Journal of Computer Security 1 (2014) 1–7
IOS Press

Time, Computational Complexity, and Probability in the Analysis of Distance-Bounding Protocols

Max Kanovich [a,b], Tajana Ban Kirigin [c], Vivek Nigam [d,g], Andre Scedrov [e,b] and Carolyn Talcott [f]

[a] *Department of Computer Science (UCL-CS), University College London, London, UK*
E-mail: m.kanovich@ucl.ac.uk
[b] *Faculty of Computer Science, National Research University Higher School of Economics, Moscow, Russian Federation*
[c] *Department of Mathematics, University of Rijeka, Rijeka, Croatia*
E-mail:bank@math.uniri.hr
[d] *Computer Science Department, Federal University of Paraíba, João Pessoa, Brazil*
E-mail: vivek@ci.ufpb.br
[e] *Department of Mathematics, University of Pennsylvania, Philadelphia, PA, USA*
E-mail: scedrov@math.upenn.edu
[f] *Computer Science Laboratory, SRI International, Menlo Park, CA, USA*
E-mail: clt@csl.sri.com
[g] *fortiss, Munich, Germany*

Abstract. Many security protocols rely on the assumptions on the physical properties in which its protocol sessions will be carried out. For instance, Distance Bounding Protocols take into account the round trip time of messages and the transmission velocity to infer an upper bound of the distance between two agents. We classify such security protocols as Cyber-Physical. Time plays a key role in design and analysis of many of these protocols. This paper investigates the foundational differences and the impacts on the analysis when using models with discrete time and models with dense time. We show that there are attacks that can be found by models using dense time, but not when using discrete time. We illustrate this with an attack that can be carried out on most Distance Bounding Protocols. In this attack, one exploits the execution delay of instructions during one clock cycle to convince a verifier that he is in a location different from his actual position. We additionally present a probabilistic analysis of this novel attack. As a formal model for representing and analyzing Cyber-Physical properties, we propose a Multiset Rewriting model with dense time suitable for specifying cyber-physical security protocols. We introduce Circle-Configurations and show that they can be used to symbolically solve the reachability problem for our model, and show that for the important class of balanced theories the reachability problem is PSPACE-complete. We also show how our model can be implemented using the computational rewriting tool Maude, the machinery that automatically searches for such attacks.

Keywords: Multiset Rewrite Systems, Cyber-Physical Security Protocols, Protocol Security, Computational Complexity, Maude

1. Introduction

With the development of pervasive cyber-physical systems and consequent security issues, it is often necessary to specify protocols that not only make use of cryptographic keys and nonces,[1] but also take into account the physical properties of the environment where its protocol sessions are carried out. We call such protocols *Cyber-Physical Security Protocols*. For instance, Distance Bounding Protocols [5] is a class of cyber-physical security protocols which infers an upper bound on the distance between two agents from the round trip time of messages. In a distance bounding protocol session, the verifier (V) and the prover (P) exchange messages:

$$\begin{array}{l} V \longrightarrow P : m \\ P \longrightarrow V : m' \end{array} \tag{1}$$

where m is a challenge and m' is a response message (constructed using m's components). To infer the distance to the prover, the verifier remembers the time, t_0, when the message m was sent, and the time, t_1, when the message m' returns. From the difference $t_1 - t_0$ and the assumptions on the speed of the transmission medium, v, the verifier can compute an upper bound on the distance to the prover, namely $(t_1 - t_0) \times v$. Typically, the verifier grants the access to the prover if the inferred upper bound on the distance does not exceed some pre-established fixed *time response bound*, R, given by the protocol specification.

Distance bounding protocol sessions are used in a number of cyber-physical security protocols to infer an upper-bound on the distance of participants. Examples include Secure Neighbor Discovery, Secure Localization Protocols [43,6,41], and Secure Time Synchronization Protocols [42,18]. The common feature in most cyber-physical security protocols is that they mention cryptographic keys, nonces and time. (For more examples, see [3,32,13] and references therein.)

A major problem of using the traditional protocol notation for the description of distance bounding protocols, as in (1), is that many assumptions about time, such as the time requirements for the fulfillment of a protocol session, are not formally specified. It is only informally described that the verifier remembers the time t_0 and t_1 and which exact moments these correspond to. Moreover, from the above description, it is not clear which assumptions about the network are used, such as the transmission medium used by the participants. Furthermore, it is not formally specified which properties does the above protocol ensure, and in which conditions and against which intruders.

It is easy to check that the above protocol is not safe against the standard Dolev-Yao intruder [14] who is capable of intercepting and sending messages anywhere at anytime. The Dolev-Yao intruder can easily convince V that P is closer than he actually is. The intruder first intercepts the message m and with zero transmission time sends it to P. Then he intercepts the message m' and instantaneously sends it to V, reducing the round-trip-time $(t_1 - t_0)$. Thus, V will believe that P is much closer than he actually is. Such an attack does not occur in practice as messages take time to travel from one point to another. Indeed, the standard Dolev-Yao intruder model is not a suitable model for the analysis of cyber-physical protocols. Since such an intruder is able to intercept and send messages anywhere at anytime, he appears faster than the speed of light. In fact, a major difference between cyber-physical protocols and traditional security protocols is that there is not necessarily a network in the traditional sense, as any transmission medium used is the part of the network.

[1]In protocol security literature [7,15] fresh values are usually called *nonces*.

Existing works have proposed and used models with time for the analysis of distance bounding protocols where the attacker is constrained by some physical properties of the system. Some models have considered dense time [3], while others have used discrete time [4]. However, although these models have included time, the foundational differences between these models and the impacts to analysis has not been investigated in more detail. For example, these models have not investigated the fact that provers, verifiers, and attackers may have different clock rates, *i.e.*, processing speeds, which can affect security. Indeed, already in the original paper on Distance Bounding Protocols, Brands and Chaum [5] suspected that a verifier may be subject to attack as it uses discrete clock ticking, *i.e.*, a processor, and thus measures time in discrete units, while the environment and the powerful attacker is not limited by a particular clock. However, until now, no careful analysis of such attacks has been carried out. This paper addresses this gap.

A key observation of this paper is that models with dense time abstract the fact that attacker clocks may tick at any rate. The attacker can mask his location by exploiting the fact that a message may be sent at any point between two clock ticks of the verifier's clock, while the verifier believes that it was sent at a particular time. Depending on the speed of the verifier, *i.e.*, its clock rate, the attacker can *in principle* convince the verifier that he is very close to the verifier (less than a meter) even though he is very far away (many meters away). We call this attack *attack in-between-ticks*.

Interestingly, from a foundational point of view, in our formalization there is no complexity increase when using a model with dense time when compared to a model with discrete time. In our previous work [24,25], we proposed a rewriting framework which assumed discrete time. We showed that the reachability problem is PSPACE-complete. Here we show how to formalize systems with dense time, and that, if we extend the model with dense time, the reachability problem is still PSPACE-complete. For this result we introduce a novel machinery called Circle-Configurations. Moreover, we show that it is possible to automate verification of whether a protocol written in our model is subject to an instance of an attack in-between-ticks.

However, our symbolic analysis only provides an yes or no answer, that is, it is only possible to infer whether a system is or not subject to an attack in-between-ticks. For a more accurate analysis, we construct a probabilistic model for analysing Distance Bounding Protocols. This model provides precise measure of how probable a system is subject to an attack. A difference, however, to the proposed symbolic model is that it seems less likely to carry out analysis in an automated fashion. We leave this investigation to future work.

The paper is organised as follows. Section 2 contains the novel attack in-between-ticks. In Section 3 we introduce a formal model based on Multiset Rewriting (MSR) which includes dense time. We also show how to specify distance bounding protocols in this language. In Section 4 we provide a probabilistic analysis related to distance bounding protocols and the attack in-between-ticks. In Section 5 we show how our model can be implemented using the computational rewriting tool Maude. In particular, Maude is able to automatically find the novel attack in-between-ticks. Section 6 introduces a novel machinery, called circle-configurations, that allows one to symbolically represent configurations that mention dense time. In Section 7 we prove that the reachability problem for timed bounded memory protocols [22] is PSPACE-complete. Finally, in Section 8, we comment on related and future work.

This paper considerably extends the conference paper [26]. Besides adding the proofs of our complexity results, Section 4 and 5 contain new material.

2. Motivating Examples

In this section we point to some subtleties of cyber-physical protocol analysis and verification. We present examples of protocols that serve to illustrate the differences between models with discrete and dense time.

One such an example is a timed version of the classical *Needham-Schroeder* protocol [33] which we already presented in our conference paper [26]. We have shown that the timed Needham-Schroeder protocol may be safe in the discrete time model, but an attack (similar to Lowe attack [30]) is possible when using models with dense time. This phenomena shows that some attacks on cyber-physical security protocols may only be found when using models with dense time.

2.1. Time-Bounding Needham-Schroeder Protocol

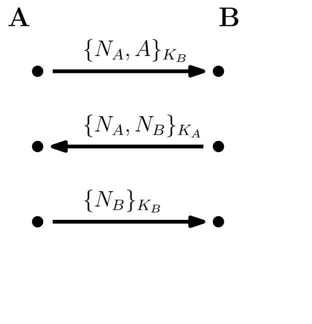
We first show some subtleties of cyber-physical security protocol analysis by re-examining the original *Needham-Schroeder* public key protocol [33] (NS), presented in Figure 1a. Although this protocol is well known to be insecure [30], we look at it from another dimension, the dimension of time. We check whether Needham and Schroeder were right after all, in the sense that their protocol can be considered secure under some time requirements. In other words, we investigate whether NS can be fixed by means of time.

We timestamp each event in the protocol execution, that is, we explicitly mark the time of sending and receiving messages by a participant. We then propose a timed version of this protocol, called *Time-Bounding Needham-Schroeder Protocol* (Timed-NS), as depicted in Figure 1b. The protocol exchanges the same messages as in the original version, but the last protocol message, *i.e.*, the confirmation message $\{N_B\}_{K_B}$, is sent by A only if the time difference $t_3 - t_0$ is smaller or equal to the given *response bounding time R*.

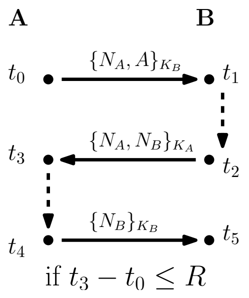
The protocol is considered *secure* in the standard way, that is, if the "accepted" N_A and N_B may never be revealed to anybody else except Alice and Bob. Recall that the well known Lowe attack on NS [30] involves a third party, Mallory who is able to learn Bob's nonce. At the same time Bob believes that he communicated with Alice and that only Alice learned his nonce.

The intriguing result of the analysis of Timed-NS is that one may not find an attack in the discrete time model, but can find one in the dense time model: Figure 2 depicts the Lowe attack scenario in Timed-NS. In particular, the attack requires that events marked with $t_0, \ldots, t_7$ take place and that the round trip time of messages, that is $t_7 - t_0$, does not exceed the given response bounding time R. Assuming that both network delay and processing time are non-zero, in the discrete time model the attack could be modeled only for response bounding time $R \geq 7$, see Figure 2a. In the discrete model, the protocol would seem safe for response bounding time $R < 7$. However, in the dense time model the attack is possible for any positive response bounding time R, see Figure 2b. Indeed, assuming continuous times, the attacker can, in principle, be as fast as needed to satisfy the given response bounding time. Notice additionally, that if R is set to be too low, then it may also turn the protocol unusable as legitimate players may not be able to satisfy the response time requirement. The same is true with Distance Bounding Protocols. If the distance bound is set to be too low, legitimate players may not be able to get access to resources. Therefore, response bounds should be set in such a way that makes the protocol secure and still grant resource to legitimate users.

This simple example already illustrates the challenges of timed models for cyber-physical security protocol analysis and verification. No rescaling of discrete time units removes the presented difference

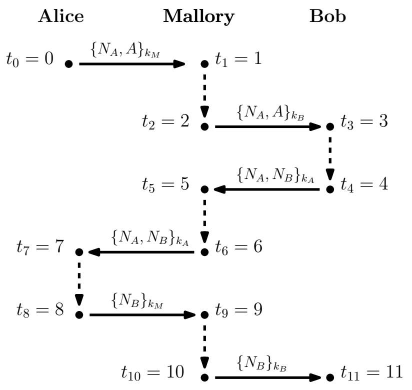


(a) Needham-Schroeder protocol

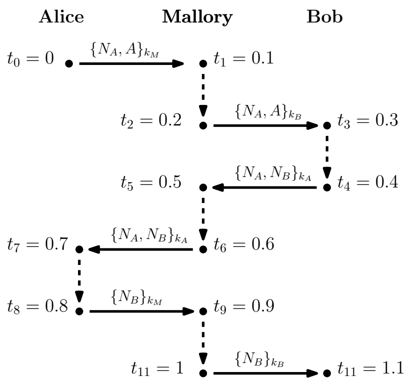


(b) Timed Needham-Schroeder Protocol

Fig. 1. Adding time to Needham-Schroeder Protocol



(a) Discrete Time Model



(b) Dense Time Model

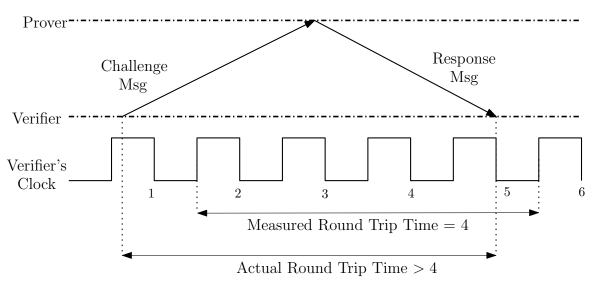
Fig. 2. Timed Version of Lowe Attack

between the models. For any discretization of time, such as seconds or any other infinitesimal time unit, there is a protocol for which there is an attack with continuous time and no attack is possible in the discrete case.

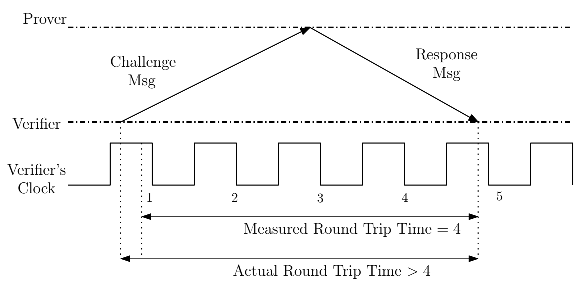
We further illustrate the challenges of timed models for cyber-physical security protocol analysis and verification by a more realistic example of a distance bounding protocol. The novel attack in-between-ticks appears and illustrates that for the analysis of distance bounding protocols it is necessary to consider time assumptions of the players involved.

2.2. *Attack In-Between-Ticks*

Regardless of the design details of a specific distance bounding protocol, we identify a new type of anomaly. We call it *Attack In-Between-Ticks*. This attack is particularly harmful when the verifier and the prover exchange messages using radio-frequency (RF), where the speed of transmission is the speed of light. In this case an error of a 1 nanosecond (*ns*) already results in a distance error of 30*cm*. This



(a) In different ticks (Sequential Execution)



(b) In the same tick (Parallel Execution)

Fig. 3. Attack In-Between-Ticks. Here time response bound $R = 4$ ticks.

is the case with a number of cyber-physical security protocols which try to establish the property that participants are *physically close* to each other, *e.g.*, Secure Neighbor Discovery, Secure Localization Protocols [43,6,41], Secure Time Synchronization Protocols [42,18], Protocols used by RFID payment devices [9,10].

Consider the illustrations in Figure 3. They depict the execution of instructions by the verifier in a distance bounding protocol with the time response bound $R = 4$. The verifier has to execute two instructions: (1) the instruction that sends the signal to the prover and (2) the instruction that measures the time when this message is sent. Figure 3a illustrates the case when the verifier is running a sequential machine (that is, a single processor), which is the typical case as the verifier is usually a not very powerful device, *e.g.*, door opening device. Here we optimistically assume that an instruction can be executed in one cycle. When the first instruction is executed, it means that the signal is sent somewhere when the clock is up, say at time 0.6. In the following clock cycle, the verifier remembers the time when the message is sent. Say that this was already done at time 1.5. If the response message is received at time 5 it triggers an interruption so that the verifier measures the response time in the following cycle, *i.e.*, at time 5.5. Thus the measured round time is $4 = 5.5 - 1.5 = R$ ticks. Therefore, the verifier grants access to the prover although the actual round trip is $5 - 0.6 = 4.4 > R = 4$ ticks. This means that the verifier is granting access to the prover although the prover's distance to the verifier may not satisfy the distance bound and thus this is a security flaw.[2]

Depending on the speed of verifier's processor, the difference of 0.4 tick results in a huge error. Many of these devices use very weak processors.[3] The one proposed in [40], for example, executes at a frequency of at most $24MHz$. This means a tick is equal to $41ns$ (in the best case). Thus, an error of 0.4 tick corresponds to an error of $16ns$ or an error of 2×2.4 *meters* when using RF. In the worst case, the error can be of 1.0 tick when the signal is sent at the beginning of the cycle, *i.e.*, at time 0.5 tick, and the measurements at the end of the corresponding cycles, *i.e.*, at times 2 and 6 ticks. An error of 1.0 tick ($41ns$) corresponds to an error greater than 2×6.15 *meters*.

Consider now the case when the verifier can execute both instructions in the same cycle. Even in this case there might be errors in measurement as illustrated in Figure 3b. It may happen that the signal is

[2]Notice that inverting the order of the instructions, *i.e.*, first collecting the time and then sending the signal, would imply errors of measurement but in the opposite direction turning the system impractical.

[3]High precision verifiers are expensive and normally used only in high security applications.

sent before the measurement is taken thus leading to errors of at most 0.5 ticks (not as great as in the sequential case). (Here we are again optimistically assuming that an instruction can be executed in one cycle.)

2.3. Vulnerability of Cyber-Physical Security Protocols to Attack In-Between-Ticks

The attack in-between-ticks is based on the foundational difference between real-time in nature and time management by discrete time processors, irrespective of their physical-layer implementations and distance bounding protocol design. The attack in-between-ticks appears because of the discrepancy between actual real-time distance and the upper bound on time distance that is calculated by a discrete verifier from the time taken as the time of sending and receiving challenge and response messages, respectively.

Distance bounding protocol sessions are used in a number of cyber-physical security protocols, such as Secure Neighbor Discovery, Secure Localization Protocols [43,6,41], and Secure Time Synchronization Protocols [42,18]. Therefore, all these protocols can be vulnerable to the attack in-between-ticks.

Such a discrepancy has been suspected in Brands and Chaum's original paper introducing distance bounding protocols [5]. However, no study on the vulnerabilities of this discrepancy has been investigated until now. The attack in-between-ticks represents a *new category of attacks that needs to be considered in the analysis of cyber-physical protocols.*

From the well-known types of attacks, such as Distance Fraud, Mafia Fraud, Terrorist Fraud and Distance Hijacking [13], the distance fraud, or the corresponding classification of Lone Distance Fraud from [13] is the closest to the attack in-between-ticks. It also leads the verifier to believe that a prover is closer than he actually is, and it does not involve additional participants, only a single prover and a verifier. However, in (lone) distance fraud, the prover is assumed to be dishonest, trying to change his distance and appear closer to the verifier than he actually is. This can be achieved in some cases by, *e.g.*, prover sending the response message(s) too soon, before receiving the challenge message(s), and can be fixed by changes in the protocol design, so that the response messages are dependant of challenge messages sent by the verifier. On the other hand, in the attack in-between-ticks, even an honest prover can result closer than he actually is, not by his intent, but because of the discrepancy between real time and discrete processor clocks.

In order to obtain high-resolution timing information about the arrival of individual data bits, distance bounding protocol, as well as secure-positioning protocols, are tightly integrated into the physical layer of the communication protocol, providing sub-microsecond timing information. In this way these protocols rely directly on the laws of physics, *i.e.*, on the assumption that the communication is bounded by the speed of light. Nevertheless, as we have shown above, substantial errors in measurement, up to several meters, may appear. Verifier may allow the access to a prover that is considerably outside the distance bound specified by the protocol. This would equally appear in any distance bounding protocol, *e.g.*, in Brands and Chaum protocol [5] or Hancke-Kuhn protocol [19].

Formal frameworks for reasoning about distance bounding protocols that are based on discrete time models, such as [4], clearly cannot capture the attack in-between-ticks. This is also the case with the real-time formalisms that do not take into account the way verifiers with discrete clocks operate. None of the real-time formalisms that we are aware of formalizes this treatment of time that is typical of processors.

Furthermore, we observe that these security flaws may happen *in principle*. In practice, distance bounding protocols carry out a large number of challenge and response rounds. This is generally believed to mitigate the chances of attacks occurring. In the future, we intend to investigate the challenge-response approach in providing security of distance bounding protocols. In particular, we are planning to analyze whether the effects of the attack in-between-ticks can be reduced by repeated challenge-response rounds of protocols.

Finally, we also point out that these attacks have been inspired by similar issues in the analysis of digital circuits [2].

3. A Multiset Rewriting Framework with Dense Time

For our multiset rewriting framework we assume a finite first-order typed alphabet, Σ, with variables, constants, function and predicate symbols. Terms and facts are constructed as usual (see [16]) by applying symbols with correct type (or sort). For instance, 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 grounded if it does not contain any variables.

In order to specify systems that explicitly mention time, we use *timestamped facts* of the form $F@T$, where F is a fact and T is its timestamp. In our previous work [25], timestamps were only allowed to be natural numbers. Here, on the other hand, in order to express dense time, timestamps are allowed to be non-negative real numbers.

We assume that there is a special predicate symbol $Time$ with arity zero, which will be used to represent the global time. A *configuration* is a multiset of ground timestamped facts,

$$\{\, Time@t,\ F_1@t_1, \ldots,\ F_n@t_n \,\},$$

with a single occurrence of a $Time$ fact. Configurations are to be interpreted as states of the system. For example, the following configuration

$$\{\, Time@7.5,\ Deadline@10.3,\ Task\,(1,\text{done})@5.3,\ Task\,(2,\text{pending})@2.13 \,\} \tag{2}$$

specifies that the current global time is 7.5, the Task 1 was performed at time 5.3, Task 2 was issued at time 2.13 and is still pending, and that the deadline to perform all tasks is 10.3.

We may sometimes denote the timestamp of a fact F in a given configuration as T_F.

3.1. Actions and Constraints

Actions are multiset rewrite rules and are either the time advancement action or instantaneous actions. The action representing the advancement of time, called *Tick Action*, is the following:

$$Time@T \longrightarrow Time@(T + \varepsilon) \tag{3}$$

Here ε can be instantiated by any positive real number specifying that the global time of a configuration can advance by any positive number. For example, if we apply this action with $\varepsilon = 0.6$ to the configuration (2) we obtain the configuration

$$\{\, Time@8.1,\ Deadline@10.3,\ Task\,(1,\text{done})@5.3,\ Task\,(2,\text{pending})@2.13 \,\} \tag{4}$$

where the global time advanced from 7.5 to 8.1.

Clearly such an action is a source of unboundedness as time can always advance by any positive real number. In particular we will need to deal with issues such as Zeno Paradoxes [1] when considering how time should advance.

The remaining actions are the Instantaneous Actions, which do not affect the global time, but may rewrite the remaining facts.

Definition 3.1. Instantaneous Actions *are actions of the form:*

$$\begin{array}{l} Time@T,\ W_1@T_1,\ldots,\ W_k@T_k,\ F_1@T_1',\ldots,\ F_n@T_n' \ \mid\ C\ \longrightarrow \\ \quad \exists\,\vec{X}.\ [\ Time@T,\ W_1@T_1,\ldots,\ W_k@T_k,\ Q_1@(T+D_1),\ldots,\ Q_m@(T+D_m)\ ] \end{array} \tag{5}$$

where $D_1,\ldots,D_m$ are natural numbers, existentially quantified variables $\vec{X}$ denote fresh values that are created by the rule, and C is the guard of the action which is a set of constraints involving the time variables appearing in the pre-condition, i.e., *the variables $T, T_1,\ldots,T_k,T_1',\ldots,T_n'$.*
Time Constraints *are expressions of the form:*

$$T' > T'' \pm D \quad \textit{and} \quad T' = T'' \pm D \tag{6}$$

where T' and T'' are time variables, and D is a natural number. [4]

We use $T' \geq T'' \pm D$ to denote the disjunction of $T' > T'' \pm D$ and $T' = T'' \pm D$.
An instantaneous action can only be applied if all the constraints in its guard are satisfied.
We say that facts $F_i@T_i'$ are *consumed* and that the facts $Q_i@(T+D_i)$ are *created* by the rule (5).

Notice that we allow only natural numbers for constants Ds and D_is that appear in time constraints and timestamps of created facts. We impose such conditions because of the relevant computational complexity issues. However, this is not as restrictive w.r.t. expressivness of the model as one might think. We will address this issue in more detail later on in Section 7 when we investigate the complexity of related computational problems.

Also, notice that the global time does not change when applying an instantaneous action. Moreover, the timestamps of the facts that are created by the action, namely the facts $Q_1,\ldots,Q_m$, are of the form $T+D_i$, where D_i is a natural number and T is the global time. That is, their timestamps are in the present or the future. For example, the following is an instantaneous action

$$\begin{array}{c} Time@T,\ Task\,(1,\text{done})@T_1,\ Deadline@T_2,\ Task\,(2,\text{pending})@T_3\ \mid\ \{\ T_2 \geq T+2\ \} \\ \longrightarrow\ Time@T,\ Task\,(1,\text{done})@T_1,\ Deadline@T_2,\ Task\,(2,\text{done})@(T+1) \end{array}$$

which specifies that one should complete Task 2, if Task 1 is already completed, and moreover, if the Deadline is at least 2 units ahead of the current time. If these conditions are satisfied, then the Task 2 will be completed in one time unit. Applying this action to the configuration (4) yields

$$\{\ Time@8.1,\ Deadline@10.3,\ Task\,(1,\text{done})@5.3,\ Task\,(2,\text{done})@9.1\ \}$$

[4]Here, and in the rest of the paper, the symbol $\pm$ stands for either $+$ or $-$, that is, constraints may involve addition or subtraction.

where Task 2 will be completed by the time 9.1.

Finally, the variables $\vec{X}$ that are existentially quantified in (5) are to be replaced by fresh values, also called *nonces* in protocol security literature [7,15]. For example, the following action specifies the creation of a new task with a fresh identifier Id, which should be completed by time $T + D$:

$$Time@T \longrightarrow \exists\, Id.\ [\ Time@T,\ Task\,(Id, \text{pending})@(T + D)\ ]$$

Whenever this action is applied to a configuration, the variable Id is instantiated by a fresh value. In this way we are able to specify that the identifier assigned to the new task is different to the identifiers of all other existing tasks. In the same way it is possible to specify the use of nonces in Protocol Security [7,15].

Formally, a rule $W \mid C \longrightarrow\ \exists\, \vec{X}.\ W'$ can be applied to a configuration $\mathcal{S}$ if there is a ground substitution σ, where the variables in $\vec{X}$ are fresh, such that $W\sigma \subseteq \mathcal{S}$ and $C\sigma$ is true. The resulting configuration is $(\mathcal{S} \setminus W) \cup W'\sigma$.

Notice that by 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 be applicable to the same configuration $\mathcal{S}$, which may lead to different resulting configurations $\mathcal{S}'$.

More precisely, an instance of an action is obtained by substituting all variables appearing in the pre- and post-condition of the action with constants. That applies to variables appearing in terms inside facts, variables representing fresh values, as well as time variables used in specifying timestamps of facts. For example, consider the following action

$$\begin{array}{l} Time@T,\ Finaltask\,(x, \text{done})@T,\ Deadline@T_1\ \mid\ \{\ T_1 \geq T + 3\ \} \\ \quad \longrightarrow \exists\, n.\ [\ Time@T,\ File\,(n, x, \text{pending})@(T + 2),\ Deadline@T_1\ ] \end{array} \tag{7}$$

specifying that a case record for a completed process should be filed under a unique record label within 2 time units. An instance of above action is obtained by substituting constants for variables x, T, T_1 and n. For example,

$$\begin{array}{c} Time@3.2,\ Finaltask\,(\text{C}, \text{done})@3.2,\ Deadline@10\ \mid\ \{\ 10 \geq 3.2 + 3\ \} \\ \longrightarrow\ Time@3.2,\ File\,(\text{N}, \text{C}, \text{pending})@5.2,\ Deadline@10 \end{array}$$

is an instance of action (7). Recall that an instance of an action can only be applied to a configuration containing all the facts from its precondition if all the corresponding time constraints, *i.e.*, all the time constraints in its guard, are satisfied. For example, since $15.3 \geq 8.1 + 3$, action (7) is applicable to configuration

$$\{\ Time@8.1,\ Finaltask\,(\text{C}_0, \text{done})@8.1,\ Deadline@15.3\ \}\,,$$

resulting in configuration

$$\{\ Time@8.1,\ File\,(\text{N}_1, \text{C}_0, \text{pending})@10.1,\ Deadline@15.3\ \}\,,$$

but it is not applicable to the following configuration

$$\{\ Time@13.1,\ Finaltask\,(\text{C}_0, \text{done})@13.1,\ Deadline@15.3\ \}\,.$$

3.2. Initial, Goal Configurations and The Reachability Problem

We write $\mathcal{S} \longrightarrow_r \mathcal{S}_1$ for the one-step relation where configuration $\mathcal{S}$ is rewritten to $\mathcal{S}_1$ using an instance of action r. For a set of actions $\mathcal{R}$, we define $\mathcal{S} \longrightarrow^*_{\mathcal{R}} \mathcal{S}_1$ as the transitive reflexive closure of the one-step relation on all actions in $\mathcal{R}$. We elide the subscript $\mathcal{R}$, when it is clear from the context.

Definition 3.2. *A* goal $\mathcal{S}_G$ *is a pair of a multiset of facts and a set of constraints, written*

$$\{\, F_1@T_1, \ldots,\ F_n@T_n \,\} \mid C$$

where $T_1, \ldots, T_n$ *are time variables,* $F_1, \ldots, F_n$ *are facts and* C *is a set of constraints involving only* $T_1, \ldots, T_n$*. We call a configuration* $\mathcal{S}_1$ *a* goal configuration w.r.t. goal $\mathcal{S}_G$ *if there is a grounding substitution* σ *replacing term variables by ground terms and time variables by real numbers such that* $\mathcal{S}_G\sigma \subseteq \mathcal{S}_1$ *and all the constraints in* $C\sigma$ *are satisfied.*

For simplicity, since goal will usually be clear from the context, we will use terminology *goal configuration* eliding the goal w.r.t. which it is defined.

The reachability problem is then defined for a given initial configuration, a goal and a set of actions.

Definition 3.3. *Given an initial configuration* $\mathcal{S}_I$*, a goal* $\mathcal{S}_G$ *and a set of actions* $\mathcal{R}$*, the* reachability problem $\mathcal{T}$ *is the problem of establishing whether there is a goal configuration* $\mathcal{S}_1$*, such that* $\mathcal{S}_I \longrightarrow^*_{\mathcal{R}} \mathcal{S}_1$*. Such a sequence of actions leading from an initial to a goal configuration is called a* plan.

We assume that goals are invariant to renaming of fresh values, that is, a goal configuration $\mathcal{S}_G$ is equivalent to the goal configuration $\mathcal{S}'_G$ if they only differ in the nonce names (see [20] for more discussion on this).

Although the reachability problem is stated as a decision problem, we follow [27] and are able to prove more than just existence of a plan. Namely, by using the notion of "scheduling" a plan we are also able to generate a plan when there is a solution. This is useful for the complexity of the plan generation, since the number of actions in the plan may be very large (see [20] for more details on this).

An algorithm is said to *schedule a plan* if it finds a plan if one exists, and on input i, if the plan contains at least i actions, then it outputs the i^{th} action of the plan, otherwise it outputs *no*.

Remark 3.4. *Notice that the feature of time constraints being attached to the rules increases expressivitiy of the model. Constraints may or may not be attached to a rule or to a configuration. With no constraints attached to rules and configurations, we deal with the reachability problem that does not make use of the dimension of time to its potential. In other words, adding constraints to rules and configurations is not a restriction of the model. Quite the contrary, using constraints, we are able to express time properties both for application of rules, and states of the system. In that way we are able to formalize time-sensitive actions and system configurations, and hence we can consider problems that involve explicit time requirements.*

For the complexity results of the reachability problem we will consider actions that are *balanced*, *i.e.*, actions that have the same number of facts in the pre-condition as in the post-condition. Balanced systems that contain only balanced actions have the special property that all configurations in their plans have the same number of facts, given by the number of facts in the initial configuration. This is because when applying a balanced action consumed facts from an enabling configuration are replaced with the same number of created facts in the resulting configuration.

Indeed, balanced systems can be conceived as systems with bounded memory [20]. More precisely, if we additionally impose a bound on the number of symbols that can be contained in a fact, such balanced systems involve configurations with a fixed number of facts of a bounded storage capacity, *i.e.*, they can only store a bounded number of symbols at a time.

Remark 3.5. *Notice that any un-balanced rule can be made balanced by adding additional facts where needed, either in the pre-condition or in the post-conditions of the rule. For that purpose we use so called* empty facts*, $E@T$, that simply denote available memory slots. Notice that in such a way we do not get the equivalent system w.r.t. the reachability problem, as some rules may not be applied in the obtained balanced system unless there is a sufficient number of empty facts in the configuration, even in cases when the corresponding rule is applicable in the original system. For example, the following unbalanced rule*

$$Time@T,\ F_1@T' \ \longrightarrow\ Time@T,\ F_1@T',\ F_1@T,\ F_2@T$$

is applicable to configuration $Time@0, F_1@0, E@0$*, but its corresponding balanced rule*

$$Time@T,\ F_1@T',\ E@T,\ E@T \ \longrightarrow\ Time@T,\ F_1@T',\ F_1@T,\ F_2@T$$

is not. Hence, the obtained balanced system may not have a solution to the given reachability problem, although the original system does. However, there is no a priori bound on the number of facts in the initial configuration, i.e.*, we could add any number of empty facts to the given initial configuration.*

For our complexity results (Section 7) we necessarily consider only systems with balanced actions, since it has been shown in [28] that the reachability problem is undecidable if actions are allowed to be un-balanced. Multiset rewriting models considered in [28] were untimed, and the undecidability of the reachability problem for those models implies undecidability of the reachability problem for timed models considered in this paper.

Additionally, we will impose a bound on the size of facts, that is, a bound on a total number of symbols contained in a fact. Namely, the reachability problem is undecidable even for (un-timed) balanced systems when the size of facts is unbounded [7,15].

Furthermore, in [25] we show that by relaxing any of the main conditions on instantaneous rules, same as conditions given in Equation (5), leads to the undecidability of the reachability problem for multiset rewriting models with discrete time, and thus lead as well to the undecidability of the reachability problems considered in this paper. For example, we get undecidability for systems with time constraints that involve three or more time variables.

3.3. Equivalence Between Configurations

Extending the model to include dense time leads to additional challenges with respect to the complexity of the corresponding problems such as the reachability problem we address in this paper. Our solution proposed in [25] for the model with discrete time is not suitable for the dense time case. In particular it does not address the unboundedness caused by the Tick action which allows time to advance for any positive real number. In order to tackle this source of unboundedness, we define an equivalence relation among configurations defined below.

Many formal definitions and results in this paper mention D_{max}, an upper bound on the numeric values of a reachability problem. This value is computed from the given problem: we set D_{max} to be a natural number such that $D_{max} > n + 1$ for any number n (both real or natural) appearing in the timestamps of the initial configuration, or the Ds and D_is in constraints or actions of the reachability problem.

The following definition establishes the equivalence of configurations.

Definition 3.6. *Given a reachability problem $\mathcal{T}$, let D_{max} be an upper bound on the numeric values appearing in $\mathcal{T}$. Let*

$$\mathcal{S} = \{\, Q_1@t_1,\ Q_2@t_2, \ldots,\ Q_n@t_n \,\} \qquad \textit{and} \qquad \widetilde{\mathcal{S}} = \{\, \widetilde{Q}_1@\widetilde{t_1},\ \widetilde{Q}_2@\widetilde{t_2}, \ldots,\ \widetilde{Q}_n@\widetilde{t_n} \,\} \tag{8}$$

be two configurations written in canonical way where the two sequences of timestamps $t_1, \ldots, t_n$ and $\widetilde{t_1}, \ldots, \widetilde{t_n}$ are non-decreasing. (For the case of equal timestamps, we sort the facts in alphabetical order, if necessary.) We say that configurations $\mathcal{S}$ and $\widetilde{\mathcal{S}}$ are equivalent configurations *if the following conditions hold:*

(i) there is a bijection σ that maps the set of all nonce names appearing in configuration $\mathcal{S}$ to the set of all nonce names appearing in configuration $\widetilde{\mathcal{S}}$, such that $Q_i\sigma = \widetilde{Q}_i$, for each $i \in \{1, \ldots, n\}$; and

(ii) configurations $\mathcal{S}$ and $\widetilde{\mathcal{S}}$ satisfy the same constraints, that is:

$$\begin{array}{lll} t_i > t_j \pm D & \textit{iff} & \widetilde{t_i} > \widetilde{t_j} \pm D \quad \textit{and} \\ t_i = t_j \pm D & \textit{iff} & \widetilde{t_i} = \widetilde{t_j} \pm D, \end{array}$$

for all $1 \le i \le n$, $1 \le j \le n$ and $D \le D_{max}$.

When $\mathcal{S}$ and $\widetilde{\mathcal{S}}$ are equivalent we write $\mathcal{S} \sim_{D_{max}} \widetilde{\mathcal{S}}$, or simply $\mathcal{S} \sim \widetilde{\mathcal{S}}$.

Notice that equivalent configurations contain the same (untimed) facts up to renaming of fresh values. Notice as well that by increasing or decreasing all the timestamps of a configuration $\mathcal{S}$ by the same value Δ the obtained configuration $\widetilde{\mathcal{S}}$ satisfies the same constraints as $\mathcal{S}$. That is because time constraints are relative, *i.e.*, involve exactly two time variables, and hence for configurations $\mathcal{S}$ and $\widetilde{\mathcal{S}}$ given in (8) the following holds:

$$\begin{array}{lll} t_i > t_j \pm D & \text{iff} & (\widetilde{t_i} + \Delta) > (\widetilde{t_j} + \Delta) \pm D \quad \text{and} \\ t_i = t_j \pm D & \text{iff} & (\widetilde{t_i} + \Delta) = (\widetilde{t_j} + \Delta) \pm D\,. \end{array}$$

When compared, facts of equivalent configurations satisfy the same order. Although facts follow the same order they need not lay at the exact same points of the real line.

For example, with $D_{max} = 3$, configurations:

$$\begin{array}{l} \mathcal{S}_1 = \{\, Time@0.2,\ F_1(n_1)@1.5,\ F_2(n_1, b)@2.3,\ F_3(b)@2.5 \,\} \quad \text{and} \\ \mathcal{S}_2 = \{\, Time@1.1,\ F_1(n_2)@2.7,\ F_2(n_2, b)@3.2,\ F_3(b)@3.7 \,\} \end{array}$$

are equivalent. However, above configurations are not equivalent to configuration:

$$\mathcal{S}_3 = \{\, Time@0.2,\ F_1(n_1)@1.5,\ F_2(n_1, b)@2.3,\ F_3(b)@2.52 \,\}\,.$$

since both $\mathcal{S}_1$ and $\mathcal{S}_2$ satisfy the constraint $T_3 - T_1 = 1$, where T_1 is the timestamp of the fact F_1 and T_3 is the timestamp of the fact F_3, while $\mathcal{S}_3$ does not.

As shown by the above example, same order of facts by itself is not enough for the equivalence of configurations because all constraints of the type $T_i = T_j \pm D$ need to be satisfied simultaneously by both configurations. Therefore, the facts corresponding to T_i and T_j from constraints $T_i = T_j \pm D$, when placed on the real line, need to lay at the points with the same decimal parts. Moreover, matching of corresponding facts at integer points needs to hold when placing any of the corresponding pairs of facts at point 0 on the real line.

Moreover, because of constraints of the type $T_i > T_j \pm D$, when placing any pair of corresponding facts at 0, all remaining pairs of corresponding facts need to lay either at the same integer point or inbetween same consecutive integers.

In Section 6 we introduce another, more illustrative, representation of the above equivalence relation.

The following proposition captures our intuition that the notion of equivalence defined above is coarse enough to identify applicable actions and thus the reachability problem.

Proposition 3.7. *Let $\mathcal{S}$ and $\mathcal{S}'$ be two equivalent configurations for a given reachability problem $\mathcal{T}$ and the upper bound D_{max}. Let an action r transform $\mathcal{S}$ into $\mathcal{S}_1$. Then there is an instance of the action r such that $\mathcal{S}' \longrightarrow_r \mathcal{S}_1'$ and that configurations $\mathcal{S}_1$ and $\mathcal{S}_1'$ are also equivalent.*

Proof. Let $\mathcal{S}$ and $\mathcal{S}'$ be two equivalent configurations, namely

$$\begin{aligned} \mathcal{S} &= \{\, Time@t,\ Q_1@t_1,\ Q_2@t_2, \dots,\ Q_n@t_n \,\} \qquad \text{and} \\ \mathcal{S}' &= \{\, Time@t',\ Q_1'@t_1',\ Q_2'@t_2', \dots,\ Q_n'@t_n' \,\}\,. \end{aligned}$$

Assume that $\mathcal{S}$ is transformed into $\mathcal{S}_1$ by means of an action r. By definition of equivalence between configurations, Definition 3.6, configuration $\mathcal{S}'$ contains the same (untimed) facts as $\mathcal{S}$ up to nonce renaming, *i.e.*, there is a bijection σ such that $Q_i\sigma = Q_i'$, for all $i = 1, \dots, n$. Also, per Definition 3.6, configuration $\mathcal{S}'$ satisfies all the time constraints corresponding to the action r, if any. Hence the action r is indeed applicable to configuration $\mathcal{S}'$ and will transform $\mathcal{S}'$ into some $\mathcal{S}_1'$, as depicted in the following diagram:

$$\begin{tikzcd} \mathcal{S} \arrow[r, "r"'] \arrow[d, phantom, "\wr"] & \mathcal{S}_1 \\ \mathcal{S}' \arrow[r, "r"'] & \mathcal{S}_1' \end{tikzcd}$$

Since configurations $\mathcal{S}$ and $\mathcal{S}'$ may differ in the actual values of the timestamps attached to their facts, possibly different instances of the action r may be applied to each of the configurations. We consider both cases for the action r, namely time advance action *i.e.*, action of type (3) and instantaneous actions *i.e.*, action of type (5), and we need to show that in both cases $\mathcal{S}_1$ is equivalent to $\mathcal{S}_1'$.

Assume that r is an instantaneous action. Recall that the action r does not change the global time, so t and t' denote the global time both in $\mathcal{S}$ and $\mathcal{S}_1$, and $\mathcal{S}'$ and $\mathcal{S}_1'$, respectively.
Hence, in showing that $\mathcal{S}_1$ is equivalent to $\mathcal{S}_1'$, only constraints involving facts created by the action r are interesting, *i.e.*, facts of the form $F_1@(t + D_1)$ and $F_2'@(t' + D_2)$, respectively. Notice that the fact $P@(t + D)$, crated by an instance of r, appears in $\mathcal{S}_1$ iff the fact $P'@(t' + D)$, created by an instance of r, appears in $\mathcal{S}_1'$, where $P' = P\sigma$ for some nonce renaming bijection σ.

Hence, the relative time difference to the global time is exactly D, both for $P@(t{+}D)$ and for $P'@(t'{+}D)$. This implies that any given time constraint attached to r involving a created fact and the global time is satisfied in $\mathcal{S}_1$ iff it is satisfied in $\mathcal{S}'_1$.
Similarly, time constraints involving two created facts, $P_1@(t+D_1)$, $P_2@(t+D_2)$ and $P'_1@(t'+D_1)$, $P'_2@(t'+D_2)$, respectively, involve the same relative difference $D_1 - D_2$. Hence, they are concurrently satisfied.
Finally, a time constraint c involving a created fact $P_1@(t+D_1)$ and a fact $Q_i@t_i$ that appears in $\mathcal{S}$, and, respectively, facts $P'_1@(t'+D_1)$ and $Q'_i@t'_i$ in $\mathcal{S}'_1$, can be associated to a time constraint involving the global time. Clearly, for all D

$$\begin{array}{l} t_i > t \pm D \ \text{ iff } \ t_i' > t' \pm D \quad \text{and} \\ t_i = t \pm D \ \text{ iff } \ t_i' = t' \pm D \end{array}$$

is equivalent to

$$\begin{array}{l} t_i > (t+D_1) \pm D \ \text{ iff } \ t_i' > (t'+D_1) \pm D \quad \text{and} \\ t_i = (t+D_1) \pm D \ \text{ iff } \ t_i' = (t'+D_1) \pm D \,. \end{array}$$

Above equivalence states that $\mathcal{S}_1$ and $\mathcal{S}'_1$ concurrently satisfy constraint c iff they concurrently satisfy some constraint involving a created fact and the global time. Since the later has already been shown, we can conclude that both $\mathcal{S}$ and $\mathcal{S}'$ satisfy constraint c.

Having considered all the relevant types of constraints, we can conclude that, since $\mathcal{S} \sim \mathcal{S}'$, it follows that $\mathcal{S}_1 \sim \mathcal{S}'_1$.

Now, assume $\mathcal{S}$ is transformed into $\mathcal{S}_1$ by means of a time advancing action r:

$$\mathcal{S}_1 = \{\, Time@(t+\varepsilon),\ Q_1@t_1,\ Q_2@t_2, \ldots,\ Q_n@t_n \,\} \,.$$

Depending on the actual value ε in r we will find the value ε' for the instance of time advancement action that will transform $\mathcal{S}'$ into $\mathcal{S}'_1$ in such a way that equivalence $\mathcal{S}_1 \sim \mathcal{S}'_1$ holds. Recall that time advancement action is applicable to any configuration.

With time advancement only the timestamp denoting the global time is increased while the rest of the configuration remains unchanged. Therefore, only time constraints involving the global time are to be considered for equivalence of $\mathcal{S}_1$ and $\mathcal{S}'_1$.

Firstly, assume that the new global time $t+\varepsilon$ in $\mathcal{S}_1$ is equal to some timestamp $t_j \pm B$, where B is an integer. Then we set

$$\varepsilon' = t'_j \pm B - t' \,.$$

Then the new global time in $\mathcal{S}'_1$ is $t' + \varepsilon' = t'_j \pm B$. Clearly, for any integer D it holds that

$$t_i \lesseqqgtr (t+\varepsilon) \pm D = t_j \pm B \pm D \quad \text{iff} \quad t_i' \lesseqqgtr (t'+\varepsilon) \pm D = t'_j \pm B \pm D$$

since t_i, t'_i and t_j, t'_j are corresponding timestamps from $\mathcal{S}$ and $\mathcal{S}'$ which are equivalent, *i.e.*, satisfy the same constraints.

Next, we consider the remaining case when the decimal part of the new global time $t + \varepsilon$ is different from the decimal part of any t_is in $\mathcal{S}_1$. More precisely, if we arrange the facts in $\mathcal{S}_1$ according only to their decimal parts, then the decimal part of $t + \varepsilon$ either lays directly in between decimal parts of t_i and t_j, or it is greater than the decimal part of any t_i. In order to get the configuration $\mathcal{S}'_1$ that is equivalent to $\mathcal{S}_1$, we need to achieve the same ordering of facts. We therefore set

$$\varepsilon' = int\,(\varepsilon) + \delta,$$

where $\delta \in \langle 0, 1\rangle$ is any number such that

$$\begin{array}{lll} & dec\,(t'_i) < dec\,(t' + \varepsilon') < dec\,(t'_j) & \text{in case when } \; dec\,(t_i) < dec\,(t + \varepsilon) < dec\,(t_j), \;\; \text{for some } \; i, j \\ \text{or} & dec\,(t'_i) < dec\,(t' + \varepsilon'), & \text{in case that } \;\; dec\,(t_i) < dec\,(t + \varepsilon), \;\; \forall\, i = 1, \ldots, n\,. \end{array}$$

This way we obtain the same ordering of $dec\,(t' + \varepsilon')$ in $\mathcal{S}'_1$ as for $dec\,(t + \varepsilon)$ in $\mathcal{S}_1$.
Here $int\,(x)$ is the integer part of x and $dec\,(x)$ is the decimal part of x. Advancing time in $\mathcal{S}'$ for such an ε' results in configuration $\mathcal{S}'_1$ that is equivalent to $\mathcal{S}_1$.

Indeed, none of the time constraints involving global time and equality, such as constraint $t_i = t \pm D$ *i.e.*, $t'_i = t' \pm D$, is satisfied since D is an integer and the decimal parts of both global times t and t' are different from the decimal part of any other fact in the configuration. Therefore, both $t_i - t$ and $t'_i - t'$ are not integers.

For the time constraints involving the global time and inequality, we consider the constraint of type $t_k > t \pm D_i$. The proof for the case of constraint of type $t_k < t \pm D_i$ is analogous.
From $\mathcal{S} \sim \mathcal{S}'$ for all integers $B \leq D_{max}$ we know that for all $k \in \{1, \ldots, n\}$

$$t_k > t \pm B \quad \text{iff} \quad {t_k}' > t' \pm B\,. \tag{9}$$

We need to prove that

$$t_k > (t + \varepsilon) \pm D \quad \text{iff} \quad {t_k}' > (t' + \varepsilon') \pm D \tag{10}$$

for all integers $D < D_{max}$. Notice that $t + \varepsilon \pm D = t + dec\,(\varepsilon) + int\,(\varepsilon) \pm D$ involves possibly an integer $int\,(\varepsilon) + D$ greater than D_{max}, for which (9) may not hold. Similarly, it is the case for integer $int\,(\varepsilon') + D$.

We will, therefore, assume that $\varepsilon < 1$, *i.e.*, $int\,(\varepsilon) = 0$. Equivalently, we could split an arbitrary ε into a finite number of ε_i such that $\sum \varepsilon_i = \varepsilon$, and prove the result for ε by induction.
From (9) and from

$$int\,(t) \leq int\,(t + \varepsilon) \leq int\,(t) + 1$$

we can conclude that

$$int\,(t_k) \geq int\,(t + \varepsilon) \pm D = int\,(t + \varepsilon \pm D) \quad \text{iff} \quad int\,(t_k)' \geq int\,(t' + \varepsilon') \pm D = int\,(t' + \varepsilon' \pm D)\,. \tag{11}$$

Additionally,

$$dec\,(t_k) > dec\,(t + \varepsilon) \quad \text{iff} \quad dec\,(t'_k) > dec\,(t' + \varepsilon') \tag{12}$$

holds because ε' is chosen so that the ordering of decimal parts of timestamps in $\mathcal{S}_1$ and $\mathcal{S}'_1$ is the same. That is, either

$$dec\,(t_i) < dec\,(t+\varepsilon) < dec\,(t_j) \quad \text{and} \quad dec\,(t'_i) < dec\,(t'+\varepsilon') < dec\,(t'_j)$$

for some $Q_i@t_i, Q_j@t_j$ that immediately precede and follow the fact $Time@(t+\varepsilon)$ when sorting the facts in $\mathcal{S}$ only by the decimal parts of their timestamps, or it is the case that

$$dec\,(t_i) < dec\,(t+\varepsilon) \quad \text{and} \quad dec\,(t'_i) < dec\,(t'+\varepsilon')\,, \quad \forall i \in \{1,\ldots,n\}\,.$$

From (11) and (12) we obtain the claim (10). □

Following the above Proposition, we now relate the equivalence of configurations given by Definition 3.6 and the reachability problem.

Theorem 3.8. *Let $\mathcal{S}_I$ and $\mathcal{S}'_I$ be two equivalent initial configurations, $\mathcal{S}_G$ be a goal and $\mathcal{R}$ a set of actions. Let D_{max} be an upper bound on the numbers in $\mathcal{R}$, $\mathcal{S}_I$, $\mathcal{S}'_I$ and $\mathcal{S}_G$. Then the reachability problem with $\mathcal{S}_I, \mathcal{S}_G$ and $\mathcal{R}$ is solvable if and only if the reachability problem with $\mathcal{S}'_I, \mathcal{S}_G$ and $\mathcal{R}$ is solvable.*

Proof. ► Suppose the reachability problem with $\mathcal{S}_I, \mathcal{S}_G$ has a solution. We prove the existence of the solution to the reachability problem with $\mathcal{S}'_I, \mathcal{S}_G$ by induction on the length of the given plan from $\mathcal{S}_I$ to $\mathcal{S}_G$. From Proposition 3.7 it follows that the plan from $\mathcal{S}_I$ to $\mathcal{S}_G$ and the plan from $\mathcal{S}'_I$ to $\mathcal{S}'_G$ contain exactly the same actions, possibly using different instances. It also follows that $\mathcal{S}_i \sim \mathcal{S}'_i$ for each configuration $\mathcal{S}_i$ along the plan. This is depicted in the following diagram:

$$\begin{tikzcd}
\mathcal{S}_I \arrow[r, "r_1"'] \arrow[d, phantom, "\wr"] & \cdots \arrow[r, "r_{i-1}"'] & \mathcal{S}_{i-1} \arrow[r, "r_i"'] \arrow[d, phantom, "\wr"] & \mathcal{S}_i \arrow[r, "r_{i+1}"'] \arrow[d, phantom, "\wr"] & \cdots \arrow[r, "r_n"'] & \mathcal{S}_G \arrow[d, phantom, "\wr"] \\
\mathcal{S}'_I \arrow[r, "r_1"'] & \cdots \arrow[r, "r_{i-1}"'] & \mathcal{S}'_{i-1} \arrow[r, "r_i"'] & \mathcal{S}'_i \arrow[r, "r_{i+1}"'] & \cdots \arrow[r, "r_n"'] & \mathcal{S}'_G
\end{tikzcd}$$

Since $\mathcal{S}_G \sim \mathcal{S}'_G$, these configurations both satisfy the same set of constraints, therefore $\mathcal{S}_G$ is a goal configuration iff $\mathcal{S}'_G$ is a goal configuration. Hence, the reachability problem with $\mathcal{S}_I, \mathcal{S}_G$ and $\mathcal{R}$ has a solution if and only if the reachability problem with $\mathcal{S}'_I, \mathcal{S}_G$ and $\mathcal{R}$ has a solution. ◄ □

3.4. Distance Bounding Protocol Formalization

To demonstrate how our model can capture the attack in-between-ticks, consider the following protocol, called DB. This protocol captures the time challenge of distance bounding protocols.[5] Verifier should allow the access to his resources only if the measured round trip time of messages in the distance-bounding phase of the protocol does not exceed the given bounding time R. We assume that the verifier and the prover have already exchanged nonces n_P and n_V:

$$\begin{array}{ll}
V \longrightarrow P : n_P & \text{at time } t_0 \\
P \longrightarrow V : n_V & \text{at time } t_1 \\
V \longrightarrow P : OK(P) & \text{iff } t_1 - t_0 \leq R
\end{array}$$

[5]Another specification that includes an intruder model, keys, and the specification of the attack described in [3] can be found in our workshop paper [23].

Encoding of verifier's clock The fact $Clock_V@T$ denotes the local clock of the verifier *i.e.*, the discrete time clock that verifier uses to measure the response time in the distance bounding phase of the protocol.

We encode ticking of verifier's clock in *discrete units of time*. Action (13) represents the ticking of verifier's clock:

$$Time@T,\ Clock_V@T_1 \mid \{\ T = T_1 + 1\ \} \longrightarrow Time@T,\ Clock_V@T \tag{13}$$

Notice that if this action is not executed and T advances too much, *i.e.*, $T > T_1$, it means that the verifier clock stopped as it no longer advances.

Network Let $D(X, Y) = D(Y, X)$ be the integer representing the minimum time needed for a message to reach Y from X. We also assume that participants do not move. Rule (14) models network transmission from X to some Y:

$$Time@T,\ \mathcal{N}_X^S\ (m)@T_1,\ E@T_2\ \mid \{\ T \geq T_1 + D(X, Y)\ \} \longrightarrow\ Time@T,\ \mathcal{N}_X^S\ (m)@T_1,\ \mathcal{N}_Y^R\ (m)@T \tag{14}$$

Facts $\mathcal{N}_X^S(m)$ and $\mathcal{N}_X^R(m)$ specify that the participant X has sent and may receive the message m, respectively. Once X has sent the message m, that message can only be received by Y once it traveled from X to Y. The fact E is an empty fact which can be interpreted as a slot of resource. This is a technical device used to turn a theory balanced. See [22] for more details.

Remark 3.9. *Notice that in the rule (14) the fact* $\mathcal{N}_X^S(m)@T_1$ *is not consumed. This models the transmission media, such as radio frequency, where messages are not consumed by recipients. In such media, and as modelled by the rule (14), it is possible for multiple participants to receive the same message m. Alternatively, as in the classical (wire) network communication, the messages are removed from the network. In our formal model we are able to represent such transmission media as well,* e.g., *using the following rule*

$$Time@T,\ \mathcal{N}_X^S\ (m)@T_1\ \mid \{\ T \geq T_1 + D(X, Y)\ \} \longrightarrow\ Time@T,\ \mathcal{N}_Y^R\ (m)@T\ .$$

Measuring the round trip time of messages A protocol run creates facts denoting times when messages of the distance bounding phase are sent and received by the verifier. Predicates *Start* and *Stop* denote the actual (real) time of these events so that the round trip time of messages is $T_2 - T_1$ for timestamps T_1, T_2 in $Start(m)@T_1$, $Stop(m)@T_2$. On the other hand predicates $Start_V$ and $Stop_V$ model the verifier's view of time: $T_2 - T_1$, for T_1, T_2 in $Start_V(m)@T_1$, $Stop_V(m)@T_2$.

Protocol Theory Our example protocol DB is formalized in Figure 4. The first rule specifies that the verifier has sent a nonce and still needs to mark the time, specified by the fact $V_1(\text{pending}, P, N_P, N_V)@T$. The second rule specifies verifier's instruction of remembering the current time. The third rule specifies prover's response to the verifier's challenge. The fourth and fifth rules are similar to the first two, specifying when verifier actually receives prover's response and when he executes the instruction to remember the time. Finally, the sixth rule specifies that the verifier grants access to the prover if he believes that the distance to the prover is under the given bound.

Attack In-Between-Ticks

We now show how attack in-between-ticks is detected in our formalization.

$$
\begin{array}{l}
Time@T,\ V_0\ (P, N_P, N_V)@T_1,\ E@T_2,\ E@T_3 \longrightarrow \\
\qquad\qquad Time@T,\ V_1\ (\text{pending}, P, N_P, N_V)@T,\ \mathcal{N}_V^S\ (N_P)@T,\ Start\ (P, N_P, N_V)@T \\
Time@T,\ V_1\ (\text{pending}, P, N_P, N_V)@T_1,\ Clock_V@T,\ P@T_2\ \mid\ \{\ T \geq T_1\ \} \longrightarrow \\
\qquad\qquad Time@T,\ V_1\ (start, P, N_P, N_V)@T,\ Clock_V@T,\ Start_V\ (P, N_P, N_V)@T \\
Time@T,\ P_0\ (V, N_V, N_P)@T_1,\ \mathcal{N}_P^R\ (N_P)@T_2\ \mid\ \{\ T \geq T_2\ \} \longrightarrow \\
\qquad\qquad Time@T,\ P_1\ (V, N_V, N_P)@T,\ \mathcal{N}_P^S\ (N_V)@T \\
Time@T,\ V_1\ (\text{start}, P, N_P, N_V)@T_1,\ \mathcal{N}_V^R\ (N_V)@T_2 \longrightarrow \\
\qquad\qquad Time@T,\ V_2\ (\text{pending}, P, N_P, N_V)@T,\ Stop\ (P, N_P, N_V)@T \\
Time@T,\ V_2\ (\text{pending}, P, N_P, N_V)@T_1,\ Clock_V@T,\ E@T_2\ \mid\ \{\ T \geq T_1\ \} \longrightarrow \\
\qquad\qquad Time@T,\ V_2\ (\text{stop}, P, N_P, N_V)@T,\ Clock_V@T,\ Stop_V\ (P, N_P, N_V)@T \\
Time@T,\ Start_V\ (P, N_P, N_V)@T_1,\ Stop_V\ (P, N_P, N_V)@T_2,\ V_2\ (\text{stop}, P, N_P, N_V)@T_3\ \mid \\
\qquad\qquad \{\ T_2 - T_1 \leq R,\ T \geq T_3\ \} \longrightarrow\ Time@T,\ V_3\ (P)@T,\ \mathcal{N}_V^S\ (Ok(P))@T,\ E@T
\end{array}
$$

Fig. 4. Protocol Rules for DB protocol

The initial configuration contains facts $Time@0$, $Clock_V@0$ denoting that global time and time on verifier's discrete time are initially set to 0.

Given the protocol specification in Figure 4, attack in-between-ticks is represented with the following configuration:

$$\{\ Start\ (P, N_P, N_V)@T_1,\ Stop\ (P, N_P, N_V)@T_2,\ \mathcal{N}_V^S\ (Ok(P))@T_3\ \}\ \mid\ \{\ T_2 - T_1 > R\ \}$$

It denotes that in the session involving nonces N_P, N_V the verifier V has allowed the access to prover P although the distance requirement has been violated.

Notice that such an anomaly is really possible in this specification. Consider the following example: verifiers actually sends the first message at time 1.7 while the prover responds at time 4.9. Between moments 1.7 and 4.9, there would be 3 ticks on the verifier's clock. The verifier would consider starting time of 2 and finishing time of 5, and confirm with the time bound $R = 3$. Actually, the real round trip time is greater than the time bound, namely $4.9 - 1.7 = 3.2$. Following facts would appear in the configuration:

$$Start_V\ (n)@2,\ Stop_V\ (n)@5,\ Start\ (n)@1.7,\ Stop\ (n)@4.9\ .$$

Since $5 - 2 = 3$ the last rule from Figure 4, the accepting rule, would apply resulting in the configuration containing the facts:

$$Start\ (p, n_P, n_V)@1.7,\ Stop\ (p, n_P, n_V)@4.9,\ \mathcal{N}_V^S\ (Ok(p))@5\ .$$

Since $4.9 - 1.7 = 3.2$ is greater than $R = 3$, this configuration constitutes an attack.

Finally, notice as well that in our formalization in Figure 4, prover immediately responds to the received challenge message, see the third rule in Figure 4. In reality there is some non-zero time of processing of messages. This could cause additional discrepancy between the actual and the measured round trip time. Namely, once the prover receives the challenge message he needs to check whether the nonce received is the agreed value, previously exchanged with the verifier. Then, he needs to compose the response message for sending, using the stored nonce value. This process would take some additional time which may be calculated in the design of the distance bounding protocol itself, *i.e.*, in the established bounding time of the protocol, R. The difference between expected and actual processing time can be the source of further inaccuracy.

4. Attack in Between Ticks - A Full Probabilistic Analysis

In Section 2.2 we presented the novel attack which we believe can be carried out on most distance bounding protocols. Our symbolic model described in Section 3 formally demonstrates that the attack can, in principle, happen. In this section we investigate how likely it is for an attacker to carry out such an attack. We explicitly calculate the probability of such an erroneous "acceptance event" happening based on a single challenge/response time measurement. Such a "measurement phase" is an indispensable part of any distance bounding protocol and a basis for the ultimate verifier's decision of whether to grant the access or not.

Our main result of this section (Theorem 4.3) is to show that the attack is not so unlikely when the prover is beyond the established perimeter up to a distance corresponding to a half-tick of the verifier, *i.e.*, 3 meters in the scenario discussed in Section 2. The probability of distance measurement error is of 1/2. This probability, however, reduces to zero once the prover is further away. That is, probability of error is zero when the prover exceeds the perimeter at the distance corresponding to one tick or more.

The attack in-between-ticks is based on the discrepancy between the *observable* time interval $t_1 - t_0$, (between the moment t_0 when the time of sending the challenge message is recorded, and the moment t_1 when the time of receiving the response message is recorded) and the *actual* time interval $s_1 - s_0$. Here, s_0, s_1, t_0 and t_1 respectively denote the actual time when the challenge message is sent, and the actual time of receiving the response, the recorded time of sending the challenge message, and the recorded time of receiving the response message.

For our probabilistic analysis we consider the *challenge-response protocol* in which the verifier reacts as quick as possible but within the time constraints that *only one operation can be executed in one clock cycle*.[6]

In a round of the challenge-response protocol the verifier performs the following actions:

(1) At a moment s_0 within an initial clock cycle 1, say $s_0 = 1 + X$, verifier sends a challenge message m. Here X is a random variable distributed on the interval $[0, \frac{1}{2}]$ with its probability density f_X.
(2) *Just after that* - that is, at a moment t_0 within the next clock cycle 2, say $t_0 = 2 + Y$, verifier records the fact that m has been sent. Here Y is a random variable distributed on the interval $[0, \frac{1}{2}]$ with its probability density f_Y.
(3) At a moment s_1 within the corresponding clock cycle $\lfloor s_1 \rfloor$, say $s_1 = s_0 + \ell$, verifier receives a response message m'.

[6]From the performance point of view, the difference between discrete time and dense time is that, in contrast with dense time, only *a fixed finite number of events may occur within a bounded time interval* in the case of discrete time. Without loss of generality, we allow here no more than one action be performed in one clock cycle.

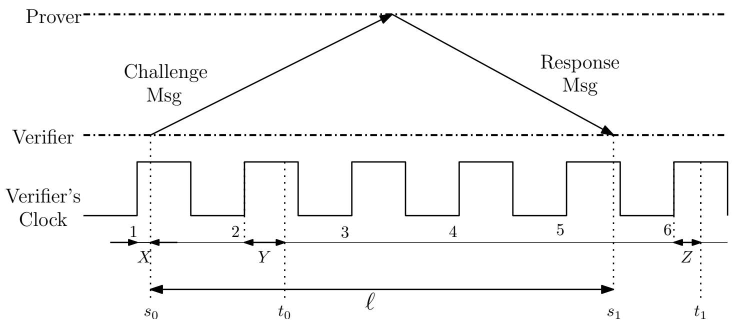


Fig. 5. In different ticks (Sequential Execution)

(4) *Just after that*, that is, at a moment t_1 within the next clock cycle $\lfloor s_1 \rfloor + 1$, say $t_1 = (\lfloor s_1 \rfloor + 1) + Z$, verifier records the fact that m' has been received. Here Z is a random variable distributed on the interval $[0, \frac{1}{2}]$ with its probability density f_Z.

For the sake of perspicuity, we assume X, Y, and Z be independent random variables uniformly distributed on the interval $[0, \frac{1}{2}]$.
Thus, we are dealing with the model given by the system as illustrated in Figure 5:

$$\begin{aligned} s_0 &= 1 + X, & s_1 &= s_0 + \ell, \\ t_0 &= 2 + Y, & t_1 &= \lfloor s_1 \rfloor + 1 + Z. \end{aligned} \tag{15}$$

The decision rule applied by the verifier is described bellow.

Definition 4.1. *For a fixed time response bound, an integer R, verifier decides to grant the access to its resources if and only if the following holds for the* measured *time interval* $t_1 - t_0$*:*

$$t_1 - t_0 \leq R.$$

Thus, the "Yes" decision taken by the verifier is erroneous if in reality the actual distance between verifier and prover, $s_1 - s_0$, turns out to be larger than R, say by some positive value h.

We now investigate the probability of such an event actually occurring. Firstly, we define the required probability.

Definition 4.2. *For a fixed time response bound, an integer R, and an extra, a positive h, we define the* probability of the erroneous decision to grant the access , $p_{error}(R, h)$,

$$p_{error}(R, h) = Prob\{\, t_1 - t_0 \leq R \;/\; s_1 - s_0 = R + h \,\} \tag{16}$$

as the conditional probability of an "acceptance event" of the form $t_1 - t_0 \leq R$, *given that* $s_1 - s_0 = R + h$.

We calculate the probability of the *erroneous decision*, $p_{error}(R, h)$, and obtain the explicit values of probabilities as stated in the theorem bellow. Recall that the protocol time response bound R and the actual challenge-response time ℓ denote the respective round trip time of messages, and that in case of an erroneous "Yes" decision taken by the verifier, the value of h, $h = \ell - R$, is positive. In such a case prover that is outside of the perimeter specified by the value $\frac{R}{2}$ appears within the perimeter from the point of view of the verifier.

The obtained results are visualized in Figure 6 which shows how the conditional probability of erroneous decision, $p_{error}(R, h)$, is classified w.r.t. time distances between the verifier and the prover. This probability is non-zero when the prover is in the zone close to the perimeter, on the outskirts up to $\frac{1}{2}$ tick time distance which is 1 tick in the round trip time.

Theorem 4.3. *Let X, Y and Z be independent random variables uniformly distributed on* $[0, \frac{1}{2}]$. *Then, for a fixed time response bound, an integer R, and an extra, a positive h, the probability of the erroneous decision to grant the access,* $p_{error}(R, h)$, *is given by*

$$p_{error}(R, h) = \begin{cases} \frac{1}{2}, & \textit{if} \;\; 0 < h \leq \frac{1}{2}, \\ 1 - h, & \textit{if} \;\; \frac{1}{2} < h < 1, \\ 0, & \textit{if} \;\; h \geq 1. \end{cases} \tag{17}$$

Theorem 4.3 provides the explicit probability of verifier making an erroneous decision to grant access to a prover that is located outside of the perimeter specified by the protocol distance bound, R.

In particular, for $\frac{1}{2} < h < 1$, *i.e.*, when the prover is between "half a tick" and "a single tick" further away form the specified perimeter, the probability of the erroneous decision decreases with h. In the case that the distance of the prover exceeds the upper bound by a "tick" or more, the probability of verifier

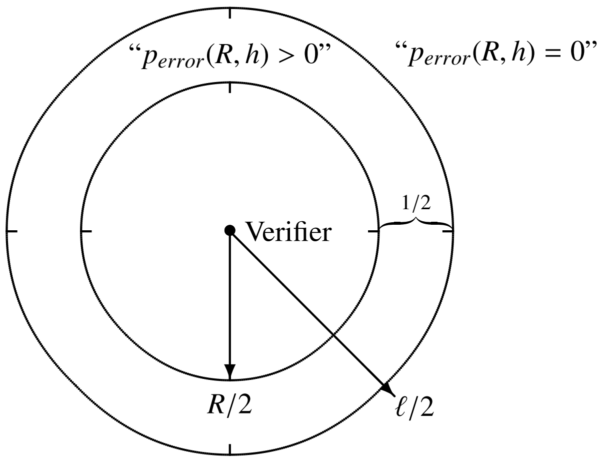


Fig. 6. Towards Theorem 4.3. Conditional probability of erroneous decision $p_{error}(R, h)$ classified w.r.t. round trip time distances, R and ℓ, between verifier and prover. The Verifier is at the center of the circles. The inner circle represents the distance bounding area. The outer circle is the actual area where Prover can be granted access with probability greater than zero.

making the erroneous decision is zero. This is not surprising, given the nature of the attack in-between-ticks.

However, notice that, contrary to our expectations, the probability of the *erroneous decision* turns out to be 50% for any $0 < h \leq \frac{1}{2}$. That is, the probability of error is rather high when the prover is close to the bound, under "half a tick" distance form the perimeter. This extra time distance would amount to up to 3 meters in our example given earlier in the paper, which is not negligible.

The remainder of this section contains the proof of Theorem 4.3. A reader that is not that interested in this more technical part of this section, can jump to Section 5.

4.1. Proof of Theorem 4.3

In order to prove Theorem 4.3 we introduce some auxiliary machinery.

Definition 4.4. *To investigate $p_{error}(R, h)$, we introduce the following distribution function $F_\ell(x)$*

$$F_\ell(x) = Prob\{\, t_1 - t_0 \leq x \;/\; s_1 - s_0 = \ell \,\} \tag{18}$$

defined as the conditional probability of the event $t_1 - t_0 \leq x$, given the actual time interval $s_1 - s_0 = \ell$.

In Figures 7 and 8 we illustrate the two cases of the graph of the conditional probability density, the derivative $F'_\ell(x)$, for the distribution function $F_\ell(x)$.

Notice that, with $\ell = R + h$, we have: $p_{error}(R, h) = F_\ell(R) = \int_{-\infty}^{R} F'_\ell(x)\,dx.$

The following lemmas provide an explicit expression for the distribution function $F_\ell(x)$ and its density $F'_\ell(x)$. Let here, and henceforth, $\widetilde{\ell}$ denote the decimal part of ℓ: $\widetilde{\ell} = \ell - \lfloor \ell \rfloor$.

Lemma 4.5. *In the model (15) we are dealing with the observable period of time $t_1 - t_0$ that is calculated as:*

$$t_1 - t_0 = \lfloor X + \widetilde{\ell} \rfloor + \lfloor \ell \rfloor + Z - Y = \begin{cases} \lfloor \ell \rfloor + Z - Y, & \text{if } \widetilde{\ell} < \frac{1}{2}, \\ \lfloor \ell \rfloor + Z - Y, & \text{if } \widetilde{\ell} \geq \frac{1}{2} \text{ but } X + \widetilde{\ell} < 1, \\ 1 + \lfloor \ell \rfloor + Z - Y, & \text{if } \widetilde{\ell} \geq \frac{1}{2} \text{ and } X + \widetilde{\ell} \geq 1. \end{cases} \tag{19}$$

Proof. By simple calculation,

$$t_1 - t_0 = \lfloor 1 + X + \ell \rfloor + 1 + Z - (2 + Y) = \lfloor X + \widetilde{\ell} \rfloor + \lfloor \ell \rfloor + Z - Y$$

□

Lemma 4.6. *Let X, Y and Z be independent random variables uniformly distributed on* $[0, \frac{1}{2}]$. *Then for the distribution function $F_\ell(x)$ given by $F_\ell(x) = Prob\{\, t_1 - t_0 \leq x \;/\; s_1 - s_0 = \ell \,\}$, the following holds:*

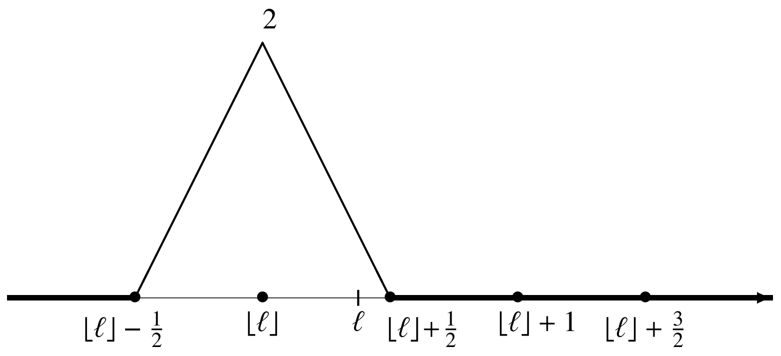


Fig. 7. The graph of the probability density, $F'_\ell(x)$, for the distribution $F_\ell(x)$ - The single-humped ("Dromedary camel") case: $\widetilde{\ell} = \ell - \lfloor\ell\rfloor < \frac{1}{2}$.

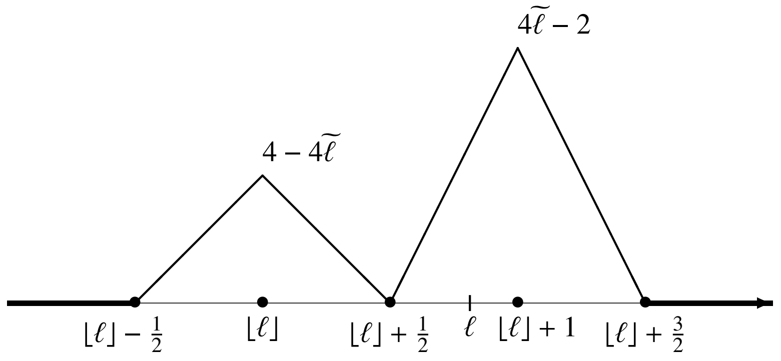


Fig. 8. The graph of the probability density, $F'_\ell(x)$, for the distribution function $F_\ell(x)$ - The 2-humped ("Bactrian camel") case of bimodal distribution: $\widetilde{\ell} = \ell - \lfloor\ell\rfloor > \frac{1}{2}$.

(i) In the case of $\widetilde{\ell} = \ell - \lfloor\ell\rfloor < \frac{1}{2}$,

$$F_\ell(x) = Prob\{\, Z - Y \leq x - \lfloor\ell\rfloor \,\} \tag{20}$$

(ii) In the case of $\widetilde{\ell} = \ell - \lfloor\ell\rfloor \geq \frac{1}{2}$,

$$F_\ell(x) = (2 - 2\widetilde{\ell}) \cdot Prob\{\, Z - Y \leq x - \lfloor\ell\rfloor \,\} \;+\; (2\widetilde{\ell} - 1) \cdot Prob\{\, Z - Y \leq x - \lfloor\ell\rfloor - 1 \,\} \tag{21}$$

Proof. Given the condition $s_1 - s_0 = \ell$, we deal the following two cases.
(i) In the case of $\widetilde{\ell} = \ell - \lfloor\ell\rfloor < \frac{1}{2}$, by Lemma 4.5, $t_1 - t_0 = \lfloor\ell\rfloor + Z - Y$, and, respectively,

$$F_\ell(x) = Prob\{\, \lfloor\ell\rfloor + Z - Y \leq x \,\} = Prob\{\, Z - Y \leq x - \lfloor\ell\rfloor \,\}$$

In Figure 7 we draw the graph of the conditional probability density, $F'_\ell(x)$, for the distribution function $F_\ell(x)$ in the case of the uniformly distributed Z and Y. The height of the triangle there is 2.

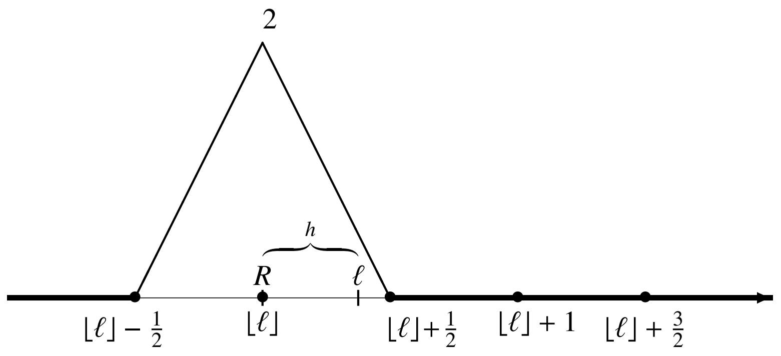


Fig. 9. Probability of the erroneous decision to grant the access, $p_{error}(R, h)$ - The single-humped case.

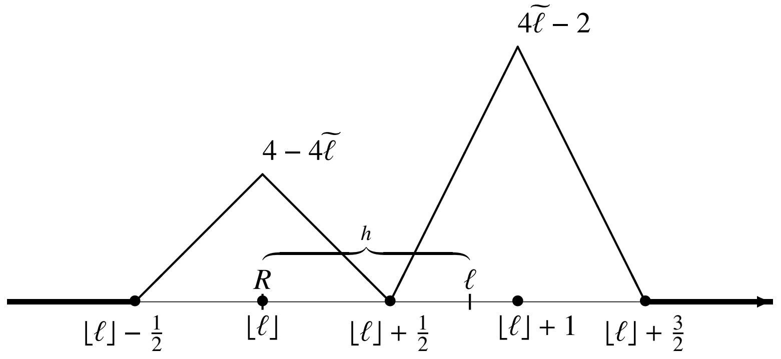


Fig. 10. Probability of the erroneous decision to grant the access, $p_{error}(R, h)$ - The 2-humped ("Bactrian camel") case.

(ii) In the case of $\widetilde{\ell} = \ell - \lfloor \ell \rfloor \geq \frac{1}{2}$, by Lemma 4.5 we have:

$$F_\ell(x) = Prob\{X + \widetilde{\ell} < 1\} \cdot Prob\{\, \lfloor \ell \rfloor + Z - Y \leq x \,\} \;+\; Prob\{X + \widetilde{\ell} \geq 1\} \cdot Prob\{\, 1 + \lfloor \ell \rfloor + Z - Y \leq x \,\}.$$

Notice that for the X uniformly distributed on $[0, \frac{1}{2}]$:

$$Prob\{X + \widetilde{\ell} < 1\} = Prob\{X < 1 - \widetilde{\ell}\} = 2(1 - \widetilde{\ell}),$$

resulting in

$$F_\ell(x) = (2 - 2\widetilde{\ell}) \cdot Prob\{\, Z - Y \leq x - \lfloor \ell \rfloor \,\} \;+\; (2\widetilde{\ell} - 1) \cdot Prob\{\, Z - Y \leq x - \lfloor \ell \rfloor - 1 \,\}$$

In Figure 8 we draw the graph of the conditional probability density, the derivative $F'_\ell(x)$, for the distribution function $F_\ell(x)$ in the case of the uniformly distributed X, Z and Y. The height of the left triangle in Figure 8 is $4 - 4\widetilde{\ell}$, and the height of the right triangle is $4\widetilde{\ell} - 2$.

□

We are now ready to prove the main theorem.

Proof of Theorem 4.3.
Recall that X, Y and Z are independent random variables uniformly distributed on $[0, \frac{1}{2}]$.
Given an integer R, here $\ell = R + h$ and $h = \widetilde{\ell} = \ell - \lfloor \ell \rfloor$.

(i) In the case of $0 < h \leq \frac{1}{2}$, we have $\lfloor \ell \rfloor = R$,

$$h = \ell - R = \ell - \lfloor \ell \rfloor = \widetilde{\ell} \leq \frac{1}{2},$$

and, by Lemma 4.6 (see Figure 9)

$$p_{error}(R, h) = \int_{-\infty}^{\lfloor \ell \rfloor} F'_\ell(x)\, dx = \frac{1}{2}$$

(ii) In the case of $\frac{1}{2} < h < 1$, we have $\lfloor \ell \rfloor = R$,

$$h = \ell - R = \ell - \lfloor \ell \rfloor = \widetilde{\ell} > \frac{1}{2},$$

and, by Lemma 4.6 (see Figure 10)

$$p_{error}(R, h) = \int_{-\infty}^{\lfloor \ell \rfloor} F'_\ell(x)\, dx = \frac{1}{2} \cdot Prob\{X + \widetilde{\ell} < 1\} = \frac{1}{2} \cdot 2(1 - \widetilde{\ell}) = 1 - h$$

(iii) Lastly, in the case of $h > 1$, we have $R \leq \lfloor \ell \rfloor - 1$, and (see Figures 9 and 10)

$$p_{error}(R, h) \leq \int_{-\infty}^{\lfloor \ell \rfloor - 1} F'_\ell(x)\, dx = 0.$$

which completes the proof of Theorem 4.3. □

5. Implementation in Maude

We have formalized the scenario of the attack in-between-ticks in an extension with SMT-solver of the rewriting logic tool Maude. The tool was able to automatically find this attack. We considered a scenario with two players, a verifier (`v`) and a prover (`p`) such that messages take `dvp` time units to navigate from one another. The function `recSend(PV)` returns `v` if `PV` is `p` and vice-versa.

Although the specification detailed in Section 3.4 could be specified in Maude, it would be impractical to use it to search for an attack as the state space is infinite. Moreover, using the machinery of circle-configurations described in Section 6, although resulting in a finite search space, is still intractable.

Instead, we show in this section that by using constrained variables and relying on SMT solvers allows us to verify distance bounding protocols in practice. During verification (search), the variables in the time constraints are not instantiated, but are accumulated. The SMT solver is, then, used to determine whether a set of accumulated constraints (in a branch of search) is consistent. If it is consistent, then search may proceed; otherwise, Maude backtracks and continues with the verification.

```
crl [Network]:
  { S (Time @ T) (Ns(PV,M) @ T1) } =>
  { S (Time @ T2) (Nr(VP,M) @ T2) }
 if (T2 >= T1 + dvp and (T2 >= T)) = true /\ VP := recSend(PV) [nonexec] .

crl [Tick]:
  { S (Time @ T) (vTime @ T1) } =>
  { S (Time @ T2) (vTime @ T1) }
 if (T2 > T and (T2 < T1 + 1/1))  = true [nonexec] .

crl [Tick-Vclock]:
  { S (Time @ T) (vTime @ T1) } =>
  { S (Time @ T1 + 1/1) (vTime @ T1 + 1/1) }
 if (T1 + 1/1 >= T)  = true [nonexec] .

crl [Real-V-Send]:
  { S (Time @ T) (V0(P,NP,NV) @ T1)  } =>
  { S (Time @ T) (V1(pending,P,NP,NV) @ T)
    (Start(P,NP,NV) @ T) (Ns(p,NP) @ T ) }
 if (T1 + 1/1 > T and T >= T1 )  = true [nonexec] .

crl [Real-P-Rcv]:
  { S (Time @ T) (P0(V,NV,NP) @ T1) (Nr(p,NP) @ T2) } =>
  { S (Time @ T3) (P1(V,NV,NP) @ T3) (Ns(p,NV) @ T3)}
 if ((T2 >= T) and (T3 >= T2)) = true [nonexec] .

crl [Real-V-Rec]:
{ S (Time @ T) (V1(start,P,NP,NV) @ T1) (Nr(v,NV) @ T2 ) } =>
{ S (Time @ T2) (V2(pending,P,NP,NV) @ T2) (Stop(P,NP,NV) @ T2)  }
if (T2 >= T ) = true [nonexec] .

crl [Dis-V-Rec]:
  { S (Time @ T) (vTime @ T1) (V2(pending,P,NP,NV) @ T2) } =>
  {S (Time @ toReal(toInteger(T2)) + 1/1) (vTime @ toReal(toInteger(T2)) + 1/1)
   (V2(stop,P,NP,NV) @ toReal(toInteger(T2)) + 1/1)
   (StopV(P,NP,NV) @ toReal(toInteger(T2)) + 1/1) }
 if ( (T2 >= T1)) = true [nonexec] .

crl [ok]:
  { S (Time @ T) (StartV(P,NP,NV) @ T1) (StopV(P,NP,NV) @ T2) (V2(stop,P,NP,NV) @ T3)} =>
  {S (Time @ T) (Ok(P) @ T) }
if (T2 - T1 <= 2/1 * 3/1)  = true [nonexec] .
```

Fig. 11. Rewrite Rules Specification of a Distance Bounding Protocol in Maude

The modifications to the theory in Section 3.4 are minor simply those involving the constrained variables. The Maude rewrite rules are depicted in Figure 11. To illustrate the use of SMT, consider the `Network` rule. It specifies that a message sent `Ns(P,M)` at time `T1` can be received at any time `T2` such that `T2 >= T1 + dvp` and `T2 >= T`. Notice that Maude does not instantiate these time variables with concrete values, but only with time symbols accumulating time constraints. Maude backtracks whenever the collection of accumulated constraints is unsatisfiable.

Our formalization encodes the Tick rule in the MSR theory which advances global time by any real number using two rewrite rules `Tick` and `Tick-Vclock`. The former rewrite rule advances time to any value `T2` within the verifier's current clock cycle. This is specified by the constraint `T2 < T1 + 1/1`. The second rewrite advances time to the beginning of the next verifier's clock cycle (`T1 + 1/1`). Intuitively, we consider a verifier's clock cycle as an event thus controlling how time advances and avoiding state space explosion.

A second difference to the the theory in Section 3.4 is that we assume a powerful verifier which measures the time of sending and receiving a message exactly at the beginning of clock cycle following the time when message is actually sent and received. This is specified, for example, by the rule `Dis-V-Rec`, in particular, by using the timestamp `toReal(toInteger(T2)) + 1/1` where `1/1` represents the real number one. If there is an attack with this more powerful verifier, then there is an attack in a less powerful verifier, *i.e.*, it is sound.

Finally, the third difference is on the way time advances. While it is convenient to separate the tick rules from the instantaneous rules in our theoretical framework as described in Section 3, this distinction increases considerably search space. Instead, we advance time according to the events processed. This is similar to the behavior of Real-Time Maude [35], but here we use time symbols. Thus, whenever a message is received or sent or whenever the verifier's clock ticks, global time advances to the time of the corresponding event. For example, in rule `Real-P-Rcv`, the global time advances to a time `T3` greater than the time, `T2`, when a message is received. This time sampling is sound as no events are processed before they should, *e.g.*, a message is not received before the corresponding time to travel elapses.

For completeness, it seems possible to apply results from the literature, *e.g.*, the completeness results of the time sampling used by Real-Time Maude [36]. More recently, Nigam *et al.* [34] have proved the completeness of time intruders using symbolic time constraints.

We used the `smt-search` to find a symbolic representation of a family of attacks. Using Maude SMT the potentially infinite search space becomes finite, by treating the distance between the verifier and the prover as a constrained variable.

As an example, we considered the distance bound to be 3 and set `(dva > 3/1) = true` which means that the prover should not succeed the distance bound challenge as shown below:

```
Maude> smt-search [1]
{ (Time @ 0/1) (vTime @ 0/1) (V0(p,n(0),n(10)) @ 0/1) (P0(v,n(10),n(0)) @ 0/1) (dist(dvp)) }
 =>+
{ (Ok(p) @ T:Real) S:Soup }
such that (dva > 3/1) = true .
```

The following (simplified) solution is obtained after few seconds (ca 10 seconds) of computation:

```
S --> ...
(Time @ toReal(toInteger(#3-T2:Real)) + 1/1)
```

```
(Start(p, n(0), n(10)) @ 0/1) (Stop(p, n(0), n(10)) @ #3-T2:Real)
V2(stop,p,n(0),n(10)) @ toReal(toInteger(#3-T2:Real)) + 1/1
where dvp > 3/1
...
and (#1-T2:Real >= 0/1 + dvp) and (#1-T2:Real >= #1-T2:Real and #2-T3:Real >= #1-T2:Real)
and (#3-T2:Real >= #2-T3:Real + dvp) and #3-T2:Real >= #3-T2:Real
and toReal(toInteger(#3-T2:Real)) + 1/1 - (0/1 + 1/1) <= 2/1 * 3/1
```

It states that the verifier grants the resource to the prover, as the message Ok has been sent, although the prover is further away than the distance bound of 3.

6. Circle-Configurations

This Section introduces the machinery, called Circle-Configurations, that can symbolically represent configurations and plans that mention dense time. Dealing with dense time leads to some difficulties, which have puzzled us for some time now, in particular, means to handle Zeno paradoxes. When we use discrete domains to represent time, such as the natural numbers, time always advances by one, specified by the rule:

$$Time@T \longrightarrow Time@(T+1)$$

There is no other choice.[7] On the other hand, when considering systems with dense time, the problem is much more involved, as the non-determinism is much harder to deal with: the value that the time advances, the ε in

$$Time@T \longrightarrow Time@(T+\varepsilon)$$

can be instantiated by any positive real number.

Our claim is that we can symbolically represent any plan involving dense time by using a canonical form called circle-configurations. We show that circle-configurations provide a sound and complete representation of plans with dense time (Theorem 6.4).

Recall Definition 3.6 and Theorem 3.8 where we show that the introduced equivalence between configurations corresponds to the reachability problem in the sense that a solution of a reachability problem is independent of the choice of equivalent (initial) configurations. Indeed, we will show that using circle-configurations we can symbolically represent the entire class of equivalent configurations and, moreover, we can consider reachability problem over circle-configurations.

A circle-configuration consists of two components: a *δ-Configuration*, Δ, and a *Unit Circle*, $\mathcal{U}$, written $\langle \Delta, \mathcal{U} \rangle$. Intuitively, the former accounts for the integer part of the timestamps of facts in the configuration, while the latter deals with the decimal part of the timestamps.

In order to define these components, however, we need some additional machinery. For a real-number, r, $int(r)$ denotes the integer part of r and $dec(r)$ its decimal part. For example, $int(2.12)$ is 2 and

[7]However, as time can always advance, a plan may use an unbounded number of natural numbers. This source of unboundedness was handled in our previous work [25]. This solution, however, does not scale to dense time.

$dec\,(2.12)$ is 0.12. Given a natural number D_{max}, the *truncated time difference (w.r.t. D_{max})* between two facts $P@t_P$ and $Q@t_Q$ such that $t_Q \geq t_P$ is defined as follows

$$\delta_{P,Q} = \begin{cases} int\,(t_Q) - int\,(t_P), \text{ if } int\,(t_Q) - int\,(t_P) \leq D_{max} \\ \infty, \text{ otherwise} \end{cases}$$

For example, if $D_{max} = 3$ for the facts $F@3.12,\ G@1.01,\ H@5.05$ we have $\delta_{F,H} = 2$ and $\delta_{G,H} = \infty$. Notice that whenever we have $\delta_{P,Q} = \infty$ for two timestamped facts, $P@t_P$ and $Q@t_Q$, we can infer that $t_Q > t_P + D$ for any natural number D in the theory. Thus, we can truncate time difference without sacrificing soundness and completeness. This was pretty much the idea used in [25] to handle systems with discrete-time.

δ-Configuration

We now explain the first component, Δ, of circle-configurations, $\langle \Delta, \mathcal{U} \rangle$, namely the δ-configuration, to only later enter into the details of the second component.

Given a configuration $\mathcal{S} = \{F_1@t_1, \ldots, F_n@t_n, Time@t\}$, we construct its δ-configuration as follows: We first sort the facts using the integer part of their timestamps, obtaining the sequence of timestamped facts $Q_1@t'_1, \ldots, Q_{n+1}@t'_{n+1}$, where $t'_i \leq t'_{i+1}$ for $1 \leq i \leq n+1$ and $\{Q_1, \ldots, Q_{n+1}\} = \{F_1, \ldots, F_n, Time\}$. We then aggregate in classes facts with the same integer part of the timestamps obtaining a sequence of classes

$$\{Q^1_1, \ldots, Q^1_{m_1}\}, \{Q^2_1, \ldots, Q^2_{m_2}\}, \ldots, \{Q^j_1, \ldots, Q^j_{m_j}\}\ ,$$

where $\delta_{Q^k_i, Q^k_j} = 0$ for any $1 \leq i \leq m_k$ and $1 \leq k \leq j$, and $\{Q^1_1, \ldots, Q^j_{m_j}\} = \{Q_1, \ldots, Q_{n+1}\}$.

The δ-configuration of $\mathcal{S}$ is then:

$$\Delta = \Big\langle\ \{Q^1_1, \ldots, Q^1_{m_1}\},\ \delta_{1,2},\ \{Q^2_1, \ldots, Q^2_{m_2}\}, \ldots,\ \{Q^{j-1}_1, \ldots,\ Q^{j-1}_{m_{j-1}}\},\ \delta_{j-1,j}, \{Q^j_1, \ldots,\ Q^j_{m_j}\}\ \Big\rangle$$

where $\delta_{i,i+1} = \delta_{Q^i_1, Q^{i+1}_1}$ is the truncated time difference between the facts in class i and class $i+1$. For such a δ-configuration, Δ, we define

$$\Delta(Q^l_i, Q^h_j) = \begin{cases} \sum\limits_{k=l}^{k=h-1} \delta_{k,k+1} & \text{if } h \geq l \\ -\sum\limits_{k=h}^{k=l-1} \delta_{k,k+1} & \text{otherwise} \end{cases}$$

which is the truncated time difference between any two facts of Δ, Q^l_i and Q^h_j from the classes l and h, respectively. Here we assume ∞ is the addition absorbing element, *i.e.*, $\infty + D = \infty$ for any natural number D and $\infty + \infty = \infty$.

Notice that, for a given upper bound D_{max}, different configurations may have the same δ-configuration. For example, with $D_{max} = 4$, configurations

$$\begin{aligned} \mathcal{S}_1 &= \{\ M@3.01,\ R@3.11,\ P@4.12,\ Time@11.12,\ Q@12.58,\ S@14\ \} \quad \text{and} \\ \mathcal{S}'_1 &= \{\ M@0.2,\ R@0.5,\ P@1.6,\ Time@6.57,\ Q@7.12,\ S@9.01\ \} \end{aligned} \qquad (22)$$

have both the following δ-configuration:

$$\Delta_{\mathcal{S}_1} = \langle\, \{M, R\}, 1, \{P\}, \infty, \{Time\}, 1, \{Q\}, 2, \{S\}\,\rangle.$$

This δ-configuration specifies the truncated time differences between the facts from $\mathcal{S}'_1$. For example, $\Delta_{\mathcal{S}_1}(R, P) = 1$, that is, the integer part of the timestamp of the fact P is ahead one unit with respect to the integer part of the timestamp of the fact R. Moreover, the timestamp of the fact $Time$ is more than D_{max} units ahead with respect to the timestamp of P. This is indeed true for both configurations $\mathcal{S}_1$ and $\mathcal{S}'_1$ given above.

Unit Circle

In order to handle the decimal part of the timestamps, we use intervals instead of concrete values. These intervals are represented by a circle, called Unit Circle, which together with a δ-configuration composes a circle-configuration. The unit circle of a configuration $\mathcal{S} = \{F_1@t_1, \ldots, F_n@t_n, Time@t\}$ is constructed by first ordering the facts from $\mathcal{S}$ according to the *decimal part* of their timestamps in the increasing order. In such a way we obtain the sequence of facts $Q_1, \ldots, Q_{n+1}$, where $\{Q_1, \ldots, Q_{n+1}\} = \{F_1, \ldots, F_n, Time\}$. Then the unit circle of the given configuration $\mathcal{S}$ is obtained by aggregating facts that have the same *decimal part* obtaining a sequence of classes:

$$\mathcal{U} = [\ \{Q_1^0, \ldots, Q_{m_0}^0\}_{\mathcal{Z}},\ \{Q_1^1, \ldots, Q_{m_1}^1\}, \ldots,\ \{Q_1^j, \ldots, Q_{m_j}^j\}\ ]$$

where $\{Q_1^1, \ldots, Q_{m_j}^j\} = \{Q_1, \ldots, Q_{n+1}\}$, the facts in the same class have the same decimal part, *i.e.*, $dec\,(Q_k^i) = dec\,(Q_l^i)$, for all $1 \leq k \leq m_i$, $1 \leq l \leq m_i$ and $1 \leq i \leq j$, classes are ordered in the increasing order, *i.e.*, $dec\,(Q_k^i) < dec\,(Q_l^{i'})$ for all $i \neq i'$, where $1 \leq k \leq m_i$, $1 \leq l \leq m_{i'}$, $0 \leq i \leq j$, $1 \leq i' \leq j$, and the first class $\{Q_1^0, \ldots, Q_{m_1}^0\}_{\mathcal{Z}}$, marked with the subscript $\mathcal{Z}$ contains all facts whose timestamp's decimal part is zero, *i.e.*, $dec\,(Q_i^0) = 0$, for $1 \leq i \leq m_0$.

We call the class $\{Q_1^0, \ldots, Q_{m_1}^0\}_{\mathcal{Z}}$ the *Zero Point*. Notice that the zero point may be empty.

For a unit circle, $\mathcal{U}$, we define: $\mathcal{U}(Q_j^i) = i$ to denote the class in which the fact Q_j^i appears in $\mathcal{U}$.

For example, the unit circle of configuration $\mathcal{S}_1$ given in (22) is the sequence:

$$\mathcal{U}_{\mathcal{S}_1} = [\,\{S\}_{\mathcal{Z}},\ \{M\},\ \{R\},\ \{P, Time\},\ \{Q\}\,]\ .$$

Notice that P and $Time$ are in the same class as the decimal parts of their timestamps are the same, namely 0.12. Moreover, we have that $\mathcal{U}_{\mathcal{S}_1}(S) = 0 < 2 = \mathcal{U}_{\mathcal{S}_1}(R)$, specifying that the decimal part of the timestamp of the fact R is greater than the decimal part of the timestamp of the fact S.

We will graphically represent a unit circle as shown in Figure 12. The (green) ellipse at the top of the circle marks the zero point, while the remaining classes are placed on the circle in the (red) squares ordered clockwise starting from the zero point. Thus, from the above graphical representation it can be seen that the decimal part of the timestamp of the fact Q_1^1 is smaller than the decimal of the timestamp of the fact Q_1^2, while the decimal part of the timestamps of the facts Q_1^i and Q_2^i are equal. The exact points where the squares are placed are not important, only their relative positions matter, *e.g.*, the square for the class containing the fact Q_1^1 should be placed on the circle somewhere in between the zero point and the square for the class containing the fact Q_1^2, clockwise.

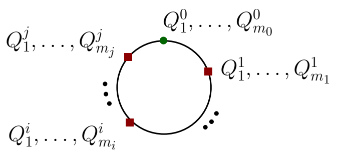

$Q_1^0, \dots, Q_{m_0}^0$
$Q_1^j, \dots, Q_{m_j}^j$
$Q_1^1, \dots, Q_{m_1}^1$
$Q_1^i, \dots, Q_{m_i}^i$

Fig. 12. Unit Circle

Definition 6.1. *Let $\mathcal{T}$ be a reachability problem, D_{max} an upper bound on the numeric values appearing in $\mathcal{T}$ and $\mathcal{S} = \{\ F_1@t_1,\ F_2@t_2, \dots,\ F_n@t_n,\ Time@t\ \}$. Let*

$$\Delta_{\mathcal{S}} = \Big\langle \{P_1^1, \dots, P_{m_1}^1\}, \delta_{1,2}, \{P_1^2, \dots, P_{m_2}^2\}, \dots \{P_1^{j-1}, \dots, P_{m_{j-1}}^{j-1}\}, \delta_{j-1,j}, \{P_1^j, \dots, P_{m_j}^j\} \Big\rangle$$

where $\{P_1^1, \dots, P_{m_1}^1, P_1^2, \dots, P_{m_j}^j\} = \{F_1, \dots, F_n, Time\}$, timestamps of facts $P_1^i, \dots, P_{m_i}^i$ have the same integer part, t^i, $\forall i = 1, \dots, j$, and $\delta_{i,i+1}$ is the truncated time difference (w.r.t. D_{max}) between a fact in class i and a fact in class $i+1$. Let

$$\mathcal{U}_{\mathcal{S}} = [\ \{Q_1^0, \dots, Q_{m_0}^0\}_{\mathcal{Z}}, \{Q_1^1, \dots, Q_{m_1}^1\}, \dots, \{Q_1^k, \dots, Q_{m_k}^k\}\ ]$$

*where $\{Q_1^0, \dots, Q_{m_0}^0, Q_1^1, \dots, Q_{m_k}^k\} = \{F_1, \dots, F_n, Time\}$, timestamps of facts $Q_1^i, \dots, Q_{m_i}^i$ have the same decimal part, $\forall i = 0, \dots, k$, and timestamps of facts $Q_1^0, \dots, Q_{m_0}^0$ are integers. We say that $\Delta_{\mathcal{S}}$ is the δ-*configuration *of $\mathcal{S}$, $\mathcal{U}_{\mathcal{S}}$ is the* unit-circle *of $\mathcal{S}$, and $\mathcal{A}_{\mathcal{S}} = \langle \Delta_{\mathcal{S}}, \mathcal{U}_{\mathcal{S}} \rangle$ is the* circle-configuration *of the configuration $\mathcal{S}$ (or the circle-configuration corresponding to $\mathcal{S}$).*

For example, with $D_{max} = 4$, configuration

$$\mathcal{S}_1 = \{\ M@3.01, R@3.11, P@4.12, Time@11.12, Q@12.58, S@14\ \} \tag{23}$$

has the circle-configuration $\langle \Delta_{\mathcal{S}_1}, \mathcal{U}_{\mathcal{S}_1} \rangle$, where

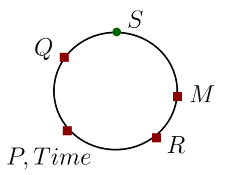
$$\begin{aligned} \Delta_{\mathcal{S}_1} &= \langle\ \{M, R\}1, \{P\}, \infty, \{Time\}, 1, \{Q\}, 2, \{S\}\ \rangle \\ \mathcal{U}_{\mathcal{S}_1} &= [\ \{S\}_{\mathcal{Z}}, \{M\}, \{R\}, \{P, Time\}, \{Q\}\ ]\ , \end{aligned}$$

as depicted in Figure 13 .

For simplicity, we sometimes write $\mathcal{A}$ and $\langle \Delta, \mathcal{U} \rangle$ instead of $\mathcal{A}_{\mathcal{S}}$ and $\langle \Delta_{\mathcal{S}}, \mathcal{U}_{\mathcal{S}} \rangle$, when the corresponding configuration is clear from the context.

Notice that $\delta_{i,i+1}$ is well defined, since all the facts in the same class $P_1^k, \dots, P_{m_k}^k$ have the same integer part, t^k, $\forall k = 1, \dots, j$. Namely,

$$\delta_{i,i+1} = \delta_{Q_1^i, Q_1^{i+1}} = \begin{cases} t^{i+1} - t^i, & \text{if } \ t^{i+1} - t^i \leq D_{max} \\ \infty, & \text{otherwise} \end{cases} \quad , \qquad i = 1, \dots, j-1\ .$$



$\langle \{M,R\}, 1, \{P\}, \infty, \{Time\}, 1, \{Q\}, 2, \{S\} \rangle$

Fig. 13. Circle-Configuration

6.1. Constraint Satisfaction

A circle-configuration $\langle \Delta, \mathcal{U} \rangle$ contains all the information needed in order to determine whether a constraint of the form used in our model, *i.e.*, given in (6), is satisfied or not.

Consider the circle-configuration in Figure 13 which corresponds to configuration $\mathcal{S}_1$ given in (23). To determine, for instance, whether $t_Q > t_{Time} + 1$, we compute the integer difference between t_Q and t_{Time} from the δ-configuration. This turns out to be 1 and means that we need to look at the decimal part of these timestamps to determine whether the constraint is satisfied or not. Since the decimal part of t_Q is greater than the decimal part of t_{Time}, as can be observed in the unit circle, we can conclude that the constraint is satisfied. Similarly, one can also conclude that the constraint $t_Q > t_{Time} + 2$ is not satisfied as $int\,(t_Q) = int\,(t_{Time}) + 1$. The following results formalize this intuition.

Lemma 6.2. *Let $\langle \Delta, \mathcal{U} \rangle$ be a circle-configuration of the configuration $\mathcal{S}$. Then for two arbitrary facts $P@t_p$ and $Q@t_Q$ in $\mathcal{S}$ and a natural number $D < D_{max}$ the following holds:*

- $t_P > t_Q + D$ *iff* $\Delta(Q,P) > D$ *or* $(\ \Delta(Q,P) = D$ *and* $\mathcal{U}(P) > \mathcal{U}(Q)\ )$ *;*
- $t_P > t_Q - D$ *iff* $\Delta(P,Q) < D$ *or* $(\ \Delta(P,Q) = D$ *and* $\mathcal{U}(P) > \mathcal{U}(Q)\ )$ *;*
- $t_P = t_Q + D$ *iff* $\Delta(Q,P) = D$ *and* $\mathcal{U}(Q) = \mathcal{U}(P)$*;*
- $t_P = t_Q - D$ *iff* $\Delta(P,Q) = D$ *and* $\mathcal{U}(Q) = \mathcal{U}(P)$*;*

Proof. Let $t_P \geq t_Q$. Recall that, as per definition of circle-configurations, there are i and j such that $Q = Q_i$ and $P = Q_j$, and that

$$\begin{aligned} \Delta(Q,P) = \Delta(Q_i,Q_j) &= \textstyle\sum \delta_{k,k+1} \\ &= int\,(Q_j) - int\,(Q_{j-1}) + int\,(Q_{j-1}) - int\,(Q_{j-2}) + \cdots - int\,(Q_i) \\ &= int\,(Q_j) - int\,(Q_i) = int\,(P) - int\,(Q) \end{aligned}$$

Then, for a natural number $D \leq D_{max}$ it holds that

$$\begin{aligned} t_P = t_Q + D \ &\text{iff}\ \ int\,(t_P - t_Q) = D \ \text{and}\ dec\,(t_P - t_Q) = 0 \\ &\text{iff}\ \ \Delta(Q,P) = D \ \text{and}\ \mathcal{U}(Q) = \mathcal{U}(P) \end{aligned}$$

because $dec\,(t_P) = dec\,(t_Q)$ implies $int\,(t_P - t_Q) = D$ iff $int\,(t_P) - int\,(t_Q) = D$. Similarly,

$$\begin{aligned} t_P > t_Q + D \ &\text{iff}\ \ int\,(t_P - t_Q) > D \ \text{or}\ (\ int\,(t_P - t_Q) = D \ \text{and}\ dec\,(t_P - t_Q) > 0\ ) \\ &\text{iff}\ \ \Delta(Q,P) > D \ \text{or}\ (\ \Delta(Q,P) = D \ \text{and}\ \mathcal{U}(P) > \mathcal{U}(Q)\ ) \end{aligned}$$

The remaining cases are proven similarly. □

Following the above result we say that a circle-configuration $\langle \Delta, \mathcal{U} \rangle$ corresponding to a configuration $\mathcal{S}$ *satisfies the constraint* c if the configuration $\mathcal{S}$ satisfies the constraint c. We also say that a circle-configuration is a *goal circle-configuration* if it corresponds to a goal configuration. Furthermore, we also say that an *action is applicable to a circle-configuration* if that action is applicable to the corresponding configuration.

6.2. Rewrite Rules and Plans with Circle-Configurations

This section shows that given a reachability problem with a set of rules, $\mathcal{R}$, involving dense time, and an upper bound on the numbers appearing in the problem, D_{max}, we can compile a set of rewrite rules, $\mathcal{A}$, over circle-configurations. Moreover, we show that any plan generated using the rules from $\mathcal{R}$ can be soundly and faithfully represented by a plan using the set of rules $\mathcal{A}$. We first explain how we apply instantaneous rules to circle-configurations and then we explain how to handle the time advancement rule.

Instantaneous Actions

Let D_{max} be an upper bound on the numeric values in the given problem and let the following rule be an instantaneous rule (see Section 3.1) from the set of actions $\mathcal{R}$:

$$\begin{array}{l} Time@T,\ W_1@T_1, \ldots,\ W_k@T_k,\ F_1@T_1', \ldots,\ F_n@T_n' \mid C \longrightarrow \\ \quad \exists\, \vec{X}.\, [\ Time@T,\ W_1@T_1, \ldots,\ W_k@T_k,\ Q_1@(T+D_1), \ldots,\ Q_m@(T+D_m)\ ] \end{array}$$

The above rule is compiled into a sequence of operations that may rewrite a given circle-configuration $\langle \Delta, \mathcal{U} \rangle$ into another circle-configuration $\langle \Delta_1, \mathcal{U}_1 \rangle$ as follows:

(1). Check whether there are occurrences of $W_1, \ldots, W_k$ and $F_1, \ldots, F_n$ in $\langle \Delta, \mathcal{U} \rangle$ such that the guard C is satisfied by $\langle \Delta, \mathcal{U} \rangle$. If it is the case, then continue to the next step; otherwise the rule is not applicable;

(2). We obtain the circle-configuration $\langle \Delta', \mathcal{U}' \rangle$ by removing a single occurrence of each of the facts $F_1, \ldots, F_n$ in $\langle \Delta, \mathcal{U} \rangle$ used in step 1, and recomputing the truncated time differences so that for all the remaining facts P and R in Δ, we have $\Delta'(P,R) = \Delta(P,R)$, *i.e.*, the truncated time difference between P and R is preserved;

(3). Create fresh values, $\vec{e}$, for the existentially quantified variables $\vec{X}$;

(4). We obtain the circle-configuration $\langle \Delta_1, \mathcal{U}_1 \rangle$ by adding the facts $Q_1[\vec{e}/\vec{X}], \ldots, Q_m[\vec{e}/\vec{X}]$ to Δ' so that $\Delta_1(Time, Q_i) = D_i$, for $1 \leq i \leq m$, and that $\Delta_1(P,R) = \Delta'(P,R)$ for all the remaining facts P and R in Δ'. We then obtain $\mathcal{U}_1$ by adding the facts $Q_1, \ldots, Q_m$ to the class of the fact $Time$ in the unit circle $\mathcal{U}'$;

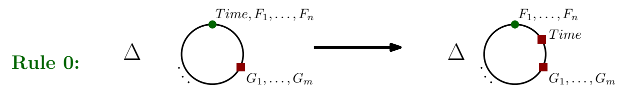
(5). Return the circle-configuration $\langle \Delta_1, \mathcal{U}_1 \rangle$.

The sequence of operations described above has the effect one would expect: replace the facts $F_1, \ldots, F_n$ in the pre-condition of the action with facts $Q_1, \ldots, Q_m$ appearing in the post-condition of the action but taking care to update the truncated time differences in the δ-configuration. Moreover, all steps can be computed in polynomial time.

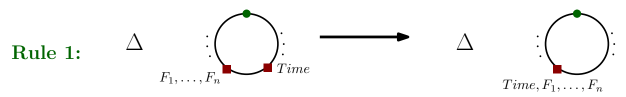
For example, consider the configuration $\mathcal{S}_1$ given in (22) and the rule:

$$Time@T,\ R@T_1,\ P@T_2 \longrightarrow Time@T,\ P@T_2,\ N@(T+2)$$

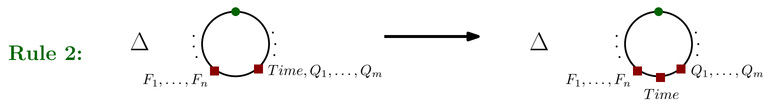
- **Time in the zero point and not in the last class in the unit circle, where $n \geq 0$:**



- **Time alone and not in the zero point nor in the last class in the unit circle:**



- **Time not alone and not in the zero point nor in the last class in the unit circle:**



- **Time not alone and in the last class in the unit circle which may be at the zero point:**

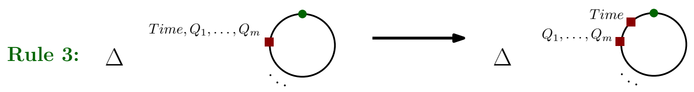
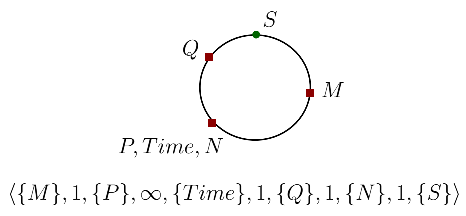


Fig. 14. Rewrite Rules for Time Advancement using Circle-Configurations.

If we apply this rule to $\mathcal{S}_1$, we obtain the configuration

$$\mathcal{S}_2 = \{\, M@3.01,\ P@4.12,\ Time@11.12,\ Q@12.58,\ N@13.12,\ S@14\,\}.$$

On the other hand, if we apply the above steps to the circle-configuration of $\mathcal{S}_1$, shown in Figure 13, we obtain the circle-configuration shown to the right. It is easy to check that this is indeed the circle-configuration of $\mathcal{S}_2$. The truncated time differences are updated and the fact N is added to the class of $Time$ in the unit circle.


S
Q
M
$P, Time, N$
$\langle \{M\}, 1, \{P\}, \infty, \{Time\}, 1, \{Q\}, 1, \{N\}, 1, \{S\} \rangle$

Time Advancement Rule
Specifying the time advancement rule

$$Time@T \longrightarrow Time@(T + \varepsilon)$$

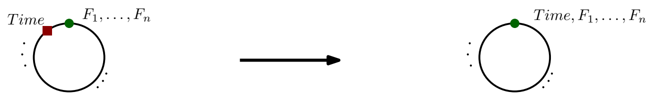
over circle-configurations is more interesting. This action is translated into the rules depicted in Figures 14 and 15. There are eight rules that rewrite a circle-configuration, $\langle \Delta, \mathcal{U} \rangle$, depending on the position of the fact $Time$ in the unit circle $\mathcal{U}$.

Rule 0 specifies the case when the fact $Time$ appears in the zero point of $\mathcal{U}$. Then $\mathcal{U}$ is re-written so that a new class is created immediately after the zero point clockwise before any other class on the unit circle, and $Time$ is moved to that new class. This denotes that the decimal part of $Time$ is greater than zero and less than the decimal part of the facts in the following class $G_1, \dots, G_n$.

Rule 1 specifies the case when the fact $Time$ appears alone in a class on the unit circle and not in the last class. This means that there are some facts, $F_1, \dots, F_n$, that appear in a class immediately after $Time$, *i.e.*, $\mathcal{U}(F_i) > \mathcal{U}(Time)$ and for any other fact G, $G \notin \{F_1, \dots, F_n\}$, such that $\mathcal{U}(G) > \mathcal{U}(Time)$,

- **Time alone and in the last class in unit circle - Case 1:** $m > 0, k \geq 0, n \geq 0$ **and** $\delta_1 > 1$**:**

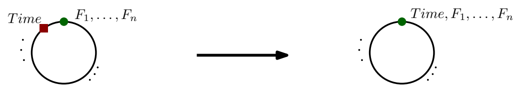
Rule 4:



$\Delta = \langle \ldots, \mathcal{P}_{-1}, \delta_{-1}, \{Time, Q_1, \ldots, Q_m\}, \delta_1, \mathcal{P}_1, \ldots, \mathcal{P}_k \rangle$ $\quad \Delta' = \langle \ldots, \mathcal{P}_{-1}, \delta_{-1}, \{Q_1, \ldots, Q_m\}, 1, \{Time\}, \delta_1 - 1, \mathcal{P}_1, \ldots, \mathcal{P}_k \rangle$

- **Time alone and in the last class in unit circle - Case 2:** $m > 0, k \geq 1$ **and** $n \geq 0$**:**

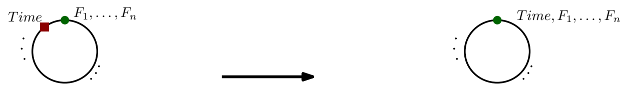
Rule 5:



$\Delta = \langle \ldots, \mathcal{P}_{-1}, \delta_{-1}, \{Time, Q_1, \ldots, Q_m\}, 1, \mathcal{P}_1, \ldots, \mathcal{P}_k \rangle$ $\quad \Delta' = \langle \ldots, \mathcal{P}_{-1}, \delta_{-1}, \{Q_1, \ldots, Q_m\}, 1, \{Time\} \cup \mathcal{P}_1, \ldots, \mathcal{P}_k \rangle$

- **Time alone and in the last class in unit circle - Case 3:** $k \geq 0$ **such that** $\delta_1 > 1$ **when** $k > 0$ **and** γ_{-1} **is the truncated time of** $\delta_{-1} + 1$**:**

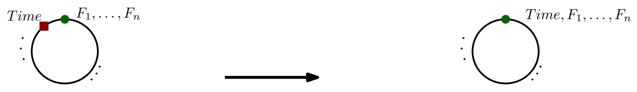
Rule 6:



$\Delta = \langle \ldots, \mathcal{P}_{-1}, \delta_{-1}, \{Time\}, \delta_1, \mathcal{P}_1, \ldots, \mathcal{P}_k \rangle$ $\quad \Delta' = \langle \ldots, \mathcal{P}_{-1}, \gamma_{-1}, \{Time\}, \delta_1 - 1, \mathcal{P}_1, \ldots, \mathcal{P}_k \rangle$

- **Time alone and in the last class in unit circle - Case 4:** $k \geq 1$ **and** γ_{-1} **is the truncated time of** $\delta_{-1} + 1$**:**

Rule 7:



$\Delta = \langle \ldots, \mathcal{P}_{-1}, \delta_{-1}, \{Time\}, 1, \mathcal{P}_1, \ldots, \mathcal{P}_k \rangle$ $\quad \Delta' = \langle \ldots, \mathcal{P}_{-1}, \gamma_{-1}, \{Time\} \cup \mathcal{P}_1, \ldots, \mathcal{P}_k \rangle$

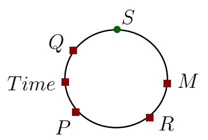
- **Time alone and in the last class in unit circle and in Δ - Case 5:** γ_{-1} **is the truncated time of** $\delta_{-1} + 1$**:**

Fig. 15. (Cont.) Rewrite Rules for Time Advancement using Circle-Configurations.

$\mathcal{U}(G) > \mathcal{U}(F_i)$ holds. In this case, then time can advance so that it ends up in the same class as F_i, *i.e.*, time has advanced so much that its decimal part is the same as the decimal part of the timestamps of $F_1, \ldots, F_n$. Therefore a constraint of the form $T_{F_i} > T_{Time} + D$ that was satisfied by $\langle \Delta, \mathcal{U} \rangle$ might no longer be satisfied by the resulting circle-configuration, depending on D and the δ-configuration Δ.

Rule 2 is similar, but is only applicable when $Time$ is not alone in the unit circle class, *i.e.*, there is at least one fact Q_i such that $\mathcal{U}(Time) = \mathcal{U}(Q_i)$ and this class is not the last one, as in Rule 1. Then, Rule 2 advances time enough so that its decimal part is greater than the decimal part of the timestamps of Q_i, but not greater than the decimal part of the timestamps of the facts in the class that immediately follows on the circle.

For example, Rule 2 could be applied to the circle-configuration $\mathcal{A}_{\mathcal{S}_1}$ shown in Figure 13. We obtain the following circle-configuration, where the δ-configuration does not change, but the fact $Time$ is moved to a new class on the unit circle, obtaining the circle-configuration $\mathcal{A}_{\mathcal{S}_3}$ shown to the right, Figure 16.



$\langle \{M, R\}, 1, \{P\}, \infty, \{Time\}, 1, \{Q\}, 2, \{S\} \rangle$

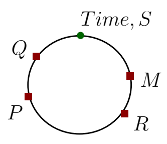
Fig. 16.: circle-configuration $\mathcal{A}_{\mathcal{S}_3}$

Rule 3 is similar to Rule 2, but it is applicable when $Time$ is in the last equivalence class, in which case a new class is created and placed clockwise immediately before the zero point of the circle.

Notice that the δ-configuration is not changed by Rules 0-3. The only rules that change the δ-configuration are the Rules 4, 5, 6 and 7, as in these cases $Time$ advances enough to complete the unit circle, *i.e.*, reach the zero point.

Rules 4 and 5 handle the case when $Time$ initially has the same integer part as timestamps of other facts $Q_1, \ldots, Q_m$, in which case it might create a new class in the δ-configuration (Rule 4) or merge with the following class $\mathcal{P}_1$ (Rule 5). Rules 6 and 7 handle the case when $Time$ does not have the same integer part as the timestamp of any other fact, *i.e.*, it appears alone in Δ, in which case it might still remain alone in the same class (Rule 6) or merge with the following class $\mathcal{P}_1$ (Rule 7). Notice that the time difference, δ_{-1}, to the class, $\mathcal{P}_{-1}$, immediately before the class of $Time$ is incremented by one and truncated by the value of D_{max} if necessary.

For example, consider the circle-configuration $\mathcal{A}_{S_3}$ illustrated in Figure 16. It is easy to check that by applying Rule 1, followed by Rule 3 to $\mathcal{A}_{S_3}$ we obtain a circle-configuration for which the Rule 7 is applicable. After applying Rule 7 we obtain the configuration $\mathcal{A}_{S_4}$ shown to the right.



$\langle \{M, R\}, 1, \{P\}, \infty, \{Time, Q\}, 2, \{S\} \rangle$

Given a reachability problem $\mathcal{T}$ and an upper bound D_{max} on the numeric values of $\mathcal{T}$ with the set of rules $\mathcal{R}$ containing an instantaneous rule r, we write $[r]$ for the corresponding rewrite rule of r over circle-configurations as described above. Moreover, let $Next$ be the set of 8 time advancing rules shown in Figures 14 and 15. Notice that for a given circle-configuration only one of these rules is applicable.

Therefore, we write

$$\widetilde{\mathcal{R}} = \{ \, [r] : r \in \mathcal{R} \, \} \cup Next$$

for the set of rules over circle-configurations corresponding to the set of rules $\mathcal{R}$ over configurations.

We use $\mathcal{A} \longrightarrow_a \mathcal{A}_1$ for the one-step reachability relation between circle-configurations using the rewrite rule a, *i.e.*, the circle-configuration $\mathcal{A}$ may be rewritten to the circle-configuration $\mathcal{A}_1$ using the rewrite rule a. Finally, $\mathcal{A} \longrightarrow^* \mathcal{A}_1$ (respectively, $\mathcal{A} \longrightarrow^*_{\mathcal{R}'} \mathcal{A}_1$) denotes the reflexive transitive closure relation of the one-step relation (respectively, using only rules from the set $\mathcal{R}' \subseteq \widetilde{\mathcal{R}}$).

Lemma 6.3. *Let $\mathcal{T}$ be a reachability problem and D_{max} be an upper bound on the numeric values in $\mathcal{T}$. Let $\mathcal{A}_1$ be the circle-configuration of the configuration $\mathcal{S}_1$, and r be an instantaneous action in $\mathcal{T}$. Then $\mathcal{S}_1 \longrightarrow_r \mathcal{S}_2$ if and only if $\mathcal{A}_1 \longrightarrow_{[r]} \mathcal{A}_2$, where $\mathcal{A}_2$ is the circle-configuration of $\mathcal{S}_2$. Moreover, $\mathcal{S}_1 \longrightarrow_{Tick} \mathcal{S}_2$ if and only if $\mathcal{A}_1 \longrightarrow^*_{Next} \mathcal{A}_2$, where $\mathcal{A}_2$ is the circle-configuration of $\mathcal{S}_2$.*

Proof. Following Lemma 6.2 we only need to prove the claim regarding $Tick$ and $Next$ rules as $\mathcal{S}_1$ satisfies the time constraints of rule r if and only if its circle-configuration $\mathcal{A}_1$ also satisfies the constraints in $[r]$. Moreover $\mathcal{A}_2$ is the circle-configuration of $\mathcal{S}_2$ by construction.

In order to prove that, given a transition $Next$ over circle-configurations, there is the matching pair of configurations and the appropriate instance of $Tick$ rule, we first show how to extract a corresponding concrete configuration $\mathcal{S}$ from a given circle-configuration $\mathcal{A} = \langle \Delta, \mathcal{U} \rangle$, where

$$\begin{aligned} \Delta &= \Big\langle \, \{P_1^1, \ldots, P_{m_1}^1\}, \delta_{1,2}, \{P_1^2, \ldots, P_{m_2}^2\}, \ldots \{P_1^{j-1}, \ldots, P_{m_{j-1}}^{j-1}\}, \delta_{j-1,j}, \{P_1^j, \ldots, P_{m_j}^j\} \, \Big\rangle \, , \\ \mathcal{U} &= [\, \{Q_1^0, \ldots, Q_{m_0}^0\}_{\mathcal{Z}}, \, \{Q_1^1, \ldots, Q_{m_1}^1\}, \ldots, \, \{Q_1^k, \ldots, Q_{m_k}^k\} \,] \, . \end{aligned}$$

Recall that $\{P^1_1, \dots, P^1_{m_1}, P^2_1, \dots, P^j_{m_j}\} = \{F_1, \dots, F_n, Time\}$ so we easily obtain (untimed) facts of the configuration $\mathcal{S}$. We timestamp these facts as follows. From Δ we can extract natural numbers to be assigned as integer values of timestamps of facts: We assign $I_1 = 0$ as the integer part of the timestamp of each of the facts $P^1_1, \dots, P^1_{m_1}$. As I_{i+1}, the integer part of the timestamp of each of the facts $P^{i+1}_1, \dots, P^{i+1}_{m_{i+1}}$, we set

$$I_{i+1} = \begin{cases} I_i + \delta_{i,i+1} \ , & \text{if } \ \delta_{i,i+1} \leq D_{max} \\ I_i + D_{max} + 1 \ , & \text{if } \ \delta_{i,i+1} = \infty \end{cases} \quad , \quad i = 1, \dots, j-1 \ .$$

Then, from $\mathcal{U}$ we can extract values to be assigned as decimal values of timestamps. We set 0 as the decimal part of facts $Q^0_1, \dots, Q^0_{m_0}$, and $\frac{i}{k+1}$ as the decimal part of facts $Q^i_1, \dots, Q^i_{m_i}$, $\forall i = 1, \dots, k$. Obtained timestamped facts form configuration $\mathcal{S}$ corresponding to circle-configuration $\mathcal{A}$.

Let $\mathcal{A}_1 \to_{Next} \mathcal{A}_2$. From $\mathcal{A}_1$ we extract a corresponding concrete configuration $\mathcal{S}_1$ as described above. Recall that *Next* is a set of 8 rules representing time advancement over circle-configurations, as given in Figures 14 and 15. Depending on the position of the fact $Time$ in $\mathcal{A}_1$ only one of the 8 rules from *Next* applies. Assume that $\mathcal{A}$ fits the first case, *i.e.*, *Rule* 0 in Figure 14. Then, any value $\varepsilon \in \langle 0, \frac{1}{k+1} \rangle$, *e.g.*, $\varepsilon = \frac{1}{2(k+1)}$, is suitable for an instance of the $Tick$ rule, $Time@T \longrightarrow Time@(T+\varepsilon)$, in $\mathcal{S}_1 \longrightarrow_{Tick} \mathcal{S}_2$ so that $\mathcal{A}_2$ corresponds to configuration $\mathcal{S}_2$.

In the same way we can take $\varepsilon \in \langle 0, \frac{1}{k+1} \rangle$ in cases corresponding to *Rule* 2 and *Rule* 3 from Figure 14. For the remaining rules, *i.e.*, *Rule* 1 and *Rule* 4 to *Rule* 7, taking $\varepsilon = \frac{1}{k+1}$ results in $\mathcal{A}_2$ that corresponds to configuration $\mathcal{S}_2$. That is because the timestamp of $Time$, say $\frac{j}{k+1}$, is increased exactly so that it matches the decimal part of next group of facts, that is $\frac{j+1}{k+1}$ or 0.

For the opposite direction let $\mathcal{S}_1 \longrightarrow_{Tick} \mathcal{S}_2$ by means of the instance of $Tick$ using the actual value ε, written $\mathcal{S}_1 \longrightarrow_{Tick_\varepsilon} \mathcal{S}_2$. Configurations $\mathcal{S}_1$ and $\mathcal{S}_2$ differ only in the timestamp of the fact $Time$. It may well be that $\mathcal{S}_1$ and $\mathcal{S}_2$ correspond to the same circle-configuration $\mathcal{A}_1 = \mathcal{A}_2$. Then, to the given $Tick$ action, $\mathcal{S}_1 \longrightarrow_{Tick} \mathcal{S}_2$, corresponds the empty sequence of *Next* actions, $\mathcal{A}_1 \longrightarrow^*_{Next} \mathcal{A}_2$. In the rest of the proof we assume that $\mathcal{S}_1$ and $\mathcal{S}_2$ do not correspond to the same circle configuration.

The effect of the $Tick_\varepsilon$ rule is the same as the effect of consecutively applying a series of $Tick$ rules using ε_i where $\varepsilon_1 + \cdots + \varepsilon_n = \varepsilon$. We can choose a sequence of values ε_i so that to the $Tick$ for ε_i, for each i, corresponds a single application of the *Next* rule:

$$\begin{tikzcd}
\mathcal{S}_1 \arrow[r, "Tick_{\varepsilon_1}"'] \arrow[d, leftrightsquigarrow] & \mathcal{S}^1 \arrow[r, "Tick_{\varepsilon_2}"'] \arrow[d, leftrightsquigarrow] & \mathcal{S}^2 \arrow[r, "Tick_{\varepsilon_3}"'] \arrow[d, leftrightsquigarrow] & \cdots \arrow[r, "Tick_{\varepsilon_n}"'] & \mathcal{S}_2 \arrow[d, leftrightsquigarrow] \\
\mathcal{A}_1 \arrow[r, "Next"'] & \mathcal{A}^1 \arrow[r, "Next"'] & \mathcal{A}^2 \arrow[r, "Next"'] & \cdots \arrow[r, "Next"'] & \mathcal{A}_2
\end{tikzcd}$$

Consecutive application of such $Tick_{\varepsilon_i}$ corresponds to a series of the *Next* rules as given in Figures 14 and 15. Namely, each ε_i corresponds to one of the 8 Next rules.

Concrete values of timestamps in $\mathcal{S}$ reflect in the position of the fact $Time$ in $\mathcal{A}_1$. Let the fact $Q@T_Q$ be one of the facts in $\mathcal{S}$ with the smallest difference $dec\,(T_Q) - dec\,(T) > 0$. If the global time T has the decimal part equal to some other fact in the configuration, taking $0 < \varepsilon < dec\,(T_Q) - dec\,(T)$ corresponds to the application of either *Rule* 0, *Rule* 2 or *Rule* 3. Otherwise, if the decimal part of T is different than the decimal part of all other facts in $\mathcal{S}_1$, taking $\varepsilon_i = dec\,(T_Q) - dec\,(T)$ results in the new global time that matches the timestamp T_Q in its decimal part. □

Theorem 6.4. *Let $\mathcal{T}$ be a reachability problem, D_{max} be an upper bound on the numeric values in $\mathcal{T}$. Then $\mathcal{S}_I \longrightarrow^* \mathcal{S}_G$ for some initial and goal configurations $\mathcal{S}_I$ and $\mathcal{S}_G$ in $\mathcal{T}$ if and only if $\mathcal{A}_I \longrightarrow^* \mathcal{A}_G$ where $\mathcal{A}_I$ and $\mathcal{A}_G$ are the circle-configurations of $\mathcal{S}_I$ and $\mathcal{S}_G$, respectively.*

Proof. The proof is by induction on the length of the given plan. Additionally, as in the proof of Lemma 6.3, we can assume that each instance of the time advancement rule in the plan over configurations uses a concrete value ε that corresponds to a single application of one of the *Next* rules in the matching plan over circle-configurations, or to no action at all (*i.e.*, to an empty sequence of *Next* actions). We get the following bisimulation:

$$\begin{tikzcd}
\mathcal{S}_I \arrow[r, "r_1"'] \arrow[d, leftrightsquigarrow] & \cdots \arrow[r, "r_{i-1}"'] & \mathcal{S}_{i-1} \arrow[r, "r_i"'] \arrow[d, leftrightsquigarrow] & \mathcal{S}_i \arrow[r, "r_{i+1}"'] \arrow[d, leftrightsquigarrow] & \cdots \arrow[r, "r_n"'] & \mathcal{S}_G \arrow[d, leftrightsquigarrow] \\
\mathcal{A}_I \arrow[r, "r'_1"'] & \cdots \arrow[r, "r'_{i-1}"'] & \mathcal{A}_{i-1} \arrow[r, "r'_i"'] & \mathcal{A}_i \arrow[r, "r'_{i+1}"'] & \cdots \arrow[r, "r'_n"'] & \mathcal{A}_G
\end{tikzcd}$$

where r'_i is either the instantaneous action $[r_i]$ over circle-configurations, one of the *Next* rule as given in Figures 14 and 15, or an empty action (in which case $\mathcal{A}_{i-1} = \mathcal{A}_i$). □

This theorem establishes that the set of plans over circle-configurations is a sound and complete representation of the set of plans with dense time. This means that we can search for solutions of problems symbolically, that is, without writing down the explicit values of the timestamps, *i.e.*, the real numbers, in a plan.

7. Complexity Results

This section details some of the complexity results for the reachability problem.

Reachability problem is a rather general problem that can have various applications. For example, the secrecy problem from the field of protocol security can be considered as an instance of the reachability problem. Namely, by interacting with the protocol run, the goal of the intruder is to learn a secret that is initially only known to another participant of the protocol.

In our previous work we have already considered such an application of our formal models in protocol security. In [20,21] we consider bounded memory protocol theories and intruder theories and the corresponding secrecy problem. These formalizations bound the number of concurrent protocol sessions. However, the total number of protocol sessions and the total number of fresh values created is unbounded.

The model introduced in this paper can similarly be applied in the protocol security analysis. As in [20,21], when using only balanced actions, one can necessarily consider only a bounded number of *concurrent* sessions. Nevertheless, the analysis would consider an unbounded number of protocol sessions in *total*.

In addition, by containing the dimension of time, our model presented in this paper can be used for the analysis of time-sensitive protocols, such as cyber-physical security protocols, and their properties.

Conditions for Decidability

From the literature, we can infer some conditions for decidability of the reachability problem in general:

(1). *Upper Bound on the Size of Facts*: In general, if we do not assume an upper bound on the size of facts appearing in a plan, where the size of facts is the total number of predicate, function, constant and variable symbols it contains (*e.g.*, the size of $P(f(a), x, a)$ is 5), then it is easy to encode the Post-Correspondence problem which is undecidable, see [7,15].[8] Thus we will assume an upper bound on the size of facts, denoted by the symbol k.

(2). *Balanced Actions*: An action is balanced if its pre-condition has the same number of facts as its post-condition [27]. The reachability problem is undecidable for (un-timed) systems with possibly unbalanced actions even if the size of facts is bounded [7,15]. In a balanced system, on the other hand, the number of facts in any configuration in a plan is the same as the number of facts of the initial configuration, allowing one to recover decidability under some additional conditions. We denote the number of facts in the configuration by the symbol m.

Both undecidability results related to (un)balanced actions as well as the upper bound on the size of facts are time irrelevant, they carry over to systems with dense time.

(3). *Conditions for Timestamps and Time Constraints*: Recall the form of instantaneous rules in our systems (Equation (5) in Section 3) and the conditions the form of the rules imposes on timestamps and time constraints.

In particular, the form of instantaneous rules imposes the restriction on using only natural numbers (for Ds and D_is) in time constraints and timestamps of created facts. This is not as restrictive as it may appear. Namely, we are able to capture systems that would allow rational numbers as well. These numbers are constants in any given concrete model, and they are used to specify concrete time requirements. By allowing rational numbers we would not add to expressivity of the system, since these constants can be multiplied with a common multiple of their denominators, to obtain natural numbers. The only difference between the original model (with rationals) and the obtained model (with only natural numbers) is the intended denotation of the time units used in the model. For example, one could use minutes instead of days or hours. The intended semantics of the system represented would not change. In fact, to maintain all our results valid, it suffices to assume that all these numerical constants mentioned within the above constraints and timestamps are commensurable, that is, these constants can be given in terms of a common unit.

Furthermore, the form of instantaneous rules restricts the form of timestamps of created facts and the form of time constraints. Timestamps are necessarily of the form $T + D$, where T is the current time of the enabling configuration and D a natural number, and time constraints are relative, *i.e.*, involve exactly two time variables from the pre-condition of the rule. In [25] we have shown that relaxing either of the form of timestamps or the form of time constraints leads to the undecidability of the reachability problem for multiset rewriting models with discrete time. The same would thus lead to the undecidability of the reachability problems considered in this paper. In particular, we get undecidability for systems with non-relative time constraints that involve three or more time variables. Similarly, we fall into undecidabilty if we allow timestamps of the created facts to be (already linear) polynomials of time variables from the pre-condition of the rule. See [25] for more details.

Corollary 7.1. *The reachability problem for our model is undecidable in general.*

[8]We leave for Future Work the investigation of specific cases, *e.g.*, protocol with tagging mechanisms, where this upper bound may be lifted [39].

Proof. To an instance of a reachability problem for (un-timed) systems with possibly unbalanced actions, we associate a timed version of the same problem as follows.

We attach arbitrary timestamps to facts in the initial configuration. Similarly, to each (untimed) fact in the goal configuration we attach a different time variable. We put no constraints to the initial and goal configurations. This way, timestamps of facts in the initial and the goal configuration play no particular role in the reachability problem.

Finally, we consider the following set of instantaneous rules: For every rule

$$W_1 \ldots,\ W_k,\ F_1, \ldots,\ F_n \ \longrightarrow\ \exists\, \vec{X}.\, [\ W_1, \ldots,\ W_k,\ Q_1, \ldots,\ Q_m\ ]$$

in the un-timed system we take the corresponding rule

$$\begin{array}{l} Time@T,\ W_1@T_1, \ldots,\ W_k@T_k,\ F_1@T_1', \ldots,\ F_n@T_n' \ \longrightarrow \\ \quad \exists\, \vec{X}.\, [\ Time@T,\ W_1@T_1, \ldots,\ W_k@T_k,\ Q_1@T, \ldots,\ Q_m@T\ ] \end{array}$$

Notice that above rules have no guards attached and since facts have arbitrary timestamps, the application of rules in the timed system is independent from the time aspect.

Clearly, the original (untimed) reachability problem has a solution if and only if its associated timed version (as described above) has a solution. Since the reachability problem for (untimed) systems with possibly unbalanced actions is undecidable, the reachability problem for timed systems with possibly unbalanced actions is undecidable as well. □

PSPACE-Completeness We show that the reachability problem for our model with dense time and balanced actions is PSPACE-complete. Interestingly, the same problem is also PSPACE-complete when using models with discrete time [24].

Given the machinery in Section 6, we can re-use many results in the literature to show that the reachability problem is also PSPACE-complete for balanced systems with dense time that can create fresh values, as given in Section 3, assuming an upper bound on the size of facts. For instance, we use the machinery detailed in [20] to handle the fact that a plan may contain an unbounded number of fresh values.

The PSPACE lower bound can be inferred from [20]. The interesting bit is to show PSPACE membership of the reachability problem. The following lemma establishes an upper bound on the number of different circle-configurations:

Lemma 7.2. *Given a reachability problem $\mathcal{T}$ under a finite alphabet Σ, an upper bound on the size of facts, k, and an upper bound, D_{max}, on the numeric values appearing in $\mathcal{T}$, then the number of different circle-configurations, denoted by $L_{\mathcal{T}}(m, k, D_{max})$, with m facts (counting repetitions) is*

$$L_{\mathcal{T}}(m, k, D_{max}) \leq J^m (D + 2mk)^{mk} m^m (D_{max} + 2)^{(m-1)},$$

where J and D are, respectively, the number of predicate and the number of constant and function symbols in Σ.

Proof. A circle-configuration consists of a δ-configuration Δ:

$$\Delta = \Big\langle\ \{Q_1^1, \ldots, Q_{m_1}^1\}, \delta_{1,2},\ \{Q_1^2, \ldots, Q_{m_2}^2\}, \ldots,\ \delta_{j-1,j},\ \{Q_1^j, \ldots, Q_{m_j}^j\}\ \Big\rangle$$

and unit circle $\mathcal{U}$:

$$\mathcal{U} = [\ \{Q^0_1, \ldots, Q^0_{m_0}\}_{\mathcal{Z}},\ \{Q^1_1, \ldots, Q^1_{m_1}\}, \ldots,\ \{Q^j_1, \ldots, Q^j_{m_j}\}\ ]\ .$$

In each component, Δ and $\mathcal{U}$, there are m facts, therefore there are m slots for predicate names and at most mk slots for constants and function symbols. Constants can be either constants in the initial alphabet Σ or names for fresh values (nonces). Following [20] and Definition 3.6, we need to consider only $2mk$ names for fresh values (nonces). Finally, only time differences up to D_{max} have to be considered together with the symbol ∞, and there are at most $m-1$ slots for time differences $\delta_{i,j}$ in Δ.

Finally, for each δ-configuration, there are at most m^m unit circles as for each fact F we can assign a class, $\mathcal{U}(F)$, and there are at most m classes. □

Intuitively, our upper bound algorithm keeps track of the length of the plan it is constructing and if its length exceeds $L_{\mathcal{T}}(m, k, D_{max})$, then it knows that it has reached the same circle-configuration twice. Hence, there is a loop in the plan which can be avoided. If there is a solution plan of length that exceeds $L_{\mathcal{T}}(m, k, D_{max})$, there is also a shorter plan that is the solution to the same reachability problem. Therefore, we can nondeterministically search for plans of bounded length. This search is possible in PSPACE since the above number, when stored in binary, occupies only polynomial space with respect to its parameters.

For the given reachability problem $\mathcal{T}$, where m is the number of facts in the initial configuration and k an upper bound on the size of facts, assume functions $\mathcal{G}$ and $\mathcal{R}$ that return the value 1 in Turing space bounded by a polynomial in m, k and $log_2(D_{max})$ when given as input, respectively, a circle-configuration that is a goal circle-configuration, and a pair of a circle-configuration and a transition that is valid, *i.e.*, an instance of an action in $\mathcal{T}$ that is applicable to the given circle-configuration, and return 0 otherwise.

Theorem 7.3. *Let $\mathcal{T}$ be a reachability problem with balanced actions, $\mathcal{S}_0$ be an initial configuration with exactly m facts, D_{max} be an upper bound on the numeric values appearing in $\mathcal{T}$, k an upper bound on the size of facts and let $\mathcal{G}$ and $\mathcal{R}$ be functions as described above. Then there is an algorithm that given an initial configuration $\mathcal{S}_0$ decides the problem $\mathcal{T}$ and the algorithm runs in polynomial space with respect to m, k and $log_2(D_{max})$.*

Proof. Let m be the number of facts in the initial configuration $\mathcal{S}_I$, k an upper bound on the size of facts, D_{max} a natural number that is an upper bound on the numeric values appearing in the reachability problem $\mathcal{T}$, that is in the timestamps of the initial configuration, or the Ds and D_is in constraints or actions of the given reachability problem.

We modify the algorithms given in [20] and in [25] in order to accommodate explicit dense time. The algorithm must accept whenever there is a sequence of actions from $\mathcal{T}$ leading from the initial configuration $\mathcal{S}_I$ to a goal configuration. In order to do so, we construct an algorithm that searches non-deterministically whether such a plan exists. Then we apply Savitch's Theorem to determinize this algorithm.

A crucial point here is that instead of searching for a plan using concrete values, we rely on the equivalence among configurations given in Definition 3.6 and use circle-configurations only. Theorem 6.4 guarantees that this abstraction is sound and faithful.

Let i be a natural number such that $0 \leq i \leq L_T(m, k, D_{max})$.

The algorithm begins with $\mathcal{A}_0$ set to be the circle-configuration of $\mathcal{S}_I$ and iterates the following sequence of operations:

(1). If $\mathcal{A}_i$ is representing a goal configuration, *i.e.*, if $\mathcal{G}(\mathcal{A}_i) = 1$, then return ACCEPT; otherwise continue;
(2). If $i > L_T(m, k, D_{max})$, then FAIL; else continue;
(3). Guess non-deterministically an action, r, from $\mathcal{T}$ applicable to $\mathcal{A}_i$, *i.e.*, such an action r that $\mathcal{R}(\mathcal{A}_i, r) = 1$. If no such action exists, then return FAIL. Otherwise replace $\mathcal{A}_i$ with the circle-configuration $\mathcal{A}_{i+1}$ resulting from applying the action r to the circle-configuration $\mathcal{A}_i$.
(4). Set i = i + 1.

We now show that this algorithm runs in polynomial space. We start with the step-counter i: The greatest number reached by this counter is $L_T(m, k, D_{max})$. When stored in binary encoding, this number only takes space that is polynomial in the given inputs:

$$\log(L_T(m, k, D_{max})) \leq m \log(J) + mk \log(D + 2mk) + m \log m + (m-1) \log(D_{max} + 2).$$

Therefore, one only needs polynomial space to store the values of the step-counter.

We must also be careful to check that any circle-configuration, $W_i = \langle \Delta, \mathcal{U} \rangle$, where

$$\begin{array}{l} \Delta = \left\langle \{Q_1^1, \ldots, Q_{m_1}^1\},\ \delta_{1,2},\ \{Q_1^2, \ldots, Q_{m_2}^2\}, \ldots,\ \delta_{k-1,k},\ \{Q_1^k, \ldots, Q_{m_k}^k\} \right\rangle\ , \\ \mathcal{U} = [\ \{Q_1^0, \ldots, Q_{m_0}^0\}_{\mathcal{Z}},\ \{Q_1^1, \ldots, Q_{m_1}^1\}, \ldots,\ \{Q_1^j, \ldots, Q_{m_j}^j\}\ ] \end{array}$$

can be stored in space that is polynomial to the given inputs. Since our system is balanced, the size of facts is bounded, and the values of the truncated time differences, $\delta_{i,j}$, are bounded, it follows that the size of any circle-configuration $\langle \Delta, \mathcal{U} \rangle$ in a plan is polynomially bounded with respect to m, k and $\log(D_{max} + 2)$.

Finally, the algorithm needs to keep track of the action r guessed when moving from one circle-configuration to another, and for the scheduling of a plan. It has to store the action that has been used at the i^{th} step. Since any action can be stored by remembering two circle-configurations, one can also store these actions in space polynomial to the inputs. □

8. Related and Future Work

The formalization of timed models and their use in the analysis of cyber-physical security protocols has already been investigated. We review this literature.

Meadows *et al.* [32] and Pavlovic and Meadows in [38] propose and use a logic called Protocol Derivation Logic (PDL) to formalize and prove the safety of a number of cyber-physical protocols. In particular, they specify the assumptions and protocol executions in the form of axioms, specifying the allowed order of events that can happen, and show that safety properties are implied by the axiomatization used. They do not formalize an intruder model. Another difference from our work is that their PDL specification is not an executable specification, while we have implemented our specification in Maude [11]. Finally, they do not investigate the complexity of protocol analysis nor investigate the expressiveness of formalizations using discrete and continuous time.

Another approach similar to [32] in the sense that it uses a theorem proving approach is given by Schaller *et al.* [3]. They formalize an intruder model and some cyber-physical security protocols in Isabelle. They then prove the correctness of these protocols under some specific conditions and also identify attacks when some conditions are not satisfied. Their work was a source of inspiration for our intruder model specified in [23], which uses the model described in Section 3. Although their model includes time, their model is not refined enough to capture the attack in-between-ticks as they do not consider the discrete behavior of the verifier.

Boureanu *et al.* [4] proposed a discrete time model for formalizing distance bounding protocols and their security requirements. Thus they are more interested in the computational soundness of distance bounding protocols by considering an adversary model based on probabilistic Turing machines. They claim that their SKI protocol is secure against a number of attacks. However, their time model is discrete where all players are running at the same clock rate. Therefore, their model is not able to capture attacks that exploit the fact that players might run at different speeds.

Pavlovic and Meadows [37] construct a probabilistic model for analysing distance bounding protocols against guessing attacks. The nature of the attack we consider and the attack considered in [37] are quite different leading to different probabilistic models with different goals.

The Timed Automata [2] (TA) literature contains models for cyber-physical protocol analysis. Corin *et al.* [12] formalize protocols and the standard Dolev-Yao intruder as timed automata and demonstrate that these can be used for the analysis. They are able to formalize the generation of nonces by using timed automata, but they need to assume that there is a bound on the number of nonces. This means that they assume a bound on the total number of protocol sessions. Our model based on rewrite theory, on the other hand, allows for an unbounded number of nonces, even in the case of balanced theories [20]. Also they do not investigate the complexity of the analysis problems nor the expressiveness difference between models with discrete and continuous time. Lanotte *et al.* [29] specify cyber-physical protocols, but protocols where messages can be re-transmitted or alternatively a protocol session can be terminated, *i.e.*, timeouts, in case a long time time elapses. They formalize the standard Dolev-Yao intruder. Finally, they also obtain a decidability result for their formalism and an EXPSPACE-hard lower bound for the reachability problem. It seems possible to specify features like timeouts and message re-transmission, in our rewriting formalism.

We also point out some important differences between our PSPACE-completeness proof and PSPACE-completeness proof for timed automata [2]. A more detailed account can be found in the Related Work section of [25]. The first difference is that we do not impose any bounds on the number of nonces created, while the TA proof normally assumes a bound. The second difference is due to the first-order nature of rewrite rules. The encoding of a first-order system in TA leads to an exponential blow-up on the number of states of the automata as one needs take into account all instantiations of rules. Finally, the main abstractions that we use, namely circle-configurations, are one-dimensional, while regions used in the TA PSPACE proof are multidimensional.

Malladi *et al.* [31] formalize distance bounding protocols in strand spaces. They then construct an automated tool for protocol analysis using a constraint solver. They did not take into account the fact that the verifier is running a clock in their analysis and therefore are not able to detect the attack in-between-ticks.

Cheval and Cortier [8] propose a way to prove the properties based on the observational equivalence of processes taking account the time processes execute with time by reducing this problem to the observational equivalence based on the length of inputs. They are able to automatically show that RFID protocols used by passports suffer a privacy attack. While the problems considered are different, it seems possible to use our machinery to prove complexity results on problems based on observational equivalence. We leave this as future work.

Nigam *et al.* [34] investigated further the properties of timed intruder models formalized in a language similar to our timed MSR framework. It is possible to infer an upper-bound on the number of time intruders to consider for cyber-physical security protocol verification. Based on this result, they implemented in Maude, also using SMT solvers, a prover that automates such a verification of protocol security including the attack in-between-ticks introduced here.

Finally, [13] introduces a taxonomy of attacks on distance bounding protocols, which include a new attack called Distance Hijacking Attack. This attack was caused by failures not in the time challenges phase of distance bounding protocols, but rather in the authentication phases. It would be interesting to understand how these attacks can be combined with the attack in-between-ticks to build more powerful attacks.

As future work, we also intend to investigate the challenge-response approach in providing security of distance bounding protocols by expanding our probabilistic analysis to such a scenario. Repeating a number of challenge and response rounds in distance bounding protocols is generally believed to mitigate the chances of an attack occurring. In particular, we are planning to analyze whether the effects of the attack in-between-ticks can be reduced by repeated challenge-response rounds of protocols. Moreover, we are investigating improvements on our implementation in order to check for a wider number of attacks automatically.

Acknowledgments: Nigam is supported by the Brazilian Research Agencies CNPq and Capes. Talcott is partially supported by NSF grant CNS-1318848, by a grant from ONR and by a grant from Capes Science without Borders. Scedrov is supported in part by the AFOSR MURI "Science of Cyber Security: Modeling, Composition, and Measurement". Additional support for Scedrov from ONR. Part of the work was done while Kanovich and Scedrov were visiting the National Research University Higher School of Economics, Moscow. They would like to thank Sergei O. Kuznetsov for providing a very pleasant environment for work. This article was prepared within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) and supported within the framework of a subsidy by the Russian Academic Excellence Project '5-100'. We would like to thank Robin Pemantle for discussions, suggestions, and advice regarding the probabilistic analysis.

References

[1] R. Alur. Principles of Cyber-Physical Systems. *MIT Press*, 2015.

[2] R. Alur and P. Madhusudan. Decision problems for timed automata: A survey. In *SFM*, pages 1–24, 2004.

[3] D. A. Basin, S. Capkun, P. Schaller, and B. Schmidt. Formal reasoning about physical properties of security protocols. *ACM Trans. Inf. Syst. Secur.*, 14(2):16, 2011.

[4] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Practical & provably secure distance-bounding. *IACR Cryptology ePrint Archive*, 2013:465, 2013.

[5] S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In *EUROCRYPT*, pages 344–359, 1993.

[6] S. Capkun and J.-P. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):221–232, 2006.

[7] 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.

[8] V. Cheval and V. Cortier. Timing attacks in security protocols: Symbolic framework and proof techniques. In *Principles of Security and Trust - 4th International Conference, POST 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings*, pages 280–299, 2015.

[9] T. Chothia, F. D. Garcia, J. de Ruiter, J. van den Breekel, and M. Thompson. Relay cost bounding for contactless emv payments. In *Financial Cryptography and Data Security*, 2015.

[10] T. Chothia and V. Smirnov. A traceability attack against e-passports. In *Financial Cryptography and Data Security*, pages 20–34, 2010.

[11] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. MartÁ́ś-Oliet, J. Meseguer, and C. Talcott. *All About Maude: A High-Performance Logical Framework*, volume 4350 of *LNCS*. 2007.

[12] R. Corin, S. Etalle, P. H. Hartel, and A. Mader. Timed analysis of security protocols. *J. Comput. Secur.*, 15(6):619–645, dec 2007.

[13] C. J. F. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun. Distance hijacking attacks on distance bounding protocols. In *SP*, 2012.
[14] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
[15] 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.
[16] H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.
[17] S. Escobar, C. Meadows, J. Meseguer Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties. *Foundations of Security Analysis and Design V*, 2009.
[18] S. Ganeriwal, C. Pöpper, S. Capkun, and M. B. Srivastava. Secure time synchronization in sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(4), 2008.
[19] G. Hancke, and M. Kuhn, An RFID distance bounding protocol. First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05), 67–73, 2005.
[20] M. Kanovich, T. Ban Kirigin, V. Nigam, and A. Scedrov. Bounded memory Dolev-Yao adversaries in collaborative systems. *Inf. Comput.*, 2014.
[21] M. I. Kanovich, T. Ban Kirigin, V. Nigam, and A. Scedrov. Bounded memory protocols. *Computer Languages, Systems & Structures*, pages 40(3-4):137–154, 2014.
[22] M. I. Kanovich, T. Ban Kirigin, V. Nigam, and A. Scedrov. Bounded memory protocols and progressing collaborative systems. In *ESORICS*, pages 309–326, 2013.
[23] M. I. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, and C. L. Talcott. Towards timed models for cyber-physical security protocols. Available on Nigam's homepage, 2014.
[24] M. I. 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.
[25] M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, C. Talcott, R. Perovic. A Rewriting Framework and Logic for Activities Subject to Regulations, Mathematical Structures in Computer Science, 2015, 44 pages, Published online 02 June 2015. http://journals.cambridge.org/article_S096012951500016X
[26] M. Kanovich, T. Ban Kirigin, V. Nigam, A. Scedrov, C. Talcott. Discrete vs. Dense Times in the Analysis of Cyber-Physical Security Protocols. 4th Conference on Principles of Security and Trust (POST), London, UK, April 2015. Springer LNCS, Volume 9036, Springer-Verlag, pages 259 - 279, 2015.
[27] M. I. Kanovich, P. Rowe, and A. Scedrov. Collaborative planning with confidentiality. *J. Autom. Reasoning*, 46(3-4):389–421, 2011.
[28] M. I. Kanovich, P. Rowe, and A. Scedrov. Policy Compliance in Collaborative Systems. In *CSF '09: Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium*, pages 218-233, 2009.
[29] R. Lanotte, A. Maggiolo-Schettini, and A. Troina. Reachability results for timed automata with unbounded data structures. *Acta Inf.*, 47(5-6):279–311, 2010.
[30] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *TACAS*, pages 147–166, 1996.
[31] S. Malladi, B. Bruhadeshwar, and K. Kothapalli. Automatic analysis of distance bounding protocols. *CoRR*, abs/1003.5383, 2010.
[32] C. Meadows, R. Poovendran, D. Pavlovic, L. Chang, and P. F. Syverson. Distance bounding protocols: Authentication logic analysis and collusion attacks. In *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*, pages 279–298. 2007.
[33] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.
[34] Vivek Nigam and Carolyn L. Talcott and Abraão Aires Urquiza Towards the Automated Verification of Cyber-Physical Security Protocols: Bounding the Number of Timed Intruders In *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Proceedings, Part II*, 450–470, 2016
[35] P. C. Ölveczky and J. Meseguer. Semantics and pragmatics of Real-Time Maude. In *Higher-Order and Symbolic Computation*, 20 (1-2) 161–196, 2007.
[36] P. C. Ölveczky and J. Meseguer. Abstraction and Completeness for Real-Time Maude. Electr. Notes Theor. Comput. Sci. volume 176, number 4, 5–27, 2007
[37] D. Pavlovic and C. Meadows. Bayesian Authentication: Quantifying Security of the Hancke-Kuhn Protocol. In *Electr. Notes Theor. Comput. Sci.*, pages 97–122, 2010.
[38] D. Pavlovic and C. Meadows. Deriving ephemeral authentication using channel axioms. In *Security Protocols Workshop*, pages 240–261, 2009.
[39] R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable with unbounded nonces as well. In *FST TCS*, 2003.
[40] K. Ravi, G.H.Varun, and P. T.Vamsi. Rfid based security system. *International Journal of Innovative Technology and Exploring Engineering*, 2, 2013.

[41] V. Shmatikov and M.-H. Wang. Secure verification of location claims with simultaneous distance modification. In *ASIAN*, pages 181–195, 2007.

[42] K. Sun, P. Ning, and C. Wang. Tinysersync: secure and resilient time synchronization in wireless sensor networks. In *CCS*, pages 264–277, 2006.

[43] N. O. Tippenhauer and S. Capkun. Id-based secure distance bounding and localization. In *ESORICS*, pages 621–636, 2009.