specify many interesting examples with our PTS model and still prove the decidability of our verification problems involving infinite traces.

3 Quantitative Temporal Properties

Following [14], we begin the Section 3.1 by discussing critical configurations, a language used to define desirable properties of systems. This is a key concept in our framework, used to describe explicit timing constraints that a system should satisfy. In Section 3.2 we discuss lazy time sampling, which is a condition on traces that intuitively enforces that systems react at the expected time. Then in Section 3.3, we discuss a number of verification problems.

3.1 Critical Configurations and Compliant Traces

Critical configurations represent bad configurations that should be avoided by a system. *Critical configuration specification* is a set of pairs $\mathcal{CS} = \{\ \langle \mathcal{S}_1, \mathcal{C}_1\rangle, \ldots, \langle \mathcal{S}_n, \mathcal{C}_n\rangle\ \}$. Each pair $\langle \mathcal{S}_j, \mathcal{C}_j\rangle$ in $\mathcal{CS}$ is of the form $\langle\ \{F_1@T_1, \ldots, F_{p_j}@T_{p_j}\}, \mathcal{C}_j\ \rangle$, where $T_1, \ldots, T_{p_j}$ are time variables, $F_1, \ldots, F_{p_j}$ are facts (possibly containing variables) and $\mathcal{C}_j$ is a set of time constraints involving only the variables $T_1, \ldots, T_{p_j}$.

Given a critical configuration specification $\mathcal{CS}$, we classify a configuration $\mathcal{S}$ as *critical* w.r.t. $\mathcal{CS}$ if for some $1 \leq i \leq n$, there is a grounding substitution σ, such that $\mathcal{S}_i\sigma \subseteq \mathcal{S}$ and all constraints in $\mathcal{C}_i\sigma$ are satisfied. The substitution application ($\mathcal{S}\sigma$) is defined as usual [10], *i.e.*, by mapping time variables in $\mathcal{S}$ to natural numbers, nonce names to nonce names (renaming of nonces), and non-time variables to terms. Notice that nonce renaming is assumed, since the particular nonce name should not matter for classifying a configuration as critical. Nonce names cannot be specified in advance, since they are freshly generated in a trace, *i.e.*, during the execution of the process being modelled. Several examples illustrating these concepts are discussed in [14] and [15], along with proofs showing how the explicit time conditions are handled symbolically.

Definition 4 (Compliant Trace). *A trace $\mathcal{P}$ of a timed MSR system is* compliant *w.r.t. a given critical configuration specification $\mathcal{CS}$ if $\mathcal{P}$ does not contain any configuration that is critical w.r.t. $\mathcal{CS}$.*

Note that if the critical configuration specification is empty, no configuration is critical, *i.e.*, all traces are compliant.

3.2 Time Sampling

To define sensible quantitative verification properties, we need to assume some conditions on when the Tick rule is applicable. Otherwise, any MSR system allows traces containing only instances of $Tick$ rules:

$\mathcal{S}_1 \longrightarrow_{Tick} \mathcal{S}_2 \longrightarrow_{Tick} \mathcal{S}_3 \longrightarrow_{Tick} \mathcal{S}_4 \longrightarrow_{Tick} \cdots$

In such a trace, the system never acts to avoid critical configurations and would easily contain a critical configuration $\mathcal{S}_j$, related to some constraint $T > T' + d$, involving global time T and sufficiently large j.

Imposing a *time sampling* is one way to avoid such traces. Time sampling is used, for example, in the semantics of verification tools such as Real-Time Maude [25]. In particular, time sampling dictates when the $Tick$ rule must be applied and when it cannot be applied. Such a treatment of time is used for both dense and discrete times in searching and model checking timed systems.

Definition 5 (Lazy Time Sampling (l.t.s.)). *A (possibly infinite) trace $\mathcal{P}$ of a timed MSR system $\mathcal{T}$ uses* lazy time sampling *if for any occurrence of the $Tick$ rule $\mathcal{S}_i \longrightarrow_{Tick} \mathcal{S}_{i+1}$ in $\mathcal{P}$, no instance of any instantaneous rule in $\mathcal{T}$ can be applied to the configuration $\mathcal{S}_i$.*

In the lazy time sampling instantaneous rules are given a higher priority than the $Tick$ rule. In the remainder of this paper, we focus on the lazy time sampling. We leave it to future work to investigate whether similar results hold for other time sampling schemes.

3.3 Verification Problems

Four properties are discussed in this section: realizability and survivability from [14] and the new properties of reliability and recoverability. Figure 1 illustrates these properties, which we define below. Since the names of the properties sound similar in English, we also introduce one-letter names for the properties for better readability and differentiation.

The first property we discuss is realizability. It guarantees that the given system can achieve the assigned goal under the given time constraints and design specifications, *e.g.*, that drones can repeatedly collect up-to-date images of the sensitive locations.

Realizability is useful for increasing confidence in a specified system, since a system that is not realizable cannot accomplish the given tasks (specified by a critical specification) and the designer would therefore have to reformulate it.

However, if a system is shown to be realizable, the trace, $\mathcal{P}$, that proves realizability could also provide insights into the sequence of actions that lead to accomplishment of the specified tasks. This can be used to refine the specification and reduce possible non-determinism.

Open distributed systems are inherently non-deterministic due to, *e.g.*, the influence of the environment. Therefore, it is important to know whether the system can avoid critical configurations despite non-determinism. We call this property *survivability.*

Definition 6 (Realizability / Z property). *A timed MSR system $\mathcal{T}$ satisfies* realizability *w.r.t. an initial configuration $\mathcal{S}_0$, a critical configuration specification $\mathcal{CS}$ and the l.t.s. if there* exists *a compliant infinite time trace from $\mathcal{S}_0$ that uses the l.t.s.* [8] *[14]*

The Z property of a timed MSR $\mathcal{T}$ w.r.t. $\mathcal{S}_0, \mathcal{CS}$ and l.t.s. can be expressed using the formula:

$$F_Z(\mathcal{T}, \mathcal{S}_0) := \exists\, t \in \mathsf{T}^{\mathcal{T},\mathcal{S}_0}.[t \in \mathsf{T}^{\mathcal{T}}_{time} \cap \mathsf{T}^{\mathcal{T}}_{lts} \cap \mathsf{T}^{\mathcal{T}}_{c}],$$

where $\mathsf{T}^{\mathcal{T},\mathcal{S}_0}$ is the set of all traces of $\mathcal{T}$ starting from $\mathcal{S}_0$, $\mathsf{T}^{\mathcal{T}}_{time}$ is the set of all infinite time traces of $\mathcal{T}$, $\mathsf{T}^{\mathcal{T}}_{lts}$ is the set of all traces of $\mathcal{T}$ that use the l.t.s. and $\mathsf{T}^{\mathcal{T}}_{c}$ is the set of all traces of $\mathcal{T}$ compliant w.r.t. $\mathcal{CS}$.

Definition 7 (Survivability / S property). *A timed MSR $\mathcal{T}$ satisfies* survivability *w.r.t. an initial configuration $\mathcal{S}_0$, a critical configuration specification $\mathcal{CS}$ and the l.t.s. if it satisfies realizability with respect to $\mathcal{S}_0$, $\mathcal{CS}$, and the l.t.s. and if* all *infinite time traces from $\mathcal{S}_0$ that use the l.t.s. are compliant. [14]*

Using the above notation, the S property of a timed MSR $\mathcal{T}$ can be expressed with:

$$F_S(\mathcal{T}, \mathcal{S}_0) := F_Z(\mathcal{T}, \mathcal{S}_0) \wedge \forall\, t \in \mathsf{T}^{\mathcal{T},\mathcal{S}_0}.[t \in \mathsf{T}^{\mathcal{T}}_{time} \cap \mathsf{T}^{\mathcal{T}}_{lts} \Rightarrow t \in \mathsf{T}^{\mathcal{T}}_{c}].$$

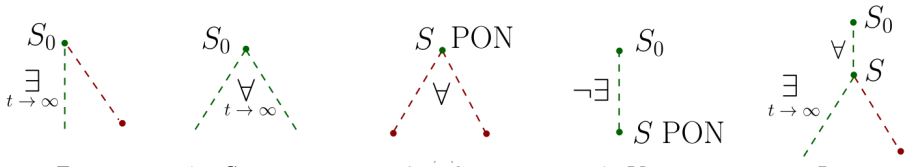
Although survivability is a desirable property, much more so than realizability, it can sometimes be a rather severe requirement for a system, or even unachievable. Hence, when designing a system, one may want to compromise and consider less demanding properties. For example, one may want to avoid configurations that appear as "dead-ends", *i.e.*, configurations that necessarily lead to critical configurations. We call such configurations *points-of-no-return*. For example, drones should not fly so far that it is no longer possible to reach a recharging station due to energy consumption.

Definition 8 (Point-of-No-Return). *Given a timed MSR system $\mathcal{T}$, a configuration $\mathcal{S}$ is called a* point-of-no-return *with respect to a critical configuration specification $\mathcal{CS}$, and the l.t.s. if $\mathcal{S}$ is* not *critical w.r.t. $\mathcal{CS}$, and if* all *infinite traces of $\mathcal{T}$ starting with $\mathcal{S}$ and using the l.t.s. are not compliant w.r.t. $\mathcal{CS}$.*

The set of all configurations that are points-of-no-return of a timed MSR $\mathcal{T}$, $\mathsf{C}^{\mathcal{T}}_{pon}$, can be described as $\mathsf{C}^{\mathcal{T}}_{pon} := \{\mathcal{S} \mid \mathcal{S} \notin \mathsf{C}^{\mathcal{T}}_{cr} \wedge \forall\, t.[t \in \mathsf{T}^{\mathcal{T},\mathcal{S}} \cap \mathsf{T}^{\mathcal{T}}_{\infty} \cap \mathsf{T}^{\mathcal{T}}_{lts} \Rightarrow t \notin \mathsf{T}^{\mathcal{T}}_{c}]\}$, where $\mathsf{C}^{\mathcal{T}}_{cr}$ is the set of all critical configurations of $\mathcal{T}$ and $\mathsf{T}^{\mathcal{T}}_{\infty}$ is the set of all infinite traces of $\mathcal{T}$.

There exists no compliant infinite trace from a point-of-no-return that uses the l.t.s. A point-of-no-return itself is not critical, but must eventually lead to a critical configuration on every infinite trace that uses the l.t.s. Therefore, points-of-no-return should be avoided when searching for (infinite) compliant traces.

[8] For simplicity, in the rest of the paper, for properties of systems and configurations, we will not always explicitly state the critical configuration specification, initial configuration, and/or time sampling with respect to which the property is considered. For example, when it is clear from the context, we simply say that a system satisfies Z property or that it is *realizable*.
Also, when for a property of an MSR $\mathcal{T}$ we only consider traces that use the l.t.s., we also say that $\mathcal{T}$ *uses the lazy time sampling*.



(a) Z property (b) S property (c) point-of-no-return (d) V property (e) L property

Fig. 1. Illustration of properties of (a) realizability, (b) survivability, (d) recoverability, and (e) reliability, as well as configurations that are a point-of-no-return (c). Green lines represent compliant traces that use the l.t.s., while red lines represent traces that use the l.t.s. but are not compliant. Red circles represent critical configurations, while green circles are not critical. Quantification marked with $t \to \infty$ denotes quantification over infinite time traces.

Remark 2. A point-of-no-return represents the system that still satisfies the required conditions, but it will inevitably fall into a bad state where this is no longer the case. Therefore, to better distinguish between points-of-no-return and critical configurations, the condition that a point-of-no-return is not critical is included in the definition.

We use the notion of points-of-no-return, to introduce new properties of our systems. Configurations that are points-of-no-return should be avoided. For example, a drone may enter an area where it may end up with empty batteries due to frequent high winds. Such points should be avoided.

Definition 9 (Recoverability / V property). *A timed MSR system $\mathcal{T}$ satisfies* recoverability *with respect to an initial configuration $\mathcal{S}_0$, a critical configuration specification $\mathcal{CS}$, and the l.t.s. if it satisfies realizability with respect to $\mathcal{S}_0$, $\mathcal{CS}$, and the l.t.s. and if no point-of-no-return is reachable from $\mathcal{S}_0$ on a compliant trace that uses the l.t.s. That is, if a configuration $\mathcal{S}$ is reachable from $\mathcal{S}_0$ on a compliant trace that uses the l.t.s.,then $\mathcal{S}$ is not a point-of-no-return.*

The V property of a timed MSR $\mathcal{T}$ can be expressed with the following formula:

$$F_V(\mathcal{T}, \mathcal{S}_0) := F_Z(\mathcal{T}, \mathcal{S}_0) \wedge [\forall t \in \mathsf{T}^{\mathcal{T},\mathcal{S}_0} \cap \mathsf{T}_c^{\mathcal{T}} \cap \mathsf{T}_{lts}^{\mathcal{T}}. \forall \mathcal{S} \in t. \mathcal{S} \notin \mathsf{C}_{pon}^{\mathcal{T}}].$$

In fact, with the V property we want to ensure that all finite compliant traces from the initial configuration that use the l.t.s. can be extended to infinite compliant traces that use the l.t.s.

Next, with the reliability property we want to ensure that as long as one follows a compliant trace, there is a way to extend the trace to a compliant infinite time trace. In our drone scenario, a reliable system should be designed so that as long as the drones follow instructions, including rules for flying in high winds, there is always a way for the drones to avoid critical configurations.

Definition 10 (Reliability / L property). *A timed MSR system $\mathcal{T}$ satisfies* reliability *w.r.t. an initial configuration $\mathcal{S}_0$, a critical configuration specification $\mathcal{CS}$ and the l.t.s. if it satisfies realizability with respect to $\mathcal{S}_0$, $\mathcal{CS}$, and the l.t.s. and if for* any *configuration $\mathcal{S}$ reachable from $\mathcal{S}_0$ on a* compliant *trace that uses the l.t.s., there exists a compliant infinite time trace from $\mathcal{S}$ that uses the l.t.s.*

The L property of a timed MSR $\mathcal{T}$ can be expressed with the following formula:

$$F_L(\mathcal{T}, \mathcal{S}_0) := F_Z(\mathcal{T}, \mathcal{S}_0) \wedge \\ [\forall t \in \mathsf{T}^{\mathcal{T},\mathcal{S}_0} \cap \mathsf{T}_c^{\mathcal{T}} \cap \mathsf{T}_{lts}^{\mathcal{T}}. \forall \mathcal{S} \in t. \exists t' \in \mathsf{T}^{\mathcal{T},\mathcal{S}}. \, t' \in \mathsf{T}_c^{\mathcal{T}} \cap \mathsf{T}_{lts}^{\mathcal{T}} \cup \mathsf{T}_{time}^{\mathcal{T}}].$$

A timed MSR system that satisfies the L property represents a system that is always able to avoid points-of-no-return. Such a system satisfies the Z property, but it may not satisfy the S property. Indeed, the class of systems satisfying the Z property is a proper superclass of the class of systems satisfying the L property. Systems satisfying the L property also satisfy the V property, while the class of systems satisfying the V property is a proper superclass of the class of systems satisfying the S property. We present these results in Section 4, for general MSR systems and PTSes.

Some of these relations clearly follow since the Z property is included in the definitions of the other properties. Although this is an intuitive approach where for the more demanding properties we only consider systems that can accomplish their tasks, *i.e.*, satisfy the Z property, given a non-critical initial configuration, the Z property would follow from the V property and from the L property anyway. Similarly, by including the Z property, we avoid the vacuously survivable systems with no infinite time traces that use the l.t.s. For details see [15, Remark 3].

Time-Bounded Versions of Verification Problems

Motivated by bounded model checking, we also investigate the time-bounded versions of the above problems. Instead of infinite traces, in time-bounded versions of verification problems we consider traces with a fixed number of occurrences of Tick rules.

Definition 11 (n-Time Realizability / n-Z property). *A timed MSR system $\mathcal{T}$ satisfies* n-time realizability *w.r.t. the l.t.s., a critical configuration specification $\mathcal{CS}$, and an initial configuration $\mathcal{S}_0$ if there exists a compliant trace, $\mathcal{P}$, from $\mathcal{S}_0$ that uses the l.t.s. such that global time advances by exactly n time units in $\mathcal{P}$. [14]*

Definition 12 (n-Time Survivability / n-S property). *A timed MSR system $\mathcal{T}$ satisfies* n-time survivability property *w.r.t. the l.t.s., a critical configuration specification $\mathcal{CS}$ and an initial configuration $\mathcal{S}_0$ if it satisfies n-Z property and if all traces with exactly n instances of the $Tick$ rule starting with $\mathcal{S}_0$ and using the l.t.s. are compliant. [14]*

Analogously, we define the n-time bounded version of the reliability problem. We consider all compliant traces covering *at most* n time units, and extend them to compliant traces over *exactly* n time units.

Definition 13 (n-Time Reliability / n-L property). *A timed MSR system $\mathcal{T}$ satisfies* n-time reliability *w.r.t. an initial configuration $\mathcal{S}_0$, a critical configuration specification $\mathcal{CS}$, and the l.t.s. if it satisfies n-time realizability with respect to $\mathcal{S}_0$, $\mathcal{CS}$, and the l.t.s. and if for any configuration $\mathcal{S}$, reachable from $\mathcal{S}_0$ on a compliant trace $\mathcal{P}$ that uses the l.t.s. and has at most n instances of the $Tick$ rule, there exists a trace $\mathcal{P}'$ that uses the l.t.s. such that $\mathcal{P}'$ extends $\mathcal{P}$, $\mathcal{P}'$ is compliant, and $\mathcal{P}'$ has exactly n instances of the $Tick$ rule.*

Since the notion of a point-of-no-return is defined to be inseparable from infinite traces, the time-bounded version of the recoverability system problem makes little sense. Hence, we do not consider this problem separately.

4 Relations Among Properties of Timed MSR

In this section we formally relate all the different properties discussed in Section 3.3. To compare these properties, we review in the accompanying technical report [15] the machinery introduced in our previous work [19] called δ-representations. This machinery is also used in Section 5 to obtain complexity results for the verification problems.

In the accompanying technical report [15] we show that the Z property implies the n-Z property, as expected, but also that for a sufficiently large n, the converse implication also holds. Namely, in a trace with a sufficiently large number of $Tick$ rules, the same δ-representation is repeated, forming a loop that can be repeated to obtain an infinite time trace showing the Z property. The same implications hold for the other properties.

4.1 Relations Among Different Properties of Timed MSR and PTS

We now relate different properties defined over infinite traces. We can distinguish all these properties for general timed MSR, but only some for PTSes, as stated below.

Proposition 3. *Let $\mathcal{T}$ be a timed MSR system that uses the l.t.s., $\mathcal{S}_0$ an initial configuration and $\mathcal{CS}$ a critical configuration specification.*

If $\mathcal{T}$ satisfies the L property, then $\mathcal{T}$ satisfies the V property.

If $\mathcal{T}$ satisfies the V property, then $\mathcal{T}$ does not necessarily satisfy the L property.

Proof. Let $\mathcal{T}$ be a timed MSR system that satisfies the L property.
Assume $\mathcal{T}$ does not satisfy the V property. Then, since $\mathcal{T}$ satisfies the Z property, there is a compliant trace from $\mathcal{S}_0$ to some point-of-no-return $\mathcal{S}_P$ that uses the l.t.s. Since $\mathcal{T}$ satisfies the L property, there is a compliant infinite time trace from $\mathcal{S}_P$ that uses the l.t.s. As $\mathcal{S}_P$ is a point-of-no-return, this contradicts the notion of point-of-no-return.

We give an example of a timed MSR system, $\mathcal{T}$, that satisfies the V property, but does not that satisfy the L property.

Let $\mathcal{S}'_0 = \{Time@0, C@1\}$, $\mathcal{CS}' = \emptyset$, and let $\mathcal{T}'$ contain only the following instantaneous rules:

$$Time@T,\ C@T' \mid T' \leq T \longrightarrow Time@T,\ D@T \tag{3a}$$
$$Time@T,\ C@T' \mid T' \leq T \longrightarrow Time@T,\ A@T \tag{3b}$$
$$Time@T,\ A@T' \longrightarrow Time@T,\ B@T \tag{3c}$$
$$Time@T,\ B@T' \longrightarrow Time@T,\ A@T \tag{3d}$$

The system $\mathcal{T}'$ satisfies the Z property since there is a compliant infinite time trace from $\mathcal{S}'_0$ that uses l.t.s.:

$$\begin{aligned} &Time@0,\ C@1 \longrightarrow_{Tick} Time@1,\ C@1 \longrightarrow_{(3a)} Time@1,\ D@1 \longrightarrow_{Tick} \\ &\quad Time@2,\ D@1 \longrightarrow_{Tick} Time@3,\ D@1 \longrightarrow_{Tick} Time@4,\ D@2 \longrightarrow_{Tick} \ \ldots \end{aligned} \tag{4}$$

There is only one other infinite trace form $\mathcal{S}_0$ that uses the l.t.s.:

$$\begin{array}{l} Time@0,\, C@1 \;\longrightarrow_{Tick}\; Time@1,\, C@1 \;\longrightarrow_{(3b)}\; Time@1,\, A@1 \;\longrightarrow_{(3c)} \\ \quad Time@1,\, B@1 \;\longrightarrow_{(3d)}\; Time@1,\, A@1 \;\longrightarrow_{(3c)}\; Time@1,\, B@2 \;\longrightarrow_{(3d)}\; \ldots \end{array} \tag{5}$$

Its subtrace obtained from $\mathcal{S}_0$ by applying the $Tick$ rule followed by the rule $(3b)$ reaches the configuration $Time@1, A@1$. This subtrace is compliant but it cannot be extended to a compliant infinite time trace that uses the l.t.s. Hence, $\mathcal{T}'$ does not satisfy the L property.

However, $\mathcal{T}'$ trivially satisfies the V property since there are no critical configurations and, hence, no points-of-no-return. □

The properties of timed MSR defined in Section 3.3 involve infinite time traces that use the l.t.s. Recall that for any given PTS $\mathcal{T}$ and any configuration $\mathcal{S}$, there exists an infinite time trace of $\mathcal{T}$ that starts with $\mathcal{S}$ and uses the l.t.s.

Although V and L are different properties of timed MSR systems in general, it turns out that for the class of PTSes these properties coincide.

Proposition 4. *Let $\mathcal{T}$ be a PTS that uses the l.t.s., $\mathcal{S}_0$ an initial configuration, and $\mathcal{CS}$ a critical configuration specification.*

System $\mathcal{T}$ satisfies the L property iff $\mathcal{T}$ satisfies the V property.

Proof. Since a PTS is a timed MSR system, it follows from Proposition 3 that a PTS, which satisfies the L property, also satisfies the V property.

Assume that a PTS $\mathcal{T}$ does not satisfy the L property. If $\mathcal{S}_0$ is critical, then $\mathcal{T}$ does not satisfy the Z property and consequently, does not satisfy the V property. If $\mathcal{S}_0$ is not critical, there is a compliant trace from $\mathcal{S}_0$ to some configuration $\mathcal{S}_1$ that uses the l.t.s. which cannot be extended to a compliant infinite time trace that uses the l.t.s.

Then, $\mathcal{S}_1$ is a point-of-no-return. Namely, if P is an infinite trace from $\mathcal{S}_1$ that uses the l.t.s., by Proposition 2, P is an infinite time trace that uses the l.t.s. Then, P is not compliant. Since the point-of-no-return $\mathcal{S}_1$ is reachable from $\mathcal{S}_0$ on a compliant trace using the l.t.s., $\mathcal{T}$ does not satisfy the V property. □

We show that the remaining properties are different even for PTSes. Furthermore, we provide the relations among the properties for PTSes and for timed MSR systems in general. We first show that L and S are different properties of PTSes, and, consequently, different properties of timed MSR systems.

Proposition 5. *Let $\mathcal{T}$ be a PTS that uses the l.t.s., $\mathcal{S}_0$ an initial configuration and $\mathcal{CS}$ a critical configuration specification.*

If $\mathcal{T}$ satisfies the S property, then $\mathcal{T}$ satisfies the L property.

If $\mathcal{T}$ satisfies the L property, it may not satisfy the S property.

Proof. Assume that $\mathcal{T}$ satisfies the S property, but does not satisfy the L property. Then, since $\mathcal{T}$ satisfies the Z property, there exists a compliant trace, $\mathcal{P}$, from $\mathcal{S}_0$ to some configuration $\mathcal{S}_1$ that cannot be extended to a compliant infinite time trace that uses the l.t.s. Let $\mathcal{P}'$ be an infinite time trace which is an extension of $\mathcal{P}$ that uses the l.t.s. Such a trace $\mathcal{P}'$ exists due to Proposition 2, but it is not compliant.

Since $\mathcal{T}$ satisfies the S property, all infinite time traces from $\mathcal{S}_0$ that use the l.t.s. are compliant, including $\mathcal{P}'$. Contradiction.

The following example of a PTS satisfies the L property, but does not satisfy the S property.
Let $\mathcal{S}_0 = \{Time@0, A@0, B@0\}$, $\mathcal{CS} = \{ \langle \{B@T, D@T'\}, \emptyset \rangle\}$ and let PTS $\mathcal{T}$ contain only the following instantaneous rules:

$$Time@T,\ A@T',\ B@T'' \mid \{T' \leq T, T'' \leq T\} \longrightarrow Time@T,\ B@T'',\ C@(T+1) \quad \text{(6a)}$$
$$Time@T,\ A@T',\ B@T'' \mid \{T' \leq T\} \longrightarrow Time@T,\ B@T'',\ D@(T+1) \quad \text{(6b)}$$
$$Time@T,\ B@T',\ C@T'' \mid \{T' \leq T, T'' \leq T\} \longrightarrow Time@T,\ A@T,\ B@(T+1) \quad \text{(6c)}$$

The following trace from $\mathcal{S}_0$ uses the l.t.s. and is not compliant:

$$Time@0, A@0, B@0 \longrightarrow_{(6b)} Time@0, B@0, D@1\ .$$

Hence, $\mathcal{T}$ does not satisfy the S property.

To show that $\mathcal{T}$ satisfies the L property, we first show that $\mathcal{T}$ satisfies the Z property. The following trace from $\mathcal{S}_0$ is a compliant infinite time trace that uses the l.t.s.:

$$Time@0, A@0, B@0 \longrightarrow_{(6a)} Time@0, B@0, C@1 \longrightarrow_{Tick} \longrightarrow_{Tick} Time@1, B@0, C@1 \longrightarrow_{(6c)} Time@1, A@1, B@2 \longrightarrow_{Tick} \longrightarrow_{Tick} Time@2, A@1, B@2 \longrightarrow_{(6a)} Time@2, B@2, C@3 \longrightarrow_{Tick} \ldots$$

Next, assume $\mathcal{P}$ is a compliant trace from $\mathcal{S}_0$ to some $\mathcal{S}_1$ that uses the l.t.s. Then $\mathcal{P}$ does not contain rule (6b), which always results in a critical configuration. Hence, only rules (6a), (6c) and $Tick$ are used in $\mathcal{P}$, so $\mathcal{S}_1$ is either $\{Time@t,\ A@t',\ B@t''\}$ or $\{Time@t,\ B@t',\ C@t''\}$. Using only the rules (6a), (6c) and $Tick$, the trace $\mathcal{P}$ can be extended to a compliant infinite time trace that uses the l.t.s. Hence, $\mathcal{T}$ satisfies the L property. □

However, the above does not hold for general MSR systems, *i.e.*, MSR systems that satisfy the S property do not necessarily satisfy the L property.

Proposition 6. *Let $\mathcal{T}$ be a timed MSR that uses the l.t.s., $\mathcal{S}_0$ an initial configuration and $\mathcal{CS}$ a critical configuration specification.*
If $\mathcal{T}$ satisfies the S property, it may not satisfy the L property.
If $\mathcal{T}$ satisfies the L property, it may not satisfy the S property.

Proof. Let $\mathcal{T}'$, $\mathcal{S}'_0$ and $\mathcal{CS}'$ be as specified in the proof of Proposition 3. Recall that $\mathcal{T}'$ does not satisfy the L property.

The system $\mathcal{T}'$ satisfies the S property. Namely, there are only two infinite traces from $\mathcal{S}'_0$ that use the l.t.s., traces (4) and (5) specified in the proof of Proposition 3. However, trace (5) is not an infinite time trace, so there is only one infinite time trace from $\mathcal{S}'_0$ that uses the l.t.s., trace (4). Therefore, since trace (4) is compliant, $\mathcal{T}'$ satisfies the S property.

By Proposition 5 there is a PTS, and therefore an MSR, that satisfies the L property but does not satisfy the S property. □

Next, we show how the V property relates to the Z property.

Proposition 7. *Let $\mathcal{T}$ be a timed MSR that uses the l.t.s., $\mathcal{S}_0$ an initial configuration, and $\mathcal{CS}$ a critical configuration specification.*

If $\mathcal{T}$ satisfies the V property, then $\mathcal{T}$ satisfies the Z property.

A system $\mathcal{T}$ that satisfies the Z property may not satisfy the V property.

Proof. Assume $\mathcal{T}$ satisfies the V property. Then, $\mathcal{T}$ satisfies the Z property by definition.

We prove the other statement by providing an example of a PTS that satisfies the Z property, but does not satisfy the V property. Let $\mathcal{S}_0'' = \{Time@0, A@0\}$, $\mathcal{CS}'' = \{ \langle \{D@T\}, \emptyset \rangle \}$ and let PTS $\mathcal{T}''$ contain only the following instantaneous rules:

$$Time@T,\ A@T' \mid \{T' \leq T\} \longrightarrow Time@T,\ B@(T+1) \tag{7a}$$
$$Time@T,\ A@T' \mid \{T' \leq T\} \longrightarrow Time@T,\ C@(T+1) \tag{7b}$$
$$Time@T,\ B@T' \mid \{T' \leq T\} \longrightarrow Time@T,\ A@(T+1) \tag{7c}$$
$$Time@T,\ C@T' \mid \{T' \leq T\} \longrightarrow Time@T,\ D@(T+1) \tag{7d}$$

The following trace, which uses the l.t.s., shows the Z property of $\mathcal{T}''$:

$$Time@0,\ A@0 \longrightarrow_{(7a)} Time@0,\ B@1 \longrightarrow_{Tick} Time@1,\ B@1 \longrightarrow_{(7c)}$$
$$Time@1,\ A@2 \longrightarrow_{Tick} Time@2,\ A@2 \longrightarrow_{(7a)} Time@2,\ B@3 \longrightarrow_{Tick} \ldots$$

The configuration $\widetilde{\mathcal{S}} = \{Time@0, C@1\}$ is reachable from $\mathcal{S}_0''$ by a compliant trace that uses the l.t.s.: $Time@0,\ A@0 \longrightarrow_{(7b)} Time@0,\ C@1$. $\widetilde{\mathcal{S}}$ is a point-of-no-return as rule (7d) is the only instantaneous rule that can be applied after a $Tick$, so all infinite traces from $\widetilde{\mathcal{S}}$ that use the l.t.s. contain the critical configuration $\{Time@1, D@2\}$.

Since $\widetilde{\mathcal{S}}$ is a point-of-no-return, $\mathcal{T}''$ does not satisfy the V property. □

Using transitivity of the subset relation, we can infer relations among all our properties for both PTSes and timed MSR systems in general. We summarize our results in the following corollaries.

Corollary 1. *Let ${}_{reali}Z^{MSR}_{ability}$, ${}_{re}L^{MSR}_{iability}$, ${}_{reco}V^{MSR}_{erability}$ and $S^{MSR}_{urvivability}$ be the classes of timed MSR systems satisfying the Z, L, V and S properties, respectively, w.r.t. the l.t.s. Then, the following relations hold:*

$$S^{MSR}_{urvivability} \neq {}_{re}L^{MSR}_{iability} \subset {}_{reco}V^{MSR}_{erability} \subset {}_{reali}Z^{MSR}_{ability}$$

Proof. The statement follows directly from the Propositions 6, 3, and 7. □

Corollary 2. *Let ${}_{reali}Z^{PTS}_{ability}$, ${}_{re}L^{PTS}_{iability}$, ${}_{reco}V^{PTS}_{erability}$ and $S^{PTS}_{urvivability}$ be the classes of PTSes satisfying the Z, L, V and S properties, respectively, w.r.t. the l.t.s. Then the following proper subset relations hold:*

$$S^{PTS}_{urvivability} \subset {}_{re}L^{PTS}_{iability} = {}_{reco}V^{PTS}_{erability} \subset {}_{reali}Z^{PTS}_{ability}$$

Proof. The statement follows directly from the Propositions 5 and 4, and the proof of proposition 7. □

Corollary 3. *Let ${}_{reali}nZ^{PTS}_{ability}$, ${}_{re}nL^{PTS}_{iability}$ and $nS^{PTS}_{urvivability}$ be the classes of PTSes satisfying the n-Z, n-L and n-S properties, respectively, w.r.t. the l.t.s. Then, the following proper subset relations hold:*

$$n\mathcal{S}^{PTS}_{urvivability} \subset {}_{re}nL^{PTS}_{iability} \subset {}_{reali}nZ^{PTS}_{ability}.$$

Proof. Let a PTS $\mathcal{T}$ satisfy the n-S property. We check that $\mathcal{T}$ satisfies the n-L property. Let $\mathcal{S}$ be a configuration that is reachable from $\mathcal{S}_0$ on a compliant trace $\mathcal{P}$ that uses the l.t.s. and has at most n instances of the $Tick$ rule. Since $\mathcal{T}$ is a PTS, only a bounded number of instantaneous rules can be applied before a $Tick$ rule appears in a trace that uses the l.t.s. (Proposition 1). Hence, the trace $\mathcal{P}$ can be extended to a compliant trace $\mathcal{P}'$ that contains exactly n instances of the $Tick$ rule and uses the l.t.s. Since $\mathcal{T}$ satisfies the n-S property, $\mathcal{P}'$ is compliant. Consequently, $\mathcal{T}$ satisfies the n-L property.

Now, let $\mathcal{T}$ satisfy the n-L property. Then, the trivial trace of length 1 from $\mathcal{S}_0$ (containing only $\mathcal{S}_0$) can be extended to a compliant trace $\mathcal{P}'$ that contains exactly n instances of the $Tick$ rule and uses the l.t.s. Hence, $\mathcal{T}$ satisfies the n-Z property.

To show that the inclusions are proper, we give examples of PTSes that satisfy one, but not the other property. The PTS given in the proof of Proposition 7 is an example of a system that satisfies the n-Z property, $\forall n > 0$, which does not satisfy even the 1-S property. Similarly, the PTS given in the proof of Proposition 5 satisfies the n-L property, $\forall n > 0$, but it does not even satisfy the 1-S property. □

5 Complexity Results for PTSes

In this section we investigate the complexity of the verification problems defined in Section 3.3 for PTSes. Recall that the L and V properties for PTSes coincide.

5.1 PSPACE-Completeness of Verification Problems for PTSes

For the Z problem and the S problem, PSPACE-completeness for PTSes was proved in [14, 16]. Here we show PSPACE-completeness of the L problem for PTSes, that is for the problem of deciding whether a given PTS satisfies the L property.

Let Σ be a finite alphabet, $\mathcal{T}$ a PTS, $\mathcal{S}_0$ an initial configuration, m the number of facts in $\mathcal{S}_0$, $\mathcal{CS}$ a critical configuration specification, k an upper-bound on the size of the facts, and D_{max} an upper-bound on the numerical values in $\mathcal{S}_0, \mathcal{T}$, and $\mathcal{CS}$. Let the functions $\mathcal{N}, \mathcal{X}$, and $\mathcal{L}$ run in Turing space bounded by a polynomial in $m, k, \log_2(D_{max})$ and return 1, respectively, when a rule in $\mathcal{T}$ is applicable to a given δ-representation, when a δ-representation is critical w.r.t. $\mathcal{CS}$, and when a $Tick$ rule should be applied to the given δ-representation using the l.t.s.

Proposition 8 (L problem for PTSes is PSPACE hard).
The L problem for PTSes that use the l.t.s. is PSPACE-hard.

Proof. The Z problem is an instance of the problem of checking whether a configuration is not a point-of-no-return. Recall that a system satisfies the Z property if there exists a compliant infinite time trace $\mathcal{P}$ from the initial configuration in which global time tends to infinity. Since $\mathcal{T}$ is progressing, we obtain the condition on time (time tends to infinity) from Proposition 2. Indeed, a system satisfies the Z property if and only if the initial configuration is not a point-of-no-return. Since PSPACE and co-PSPACE are the same complexity class and the Z property is PSPACE-hard, the problem of determining whether a configuration is a point-of-no-return is PSPACE-hard.

Since the L property problem comprises checking whether a configuration is a point-of-no-return, it is PSPACE-hard. □

Proposition 9 (L problem for PTSes is in PSPACE).
For a PTS $\mathcal{T}$ assume $\Sigma, \mathcal{S}_0, m, \mathcal{CS}, k, D_{max}$, $\mathcal{N}, \mathcal{X}$, and $\mathcal{L}$ as described above.

There is an algorithm that, given an initial configuration $\mathcal{S}_0$, decides whether $\mathcal{T}$ satisfies the L property with respect to the l.t.s.,$\mathcal{CS}$, and $\mathcal{S}_0$ and the algorithm runs in a space bounded by a polynomial in m, k and $log_2(D_{max})$.

Proof. We first propose an algorithm that, for a fixed system $\mathcal{T}$ and a fixed critical configuration specification $\mathcal{CS}$, checks whether some δ-representation corresponds to a configuration that is a point-of-no-return w.r.t. $\mathcal{T}$ and $\mathcal{CS}$.

Let $REAL$ denote the Z problem PSPACE algorithm over δ-representations (see proof of [16, Theorem 1] for details). When given δ-representation $\mathcal{W}$, as input, the algorithm $REAL(\mathcal{W})$ returns ACCEPT if and only if there is an infinite time trace of $\mathcal{T}$ that starts with $\mathcal{W}$, uses the l.t.s., and is compliant w.r.t. $\mathcal{CS}$, and it runs in polynomial space w.r.t. the given parameters, m, k and $log_2(D_{max})$. Since PSPACEand co-PSPACE are the same complexity class, we switch the ACCEPT and FAIL and obtain a deterministic algorithm $NOTREAL$ that runs in polynomial space w.r.t. m, k and $log_2(D_{max})$. $NOTREAL(\mathcal{W})$ accepts if and only if there is no compliant infinite time trace from $\mathcal{W}$ that uses the l.t.s. Then, using $NOTREAL$ we construct the algorithm PON that checks whether the given δ-representation $\mathcal{W}$ corresponds to a point-of-no-return. Let PON be the following algorithm, which takes a δ-representation $\mathcal{W}$ as input:

1. If $\mathcal{X}(W) = 1$, *i.e.*, if W represents a critical configuration, then return FAIL, otherwise continue;
2. If $NOTREAL(W) = 1$, *i.e.*, if W represents a point-of-no-return, then return ACCEPT, otherwise return FAIL.

When given δ-representation $\mathcal{W}$, as input, the algorithm $PON(\mathcal{W})$ accepts if and only if $\mathcal{W}$ is a δ-representation of a point-of-no-return w.r.t. $\mathcal{T}$ and $\mathcal{CS}$. Since $\mathcal{X}$, and $NOTREAL$ run in the polynomial space w.r.t. m, k and $log_2(D_{max})$, PON is a deterministic algorithm that also runs in such a polynomial space.

Next, we check that for any configuration $\mathcal{S}$ reachable from $\mathcal{S}_0$ using the l.t.s., there is a compliant infinite time trace from $\mathcal{S}$, *i.e.*, that $\mathcal{S}$ is not a point-of-no-return. The following algorithm accepts when no point-of-no-return is reachable from $\mathcal{S}_0$ in $\mathcal{T}$ on a compliant trace that uses the l.t.s., and fails otherwise. It begins with $i = 0$ and W_0 set to be the δ-representation of $\mathcal{S}_0$, and iterates the following sequence of operations:

1. If W_i represents a critical configuration, *i.e.*, if $\mathcal{X}(W_i) = 1$, then return FAIL, otherwise continue;
2. If W_i represents a point-of-no-return, *i.e.*, if $PON(W_i) = 1$, then return FAIL, otherwise continue;
3. If $i > L_\Sigma(m, k, D_{max})$, then ACCEPT; else continue;
4. If $\mathcal{L}(W_i) = 1$, then replace W_i by W_{i+1} obtained from W_i by applying the $Tick$ rule; Otherwise non-deterministically guess an instantaneous rule, r, from $\mathcal{T}$ applicable to W_i, *i.e.*, such a rule r that $\mathcal{N}(r, W_i) = 1$. If so, replace W_i with the δ-representation W_{i+1} resulting from applying the rule r to the δ-representation W_i. Otherwise, continue;
5. Set $i = i + 1$.

Since PON, $\mathcal{N}$, $\mathcal{X}$, and $\mathcal{L}$ run in Turing space bounded by a polynomial in m, k and $\log_2(D_{max})$, the above algorithm runs in deterministic polynomial space. □

The following result follows directly from Propositions 8 and 9.

Theorem 1. *For a PTS $\mathcal{T}$ assume $\Sigma, \mathcal{S}_0, m, \mathcal{CS}, k, D_{max}$, $\mathcal{N}, \mathcal{X}$, and $\mathcal{L}$ as described at the beginning of Section 5.1. The L problem for PTSes that use the l.t.s. is PSPACE-complete.*

Complexity results for the time bounded versions of the Z and S problems were also obtained in [14]. It was shown that the n-Z problem is NP-complete and that the n-S problem is in the Δ_2^p class of the polynomial hierarchy, a subclass of PSPACE. The technical report accompanying this paper [15] also discusses the complexity result for the time bounded version of the L problem.

Theorem 2. *For a PTS $\mathcal{T}$ assume $\Sigma, \mathcal{S}_0, m, \mathcal{CS}, k, D_{max}$, $\mathcal{N}, \mathcal{X}$, and $\mathcal{L}$ as described at the beginning of Section 5.1.*

The n-Z problem for $\mathcal{T}$ w.r.t. the l.t.s., $\mathcal{CS}$, and $\mathcal{S}_0$ is NP-complete with input $\mathcal{S}_0$. [14]

The n-S problem for $\mathcal{T}$ w.r.t. the l.t.s., $\mathcal{CS}$, and $\mathcal{S}_0$ is both NP-hard and coNP-hard with input $\mathcal{S}_0$. Furthermore, the n-S problem for $\mathcal{T}$ w.r.t. the l.t.s., $\mathcal{CS}$, and $\mathcal{S}_0$ is in the class Δ_2^p of the polynomial hierarchy (P^{NP}) with input $\mathcal{S}_0$. [14]

The n-L problem for $\mathcal{T}$ w.r.t. the l.t.s., $\mathcal{CS}$ and $\mathcal{S}_0$ is in the class Δ_2^p of the polynomial hierarchy with input the $\mathcal{S}_0$.

The proof of Theorem 2 can be found in [15]. The upper bound results rely on Proposition 1, which provides bounds on the length of traces with a bounded number of instances of $Tick$ rules for PTSes. The complexity lower bounds are obtained by encoding the 3-SAT problem. For details see [15, Section 6.2].

6 Related and Future Work

In this paper, we study a subclass of timed MSR systems called progressing timed systems introduced in [14], which is defined by imposing syntactic restrictions on MSR rules.

We discuss two verification problems, namely realizability and survivability, introduced in [14], and also consider two new properties, reliability and recoverability, defined over infinite traces. We show that these problems are PSPACE-complete for progressing timed systems, and when we additionally impose a bound on time, the realizability becomes NP-complete and the survivability and reliability are in Δ_2^p of the polynomial hierarchy, and that the survivability is both NP-hard and coNP-hard. The lower bound for the n-time reliability is left for future work.

These problems involve quantitative temporal properties of timed systems and explicit time constraints, and to the best of our knowledge have not been studied in the rewriting literature. We review some of the formalisms for specifying quantitative temporal properties of timed systems such as timed automata, temporal logic, and rewriting.

Our progressing condition is related to the finite-variability assumption used in the temporal logic and timed automata literature [11, 22, 23, 2, 3], requiring that in any

bounded interval of time, there can be only finitely many observable events or state changes. Similarly, progressing systems have the property that only a finite number of instantaneous rules can be applied in any bounded interval of time (Proposition 1). Such a property seems to be necessary for the decidability of many temporal verification problems.

The work [1, 6] classifies (sets of) traces as safety, liveness, or properties that can be reduced to subproblems of safety and liveness. Following this terminology, properties that relate to our verification problems over infinite traces contain alternation of quantifiers, *i.e.*, involve both elements of safety and elements of liveness. We do not see how this can be expressed precisely in terms of [1, 6]. We leave this investigation to future work.

As discussed in detail in the Related Work section of our previous work [19], there are some important differences between our timed MSR model and timed automata [2, 3], both in terms of expressive power and decidability proofs. For example, a description of a timed MSR system uses first order formulas with variables, whereas timed automata can only refer to transitions on ground states. That is, timed MSR is essentially a first-order language, while timed automata are propositional. Replacing a first order description of timed MSR by all its instantiations, would lead to an exponential explosion. Furthermore, in contrast with the timed automata paradigm, in timed MSR we can naturally manipulate facts both in the past, in the future, as well as in the present.

The temporal logic literature has proposed many languages for the specification and verification of timed systems. Many temporal logics contain quantitative temporal operators, *e.g.*, [23, 22], including time-constrained temporal operators. Metric Temporal Logic (MTL) [21] involves (bounded or unbounded) timing constraints on temporal operators similar to our time constraints. The growing literature on MTL explores the expressivity and decidability of fragments of such temporal logics [26]. However, the temporal logic literature does not discuss notions similar to *e.g.*, realizability or survivability. In addition to that, an important feature of our model is that the specifications are executable. As we have shown through experiments in [14], it is feasible to analyze fairly large progressing systems using the rewriting logic tool Maude [7].

Real-Time Maude [25] is a tool for simulation and analysis of real-time systems. Rewrite rules are partitioned into instantaneous rules and rules that advance time, with instantaneous rules taking priority. Our the lazy time sampling is inspired by such management of time in traces. Time advance rules in Real-Time Maude may place a bound on the amount of time to advance, but do not determine a specific amount, thus, allowing the system to be continuously observed. Time sampling strategies are used to implement search and model-checking analysis. Ölveczky and Messeguer [24] investigate conditions under which the maximal time sampling strategy used in Real-Time Maude is complete. One of the required conditions is tick-stabilizing, which is similar to progressing and finite variability assumption in that one assumes a bound on the number of actions that can be applied in a finite time.

Cardenas *et al.* [4] discuss possible verification problems of cyber-physical systems in the presence of malicious intruders. They discuss surviving attacks, such as denial of service attacks on control mechanisms of devices. We believe that our progressing timed systems can be used to define meaningful intruder models and formalize the corresponding survivability notions. This may lead to automated analysis of such systems

similar to the successful use of the Dolev-Yao intruder model [8] for protocol security verification. Given the results of this paper, the decidability of any security problem would most likely involve a progressing timed intruder model. We intend to investigate the security aspects of this work in the future. For example, the introduction of timed intruder models [12], and resource-bounded intruder models [28] may enable verification of whether intruders can cause PTSes to reach hazardous situations, *e.g.*, harm to people or crashes.

We believe that some of our properties, in particular survivability, can be interpreted using game theory. We find that our model has some features that are better suited for applications relating to TSDSes, in particular explicit time, quantitative time conditions and nonces. It would be interesting to investigate connections and differences between our rewriting approach to these problems and the game theory approach.

Finally, we have already done some preliminary research into ways to extend this work to dense time domains. We expect our results to hold for dense time domains as well, given our previous work [17, 29, 28]. There, instead of the $Tick$ rule (Eq. 1), we assume a $Tick$ rule of the form $Time@T \longrightarrow Time@(T + \varepsilon)$, where ε can be instantiated by any positive real number. The assumption of dense time is a challenge that considerably increases the machinery needed to prove our results, but we are confident of finding ways to combine the results of [17] with those presented in this paper. Similarly, for our future work, we intend to investigate extensions of our models with probabilities.

Acknowledgments: We thank the anonymous reviewers for their valuable comments and careful remarks, which have significantly improved the presentation of the paper. Ban Kirigin is supported in part by the Croatian Science Foundation under the project UIP-05-2017-9219. The work of Max Kanovich was partially supported by EPSRC Programme Grant EP/R006865/1: "Interface Reasoning for Interacting Systems (IRIS)." Nigam is partially supported by NRL grant N0017317-1-G002, and CNPq grant 303909/2018-8. Scedrov was partially supported by the U. S. Office of Naval Research under award numbers N00014-20-1-2635 and N00014-18-1-2618. Talcott was partially supported by the U. S. Office of Naval Research under award numbers N00014-15-1-2202 and N00014-20-1-2644, and NRL grant N0017317-1-G002.

References

1. B. Alpern and F. B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.
2. R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In *Real-Time: Theory in Practice, REX Workshop*, pages 74–106, 1991.
3. R. Alur and P. Madhusudan. Decision problems for timed automata: A survey. In *SFM*, pages 1–24, 2004.
4. A. A. Cárdenas, S. Amin, and S. Sastry. Secure control: Towards survivable cyber-physical systems. In *ICDCS*, pages 495–500, 2008.
5. I. Cervesato, N. A. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *CSFW*, pages 55–69, 1999.
6. M. R. Clarkson and F. B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.

7. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude: A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer, 2007.
8. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
9. N. A. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.
10. H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.
11. M. Faella, A. Legay, and M. Stoelinga. Model checking quantitative linear time logic. *Electr. Notes Theor. Comput. Sci.*, 220(3):61–77, 2008.
12. M. Kanovich, T. Ban Kirigin, V. Nigam, and A. Scedrov. Bounded memory dolev–yao adversaries in collaborative systems. *Information and Computation*, 238:233–261, 2014.
13. M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, and C. Talcott. Discrete vs. dense times in the analysis of cyber-physical security protocols. In *Principles of Security and Trust - 4th International Conference, POST*, pages 259–279, 2015.
14. M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, and C. Talcott. Timed multiset rewriting and the verification of time-sensitive distributed systems. In *14th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2016.
15. M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, and C. Talcott. On the complexity of verification of time-sensitive distributed systems: Technical report. Available at http://arxiv.org/abs/2105.03531, 2021.
16. M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, and C. L. Talcott. Timed multiset rewriting and the verification of time-sensitive distributed systems: Technical report. Available at http://arxiv.org/abs/1606.07886, 2016.
17. M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, and C. L. Talcott. Time, computational complexity, and probability in the analysis of distance-bounding protocols. *Journal of Computer Security*, 25(6):585–630, 2017.
18. M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, C. L. Talcott, and R. Perovic. A rewriting framework for activities subject to regulations. In *RTA*, pages 305–322, 2012.
19. M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, C. L. Talcott, and R. Perovic. A rewriting framework and logic for activities subject to regulations. *Mathematical Structures in Computer Science*, 27(3):332–375, 2017.
20. M. Kanovich, P. Rowe, and A. Scedrov. Collaborative planning with confidentiality. *J. Autom. Reasoning*, 46(3-4):389–421, 2011.
21. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299, 1990.
22. F. Laroussinie, P. Schnoebelen, and M. Turuani. On the expressivity and complexity of quantitative branching-time temporal logics. *Theor. Comput. Sci.*, 297(1-3):297–315, 2003.
23. C. Lutz, D. Walther, and F. Wolter. Quantitative temporal logics: PSPACE and below. In *TIME*, pages 138–146, 2005.
24. P. C. Ölveczky and J. Meseguer. Abstraction and completeness for real-time maude. *Electr. Notes Theor. Comput. Sci.*, 176(4):5–27, 2007.
25. P. C. Ölveczky and J. Meseguer. The real-time maude tool. In *TACAS 2008*, pages 332–336, 2008.
26. J. Ouaknine and J. Worrell. Safety metric temporal logic is fully decidable. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 411–425. Springer, 2006.
27. C. H. Papadimitriou. *Computational complexity*. Academic Internet Publ., 2007.
28. A. Urquiza, M. A. Alturki, T. Ban Kirigin, M. Kanovich, V. Nigam, A. Scedrov, and C. Talcott. Resource and timing aspects of security protocols. *Journal of Computer Security*, 29(3):299–340, 2021.

29. A. Urquiza, M. A. AlTurki, M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, and C. Talcott. Resource-bounded intruders in denial of service attacks. In *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, pages 382–396. IEEE, 2019.