

# Reachability Results for Timed Automata with Unbounded Data Structures

Ruggero Lanotte ·  
Andrea Maggiolo-Schettini ·  
Angelo Troina

Received: date / Accepted: date

**Abstract** Systems of Data Management Timed Automata (SDMTAs) are networks of communicating timed automata with structures to store messages and functions to manipulate them. We prove the decidability of the reachability problem for a subclass of SDMTAs which assumes an unbounded knowledge, and we analyze the expressiveness of the model and the considered subclass. In particular, while SDMTAs can simulate a Turing machine, and hence the reachability problem is in general undecidable, the subclass for which reachability is decidable, when endowed with a concept of recognized language, accepts languages that are not regular. As an application, we model and analyze a variation of the Yahalom protocol.

**Keywords** Timed Automata · Vector Addition Systems · Unbounded Data Structures

## 1 Introduction

Automata based formalisms have been successfully used to describe concurrent and distributed systems and analyze their properties. Basic formalisms allow describing the nondeterministic behaviour of a system. This is enough to model the functionality of systems, and hence to capture the qualitative behaviour of a system, but if one wants to capture also time-dependent behaviour, formalisms must be extended with real-time features. Alur and Dill have proposed Timed Automata [4]. In general, timed automata models have an infinite state space. However the region automaton construction shows

---

Ruggero Lanotte  
Dipartimento di Informatica e Comunicazione, Università dell'Insubria  
Via Carloni 78, 22100 Como, Italy  
E-mail: ruggero.lanotte@uninsubria.it

Andrea Maggiolo-Schettini  
Dipartimento di Informatica, Università di Pisa  
Largo B. Pontecorvo 3, 56127 Pisa, Italy  
E-mail: maggiolo@di.unipi.it

Angelo Troina  
Dipartimento di Informatica, Università di Torino  
Corso Svizzera 185, 10149 Torino, Italy  
E-mail: troina@di.unito.it

that this infinite state space can be mapped to an automaton with a finite number of equivalence classes (regions) as states, and finite-state model checking techniques can be applied to the reduced finite region automaton. Among the model checkers for timed automata we quote Kronos [25] and UPPAAL [5]. Systems of communicating agents can be described by automata composed in parallel and sharing synchronization channels. Transitions labelled with complementary channel names can be taken at the same moment and data transmission is typically modelled by a synchronization, where global variables are updated [10].

In this paper we define Systems of Data Management Timed Automata (SDMTAs). An SDMTA has a finite set of elementary messages, a finite set of functions to elaborate them, and a finite set of Data Management Timed Automata (DMTAs). Each DMTA has a finite set of channel labels, a finite set of clocks, a finite set of states (one of them is the initial state), and a finite set of transitions. Transitions from state to state of a DMTA represent either an internal move with the computation of a term (which enriches the knowledge of the automaton) or the input or the output of a term on a channel. The execution of a transition is conditioned by the fulfillment of a constraint. Two DMTAs of a system may perform a communication step modeling the communication of a term through a channel. Time elapsing is modelled by a time step of the automata of the system. Both the term domain (which may include any natural value) and the store of a DMTA are unbounded.

We show that SDMTAs can simulate Turing machines, hence the reachability problem for SDMTAs is, in general, undecidable. However, we consider a subclass of SDMTAs called 1-SDMTAs (where only one integer variable can occur) for which we are able to show that the reachability problem is decidable. Since we cannot simulate unbounded knowledge with a finite state machine, we prove the decidability result by reducing 1-SDMTAs to Vector Addition Systems (VASs). As we are dealing with a dense time model and exponential reductions to VASs, large 1-SDMTAs we will hardly be practically amenable for automated verification techniques. Moreover, we prove that our formalism extends the formalism of [21] in order to deal with a general class of distributed communicating systems with data structures. Actually, the formalism of [21] assumes bounded knowledge and has the power of regular languages, while the decidable class of 1-SDMTAs assumes an unbounded knowledge and, when endowed with a concept of recognized language, accepts languages that are not regular.

With the purpose of modeling general systems, we consider the most general time model, hence the dense time model. The results in this paper hold also if one considers a discrete-time model. Actually, the undecidability result does not depend on the choice of the time model. Moreover, decidability results for a dense-time model imply decidability of the discrete time model. Finally, the problem still has the same complexity, since changing from dense to discrete time does not simplify the construction.

Our purpose is to lay the foundations of a general model where features of time and information storage are considered. The applications of SDMTAs in the modelling and verification of realistic systems can be applied in any context where it is important to keep track of the timing behaviour of a system or of the information it may store. The first field that comes to mind is that of security analysis, where the information circulating on the network and the knowledge of the adversary should be modelled.

Actually, time aspects can influence the flow of messages during the execution of a protocol. For instance, if a message does not arrive in a certain time interval, retransmissions or other behaviour should be considered and, in this case, the protocol specification should model these implementation details. Time information can also be

used within a protocol in order to enrich the information contained in a message (for example by constructing *timestamps*). Finally, the timing of the message flow may be exploited by an adversary to violate the security of the protocol.

As an example we model and analyze with SDMTAs a version of the well-known Yahalom protocol adapted to take timeouts and retransmissions into account.

## 1.1 Related Works

A preliminary version of this paper appeared in [19].

Among the frameworks which aim at describing timed systems with data structures, we may cite the following ones.

In [21] the authors propose a model of communicating automata tailored for describing and analyzing protocols with timing constraints and verifying their security. Each participant in the protocol is described by a state transition diagram where transitions are labelled by events which represent the sending of a structured message along a channel. The performance of a transition is conditioned by the trigger of a communication event and by the temporal constraints imposed by a delay and/or timeout. Communication is synchronous. Primitive messages (public/private keys, identities, nonces, etc.) can be composed by using cryptographic primitives (encryption, hashing, signature, etc.). An initial evidence function assigns each initial state the set of evidence that is known by each participant at the beginning. Such evidence can be augmented by receiving messages from other participants and is reset to the initial conditions every time an input state is reached. The semantics of these descriptions is given in terms of Timed Automata, thus obtaining a specification on which properties can be verified.

Our formalism extends the formalism of [21] to deal with a general class of distributed communicating systems with data structures. Note that the formalism of [21] assumes a bounded memory and has the power of regular languages, while our formalism assumes an unbound memory and, when endowed with a concept of recognized language, does accept languages which are not regular. Differently with respect to [21], we define a direct operational semantics of SDMTAs in terms of steps and runs, and we prove the decidability of the reachability. This allows proving properties expressible in terms of reached states.

In [12], a model of Pushdown Timed Automata, i.e. Timed Automata augmented with a pushdown stack, is considered and a decidable characterization for reachability is given.

In [8], the authors study the class of Pushdown Counter Linear Hybrid Systems, i.e. linear hybrid systems equipped with a stack and integer counters. Decidability of invariance/reachability properties of some subclasses of this model are proved. The considered subclasses are Timed Automata equipped with a stack and integer monotonic variables.

In [15], similar results for Timed Automata with discrete clocks are proved.

In [18], the author considers the class of Hybrid Systems (i.e. Timed Automata where variables can evolve by following general evolution laws) equipped with arrays. For this kind of systems the reachability problem is undecidable. However, an algorithm based on predicate transformation is provided for computing the successor operator (which implies the semidecidability of the reachability problem). Moreover, more

general data-structures can be handled since, by using arrays, one can simulate stacks, queues, etc..

Communicating Timed Automata (CTAs) are systems of Timed Automata that can send/receive messages exchanged via unbounded FIFO channels [17]. While CTAs with two (or more) channels have the power of Turing machines, state reachability is decidable for CTAs with just one channel (as it happens for 1-SDMTAs). Among the differences between the model of CTAs and the model of SDMTAs introduced in the present paper (whose preliminary version appeared in [19]) is that in CTAs the local store is not unbounded, namely, even if channels are not bounded, the Communicating Timed Automata can only operate on the head of a channel buffer. Moreover, the domain of the messages that could be sent/received by CTAs is finite.

In [2] a method for deciding reachability properties of networks of timed processes is presented. Such a network consists of an arbitrary set of identical timed automata, each with a single real-valued clock. Their systems are infinite-state in two directions. They contain an arbitrary set of processes, and they use infinite data structures, namely real-valued clocks.

In [3] data structures and real-time features are studied in an original combination of three well established formal techniques for the specification of concurrent processes: Hoare's CSP, Object-Z [23] and the Duration Calculus [26].

## 2 Basic Notions

Let us assume a set  $X$  of positive real variables  $x$  called *clocks*. A *valuation* over a set of clocks is a mapping  $v : X \rightarrow \mathbb{R}^{\geq 0}$  assigning real values to clocks. For a valuation  $v$  and a time value  $t \in \mathbb{R}^{\geq 0}$ , let  $v+t$  denote the valuation such that  $(v+t)(x) = v(x) + t$ , for each clock  $x \in X$ .

Let  $C = \{C_1, \dots, C_m\}$  be a finite set of disjoint sets where  $C_i$  denotes a finite set of elementary messages. By an elementary message we mean a non composed/manipulated message (i.e., names are elementary messages, lists of elementary messages are not).

Let us assume a set  $\mathcal{Y}$  of message variables  $\mu$  that can take values in  $(\cup_{j=1}^m C_j) \cup \mathbb{N}$ . Given a finite set of message variables, an instance  $I$  relates a message variable  $\mu$  to a value in  $(\cup_{j=1}^m C_j) \cup \mathbb{N}$ . Namely,  $I : \mathcal{Y} \rightarrow (\cup_{j=1}^m C_j) \cup \mathbb{N}$ .

Let  $\Omega = \{f_1, \dots, f_n\}$  denote a finite set of function symbols. Given a finite set of message variables  $\mathcal{Y}$ , the set of terms  $\mathcal{T}(\mathcal{Y})$  is defined as:

$$\tau ::= c \mid w \mid \mu \mid f(\tau_1, \dots, \tau_k)$$

where  $c \in C_i$  for some  $i$ ,  $w \in \mathbb{N}$ ,  $\mu \in \mathcal{Y}$  is a message variable,  $f$  is a function in  $\Omega$  with arity  $k$ .

We use  $C$  and  $\Omega$  to represent a set of data structures and a set of functions to manipulate such structures. In general,  $C$  may be any set of data structures of different types and  $\Omega$  may be any set of functions representing operations on such structures.

*Example 1* Consider  $C = \{m_1, m_2, \dots\}$  a set of basic messages (i.e. a set of plaintext messages represented by bitstrings of a fixed length).

In a framework where agents exchange messages, we want to describe concatenations of messages and the possibility of distinguishing two messages sent at different

instants of time (this is done usually with a time stamp or a nonce in a security protocol).

Hence we assume  $\Omega = \{Pair, Occurrence\}$ .  $Pair(\tau_1, \tau_2)$  denotes the concatenation of the terms  $\tau_1$  and  $\tau_2$ , and,  $Occurrence(\tau, n)$ , with  $n \in \mathbb{N}$ , associates with  $\tau$  a natural number in such a way that  $Occurrence(\tau, n)$  and  $Occurrence(\tau, n')$  with  $n \neq n'$  are distinguishable.

With  $Var(\tau)$  we denote the message variables appearing in the term  $\tau$ . For example, we have that  $Var(Pair(\mu_1, Occurrence(\mu_2, 100))) = \{\mu_1, \mu_2\}$ .

We say that two terms  $\tau$  and  $\tau'$  in  $\mathcal{T}(\mathcal{Y})$  are *reducible* (written  $\tau \simeq \tau'$ ) if they have the same structure, namely  $\tau \simeq \tau'$  if there exist variables  $\mu_1, \dots, \mu_n$  and  $\bar{\mu}_1, \dots, \bar{\mu}_n \in \mathcal{Y}$  such that  $\tau = \tau'[\bar{\mu}_1/\mu_1] \dots [\bar{\mu}_n/\mu_n]$ .

Finally, with  $\mathcal{K}$  we denote a knowledge. A knowledge  $\mathcal{K} \subset \mathcal{T}(\mathcal{Y})$  is a finite set of terms  $\tau$  such that  $Var(\tau) = \emptyset$ .

Given a finite set of clocks  $X$  and a finite set of message variables  $\mathcal{Y}$ , we define the set  $\Phi(X, \mathcal{Y})$  of *formulae* as follows:

$$\begin{aligned} \phi ::= & true \mid \tau \in \tilde{\mathcal{K}} \mid \tau = \tau' \mid \mu \in \mathbb{N} \mid \\ & x \sim c \mid x \sim y + c \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \end{aligned}$$

where  $\phi, \phi_1, \phi_2 \in \Phi(X, \mathcal{Y})$ ,  $\tau, \tau' \in \mathcal{T}(\mathcal{Y})$ ,  $\mu \in \mathcal{Y}$ ,  $x, y \in X$ ,  $\sim \in \{<, \leq, =, \geq, >\}$  and  $c \in \mathbb{Q}$  (note that for decidability results  $c$  cannot be a real number).

The difference between  $\tilde{\mathcal{K}}$  and  $\mathcal{K}$  is that  $\tilde{\mathcal{K}}$  is a metavariable occurring in the conditions  $\Phi(X, \mathcal{Y})$  and  $\mathcal{K}$  (or  $\mathcal{K}_i$  with  $i = 1, \dots, m$  when we have a parallel compositions of  $m$  components) is a knowledge instance of  $\tilde{\mathcal{K}}$ . This is necessary, since using the same notation could be ambiguous.

We will write  $\tau \neq \tau'$  for  $\neg(\tau = \tau')$ ,  $\mu \notin \mathbb{N}$  for  $\neg(\mu \in \mathbb{N})$ , and  $\tau \in \{\tau_1, \dots, \tau_k\}$  for  $\tau = \tau_1 \vee \dots \vee \tau = \tau_k$ . As an example, with  $\tau \in C_i$ , for some  $i$ , we denote the formula  $\bigvee_{m \in C_i} \tau = m$ .

Given a term  $\tau$  and an instance  $I$ , we define the instantiation of  $\tau$  as  $I(\tau) = \tau'$ , where  $\tau'$  is the term resulting after each  $\mu$  syntactically occurring in  $\tau$  is replaced with  $I(\mu)$ .

*Example 2* Given the instance  $I$  such that  $I(\mu_1) = 10$  and  $I(\mu_2) = msg$  for an elementary message  $msg \in C$ , then:

$$I(Pair(\mu_1, Occurrence(\mu_2, 100))) = Pair(10, Occurrence(msg, 100)).$$

We recall that since a message variable cannot be instantiated with a term, there is no instance  $I'$  such that  $I'(Pair(\mu_1, \mu_2)) = Pair(10, Occurrence(msg, 100))$ . Actually,  $I'$  cannot assign the term  $Occurrence(msg, 100)$  to  $\mu_2$ .

Let  $\phi \in \Phi(X, \mathcal{Y})$ ,  $I$  be an instance,  $v$  a valuation of clocks, and  $\mathcal{K}$  a knowledge; we say that  $I, v$  and  $\mathcal{K}$  *satisfy*  $\phi$ , written  $I, v, \mathcal{K} \models \phi$ , in the following cases:

$$\begin{aligned} I, v, \mathcal{K} \models & true \\ I, v, \mathcal{K} \models & \tau \in \tilde{\mathcal{K}} \quad \text{iff } I(\tau) \in \mathcal{K} \\ I, v, \mathcal{K} \models & \tau = \tau' \quad \text{iff } I(\tau) = I(\tau') \\ I, v, \mathcal{K} \models & \mu \in \mathbb{N} \quad \text{iff } I(\mu) \in \mathbb{N} \\ I, v, \mathcal{K} \models & x \sim c \quad \text{iff } v(x) \sim c \\ I, v, \mathcal{K} \models & x \sim y \quad \text{iff } v(x) \sim v(y) \\ I, v, \mathcal{K} \models & \neg\phi_1 \quad \text{iff } I, v, \mathcal{K} \not\models \phi_1 \\ I, v, \mathcal{K} \models & \phi_1 \vee \phi_2 \quad \text{iff either } I, v, \mathcal{K} \models \phi_1 \text{ or } I, v, \mathcal{K} \models \phi_2 \\ I, v, \mathcal{K} \models & \phi_1 \wedge \phi_2 \quad \text{iff both } I, v, \mathcal{K} \models \phi_1 \text{ and } I, v, \mathcal{K} \models \phi_2. \end{aligned}$$

*Example 3* The formula  $\mu_1 \neq \mu_2 \wedge \mu_1 \in \tilde{\mathcal{K}} \wedge \mu_2 \in \tilde{\mathcal{K}}$  says that in the knowledge  $\mathcal{K}$  there are at least two elementary messages.

### 3 Data Management Timed Automata

Given a finite set of clocks  $X$  and a finite set of message variables  $\mathcal{Y}$ , with  $X'$  and  $\mathcal{Y}'$  we denote new sets of clocks and message variables such that  $x' \in X'$  and  $\mu' \in \mathcal{Y}'$  iff  $x \in X$  and  $\mu \in \mathcal{Y}$ , respectively.

A System of Data Management Timed Automata (SDMTA) is a tuple expressed by  $\mathcal{A} = (C, \Omega, A_1, \dots, A_m)$ , where:

- $C = \{C_1, \dots, C_k\}$  is a set of elementary messages;
- $\Omega = \{f_1, \dots, f_n\}$  is a set of functions;
- $A_1, \dots, A_m$  are Data Management Timed Automata (DMTAs).

A DMTA is a tuple  $A = (\Sigma, X, \mathcal{Y}, Q, q_0, \delta)$ , where:

- $\Sigma$  is a finite set of channel labels;
- $X$  is a finite set of clocks;
- $\mathcal{Y}$  is a finite set of message variables;
- $Q$  is a finite set of states with  $q_0 \in Q$  initial state;
- $\delta$  is a finite set of transitions. Each transition is a tuple  $(q, \alpha, \phi, q')$ , where  $q, q' \in Q$  are the source and the target states respectively, for  $a \in \Sigma$ ,  $\alpha \in \{\epsilon(\tau), a?(\tau), a!(\tau)\}$  represents, respectively, an internal move producing the term  $\tau$ , which enriches the knowledge of  $\mathcal{A}$ , or the input or the output of the term  $\tau \in \mathcal{T}(\mathcal{Y})$  on channel  $a$ ,  $\phi$  is a formula in  $\Phi(X \cup X', \mathcal{Y} \cup \mathcal{Y}')$ . Variables in  $X \cup \mathcal{Y}$  and in  $X' \cup \mathcal{Y}'$  represent the value of variables before and after the firing of the transition, respectively.

An example transition is  $(q_0, a!(f(\mu)), x < 5 \wedge \mu \in \tilde{\mathcal{K}} \wedge x' = x \wedge \mu' \in \mathbb{N}, q_1)$ , stating that from the initial state  $q_0$  the DMTA may reach state  $q_1$  and output a message  $f(\mu)$  when  $\mu \in \tilde{\mathcal{K}}$  and  $x < 5$ . The condition  $x' = x$  means that the transition does not change the clock  $x$ , and the condition  $\mu' \in \mathbb{N}$  means that the variable  $\mu$  nondeterministically assumes a natural number as value after the transition.

Given an SDMTA  $\mathcal{A} = (C, \Omega, A_1, \dots, A_m)$ , the states of  $\mathcal{A}$  are represented by tuples  $(q_1, \dots, q_m)$ , where  $q_i \in Q^i$ , where  $Q^i$  is the set of states of  $A_i$ .

#### 3.1 Semantics

Given two valuations  $v_1, v_2$  over  $X$ , with  $v_1 \oplus v_2$  we denote the valuation on  $X \cup X'$  such that for any  $x \in X$ ,  $(v_1 \oplus v_2)(x) = v_1(x)$  and  $(v_1 \oplus v_2)(x') = v_2(x)$ . The idea is the the operator  $\oplus$  is to split the valuation on  $X \cup X'$  in two parts, one arguing on the actual values of the clocks and one arguing on the values of the clocks after the step (hence the value of clocks in  $X'$ ).

In a similar manner we can define the same operator for the instances. Actually, given two instances  $I_1, I_2$  over  $\mathcal{Y}$ , with  $I_1 \oplus I_2$  we denote the instance over  $\mathcal{Y} \cup \mathcal{Y}'$  such that for any  $\mu \in \mathcal{Y}$ ,  $(I_1 \oplus I_2)(\mu) = I_1(\mu)$  and  $(I_1 \oplus I_2)(\mu') = I_2(\mu)$ .

A configuration of an SDMTA  $\mathcal{A} = (C, \Omega, A_1, \dots, A_m)$  with DMTAs defined as  $A_i = (\Sigma^i, X^i, \mathcal{Y}^i, Q^i, q_0^i, \phi_0^i, \mathcal{K}_0^i, \delta^i)$ , is a tuple  $(s_1, \dots, s_m)$  such that  $s_i = (q, v, I, \mathcal{K})$  is

a configuration of the DMTA  $A_i$  with  $q \in Q^i$  a state of  $A_i$ ,  $v$  a valuation over  $X^i$ ,  $I$  an instance over  $\mathcal{T}^i$  and  $\mathcal{K}$  a knowledge.

We have three kinds of steps:  $\epsilon$ -transition steps, communication transition steps and time steps. An  $\epsilon$ -transition step is performed by one only component when a transition labeled with  $\epsilon$  is performable. A communication transition step consists in a synchronization with message exchange between two components. Time steps represent the elapsing of time.

Given two configurations  $s = (s_1, \dots, s_m)$  and  $s' = (s'_1, \dots, s'_m)$  such that  $s_i = (q_i, v_i, I_i, \mathcal{K}_i)$  and  $s'_i = (q'_i, v'_i, I'_i, \mathcal{K}'_i)$ , we have that:

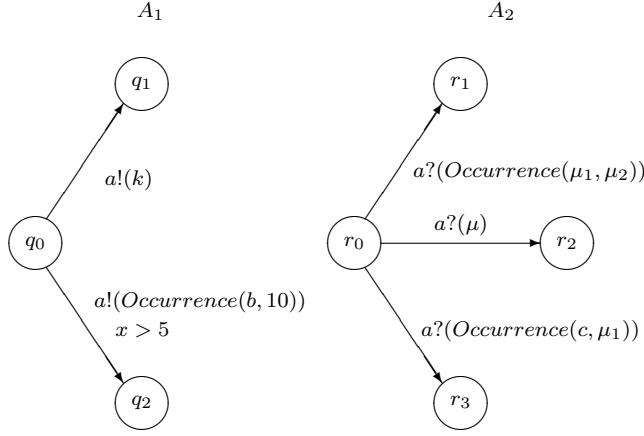
- there is an  $\epsilon$ -transition step from  $s$  to  $s'$  (denoted  $s \rightarrow_\tau s'$ ) if there exists an index  $i$  and a transition  $e = (q_i, \epsilon(\tau_1), \phi, q'_i) \in \delta^i$  such that  $I_i(\tau_1) = \tau$ ,  $(I_i \oplus I'_i), (v_i \oplus v'_i), \mathcal{K}_i \models \phi$ ,  $\mathcal{K}'_i = \mathcal{K}_i \cup \{\tau\}$  and, for all  $j \neq i$ ,  $s'_j = s_j$ ;
- there is a communication transition step from  $s$  to  $s'$  on channel  $a$  with term  $\tau$  (denoted  $s \rightarrow_{a(\tau)} s'$ ) if there exist two different indexes  $i$  and  $j$ ,  $(q_i, a!(\tau_1), \phi_1, q'_i) \in \delta^i$  and  $(q_j, a?(\tau_2), \phi_2, q'_j) \in \delta^j$  such that:
  - $\tau_1 \simeq \tau_2$  and  $I_i(\tau_1) = I_j(\tau_2) = \tau$ ;
  - $(I_i \oplus I'_i), (v_i \oplus v'_i), \mathcal{K}_i \models \phi_1$  and  $(I_j \oplus I'_j), (v_j \oplus v'_j), \mathcal{K}_j \models \phi_2$ ;
  - $\mathcal{K}'_i = \mathcal{K}_i \cup \{\tau\}$  and  $\mathcal{K}'_j = \mathcal{K}_j \cup \{\tau\}$ ;
  - for all  $k \notin \{i, j\}$ , it holds that  $s'_k = s_k$ ;
- there is a time step from  $s$  to  $s'$  with delay  $t \in \mathbb{R}^{>0}$ , written  $s \rightarrow_t s'$ , if, for any  $i$ ,  $q'_i = q_i$ ,  $v'_i = (v_i + t)$ ,  $I'_i = I_i$ , and  $\mathcal{K}'_i = \mathcal{K}_i$ .

With the  $\epsilon$ -transition step we model the internal data manipulation executed by a DMTA without communication. For example, given a configuration  $(q, v, I, \mathcal{K})$ , transition  $(q, \epsilon(\mu_1), \phi, q')$ , where  $\phi = \text{Pair}(\mu_1, \mu_2) \in \tilde{\mathcal{K}}$ , states that if  $(I \oplus I'), (v \oplus v'), \mathcal{K} \models \phi$ , the DMTA may enrich its knowledge with the left part of a concatenated message.

A communication transition step  $\rightarrow_{a(\tau)}$  models the communication of term  $\tau$  on channel  $a$ . There is a synchronization between two DMTAs, and they both change their configuration by following the formula in the transition and by augmenting their knowledge. If variables appear in the output and input terms ( $\tau_1$  and  $\tau_2$ , respectively), they should be *reducible* for transmission. Hence, we require that  $\tau \simeq \tau'$  and that  $I_i(\tau_1) = I_j(\tau_2)$  (also see Example 4). The DMTAs of the system not involved in the communication remain in their respective original configurations. Although it is often the case that the parties involved in a communication send data which are already present in their knowledge, our semantics allows a component to send a message even if this is not within its knowledge. As a consequence, components are able to provide new messages. This is necessary when fresh terms (for example nonces or keys in cryptographic protocols) should be created. On the other hand, one can always impose that a term to be exchanged is within the knowledge of the sending component by imposing it on the transition guard (i.e. through the formula  $\tau \in \tilde{\mathcal{K}}$ ).

Finally, a time step models time elapsing. When time elapses we assume that each DMTA  $A_i$  in the system performs a time step by changing its valuation  $v_i$ .

Sometimes, an initial condition and, in our case, also an initial knowledge could be introduced in the definition. The purpose of these conditions is to initialize the variables and the knowledge. It is obvious that, if one wants to simulate an initial knowledge and initial values for clocks and variables, it is sufficient to create a sequence of new initial states linked by new transitions initializing knowledge, clocks and variables.



**Fig. 1** A System of Data Management Timed Automata.

*Example 4* Consider the DMTAs  $A_1$  and  $A_2$  in Figure 1, where  $q_0$  and  $r_0$  are the initial states of  $A_1$  and  $A_2$ , respectively. The only communications that may happen between  $A_1$  and  $A_2$  through synchronization steps of matching terms are either  $a(\text{Occurrence}(b, 10))$  (synchronizing with the transition from  $r_0$  to  $r_1$  when  $x$  has value greater than 5) or  $a(k)$  (synchronizing with the transition from  $r_0$  to  $r_2$ ). In the first case, variable  $\mu_1$  is instantiated with value  $b$  and variable  $\mu_2$  with the natural number value 10, in the second case, variable  $\mu$  assumes the value  $k$ . In both cases, the knowledge of  $A_2$  is augmented with the term received by  $A_1$ .

Given a DMTA  $A = (\Sigma, X, \mathcal{T}, Q, q_0, \delta)$ , a configuration  $s = (q_0, v, I, \mathcal{K})$  of  $A$  is *initial* if  $I(\mu) = 0$ , for any  $\mu$ ,  $v(x) = 0$ , for any  $x$ , and,  $\mathcal{K} = \emptyset$ . Given an SDMTA  $\mathcal{A} = (C, \Omega, A_1, \dots, A_m)$ , we say that the configuration  $s = (s_1, \dots, s_m)$  of  $\mathcal{A}$  is initial iff  $s_i$  is an initial configuration of the DMTA  $A_i$  for all  $i \in [1, m]$ .

A *run* of an SDMTA  $\mathcal{A}$  is a finite sequence of steps  $\sigma = s_0 \rightarrow_{\alpha_1} s_1 \rightarrow_{\alpha_2} \dots \rightarrow_{\alpha_l} s_l$ , where  $s_0$  is an initial configuration of  $\mathcal{A}$ ,  $s_j$  is a configuration of  $\mathcal{A}$  and  $\alpha_j \in \{\tau, a(\tau)\} \cup \mathbb{R}^{>0}$  for each  $j \in [1, l]$ .

A state  $\bar{q} = (q_1, \dots, q_m)$  of an SDMTA  $\mathcal{A}$  is *reachable* iff there is a run  $\sigma = (s_0^1, \dots, s_0^m) \rightarrow_{\alpha_1} \dots \rightarrow_{\alpha_l} (s_l^1, \dots, s_l^m)$  of  $\mathcal{A}$  such that, for any  $i$ , if  $s_l^i = (q, v, I, \mathcal{K})$ , then  $q = q_i$ .

A state  $q$  of a DMTA  $A_i$  in the SDMTA  $\mathcal{A}$  is *reachable* iff there is a run  $\sigma = (s_0^1, \dots, s_0^m) \rightarrow_{\alpha_1} \dots \rightarrow_{\alpha_l} (s_l^1, \dots, s_l^m)$  of  $\mathcal{A}$  such that  $s_l^i = (q, v, I, \mathcal{K})$ , for some  $v, I, \mathcal{K}$ .

### 3.2 $n$ -SDMTAs

We define the class of  $n$ -SDMTAs. The idea is that an  $n$ -SDMTA can store in its knowledge an unbounded number of occurrences of instances of a term  $\tau$  by varying in an unbounded manner at most  $n$  arguments of  $\tau$ .

As an example the term  $f(2)$  has a finite number of instances, while  $f(\mu)$  has an unbounded number of instances (and hence in the knowledge we can have an unbounded number of occurrences of instances of  $f(\mu)$  if  $\mu$  can be a natural number.

If we consider the term  $g(\mu_1, 5, \mu_2)$ , then also in this case the term has an unbounded number of instances but the “degrees of freedom” are double with respect to  $f(\mu)$ . Actually in  $g(\mu_1, 5, \mu_2)$  we have two possible natural variables against the one of  $f(\mu)$ .

The idea is that the class  $n$ -SDMTAs has  $n$  different natural variables appearing contemporarily in the same term.

With the purpose of defining the class of  $n$ -SDMTAs we define a function  $natVar$ .

Given a term  $\tau$  and a set of instances  $\mathcal{I}$ , with  $natVar(\tau, \mathcal{I})$  we denote the maximum number of natural number variables that can appear in  $\tau$  w.r.t. instances in  $\mathcal{I}$ , namely,  $max_{I \in \mathcal{I}} |\{\mu \mid \mu \in Var(\tau) \text{ and } I(\mu) \in \mathbb{N}\}|$ .

We can extend this definition on  $\phi$  as follows:

- $natVar(true, \mathcal{I})$ ,  $\mu \in \mathbb{N}$ ,  $x \sim c$  and  $x \sim y$  is equal to 0;
- $natVar(\tau \in \tilde{\mathcal{K}}, \mathcal{I})$  is equal to  $natVar(\tau, \mathcal{I})$ ;
- $natVar(\tau = \tau', \mathcal{I})$  is equal to the maximum between  $natVar(\tau, \mathcal{I})$  and  $natVar(\tau', \mathcal{I})$ ;
- $natVar(\phi_1 \vee \phi_2, \mathcal{I})$  and  $natVar(\phi_1 \wedge \phi_2, \mathcal{I})$  is equal to the maximum between  $natVar(\phi_1, \mathcal{I})$  and  $natVar(\phi_2, \mathcal{I})$ ;
- $natVar(\neg\phi_1, \mathcal{I})$  is equal to  $natVar(\phi_1, \mathcal{I})$ .

When we omit the set of instances  $\mathcal{I}$  we consider the case in which  $\mathcal{I} = \{I \mid I \models \phi\}$ . Hence, with  $natVar(\phi)$  we denote the maximum number of natural number variables that can appear simultaneously in a term in  $\phi$ . Namely,  $natVar(\phi) = natVar(\phi, \{I \mid I \models \phi\})$ .

As an example,  $natVar(f(\mu, \mu') \in \tilde{\mathcal{K}}) = 2$  and  $natVar(f(\mu, \mu') \in \tilde{\mathcal{K}} \wedge \mu \notin \mathbb{N}) = 1$ . In the condition  $f(\mu, \mu') \in \tilde{\mathcal{K}}$  both  $\mu$  and  $\mu'$  can assume contemporarily natural number values. On the contrary, in the condition  $f(\mu, \mu') \in \tilde{\mathcal{K}} \wedge \mu \notin \mathbb{N}$  only  $\mu'$  can assume natural number values thanks to the subformula  $\mu \notin \mathbb{N}$ .

With  $Form(\mathcal{A})$  we denote the set of formulas  $\phi$  appearing in the transitions of  $\mathcal{A}$ .

Finally, we can extend this definition to SDMTAs and DMTAs.

**Definition 1** With  $n$ -SDMTAs (resp.  $n$ -DMTAs) we denote the class of automata  $\mathcal{A}$  such that  $(max_{\phi \in Form(\mathcal{A})} natVar(\phi)) = n$ .

### 3.3 $n$ -Standard Form for $n$ -SDMTAs

We define an  $n$ -standard form for SDMTAs and we prove that the reachability problem for  $n$ -SDMTAs can be translated into the reachability problem for  $n$ -SDMTAs in  $n$ -standard form.

Our purpose is to simplify the definition of SDMTA in order to simplify the construction of the proof showing the decidability of the reachability problem. Hence, the  $n$ -standard form allows:

- only one component;
- only natural number variables that are at the beginning pairwise equal;
- 0 as starting values of the clocks;
- $\emptyset$  as starting value of the knowledge;
- only terms of the form  $f(\mu_1, \dots, \mu_n)$  (which we will call  $n$ -simple).

Thus, the first step is to transform an  $n$ -SDMTA with more components into an  $n$ -SDMTA with just one component.

**Proposition 1** *Given an  $n$ -SDMTA  $\mathcal{A}$  and a state  $\bar{q} = (q_1, \dots, q_m)$  of  $\mathcal{A}$ , an  $n$ -SDMTA  $\mathcal{A}'$  composed of only one  $n$ -DMTA with a state  $\hat{q}$  can be constructed such that  $\bar{q}$  is reachable in  $\mathcal{A}$  iff  $\hat{q}$  is reachable in  $\mathcal{A}'$ .*

**Proof.** It is sufficient to consider the cartesian product of the sequential components of  $\mathcal{A}$ .

Now, a configuration fixes, for each component, its knowledge. Therefore if a component  $A_i$  checks for the condition  $\tau \in \tilde{\mathcal{K}}$ , this is checked in the knowledge of  $A_i$ .

If we consider the cartesian product, the knowledge is unique. This is a problem because in the cartesian product of the parallel composition of a set of components, the presence in the knowledge of a term  $\tau$  is ambiguous, actually, is not clear which component knows  $\tau$ .

Therefore, when a term enters into the whole knowledge, we must mark it with the index of components knowing this term.

Now, for the parallel composition, there are three manners for introducing a term  $\tau$  in the knowledge:  $\tau$  is sent to another component,  $\tau$  is received from another component, and,  $\tau$  is inserted by an  $\epsilon$ -transition.

Hence, the cartesian product of the parallel composition simulates the steps of the parallel composition and therefore it must update its knowledge when simulating these three cases. For a simple and uniform construction, we consider a new function *triple*. The idea is that, the term *triple*( $i, j, \tau$ ) is in the knowledge of the cartesian product of the parallel components for expressing that  $\tau$  is in the knowledge of the  $i^{th}$  and  $j^{th}$  components thanks to a communication between  $i$  and  $j$ . To express the  $\epsilon$ -transition, we consider the special term *triple*( $i, i, \tau$ ) expressing the fact that  $\tau$  is in the knowledge of the  $i^{th}$  component thanks to an  $\epsilon$ -transition step.

Hence a communication step or an  $\epsilon$ -step of the parallel composition over the term  $\tau$  is simulated by the cartesian product by an  $\epsilon$  step that update the knowledge with the term  $\tau$  marked with the index of the component knowing it.

Before formalizing the construction, we define the construction of a formula  $\phi(i)$  that simulates in the cartesian product the condition  $\phi$  appearing in the component  $i^{th}$  of the parallel composition. As exposed before, if the component  $i^{th}$  checks for  $\tau \in \tilde{\mathcal{K}}$ , then the cartesian product must check whether in its knowledge there is:

- or *triple*( $i, j, \tau$ ) for some  $j \neq i$ , namely component  $i^{th}$  knows  $\tau$  because it sent it to component  $j^{th}$ ;
- or *triple*( $j, i, \tau$ ) for some  $j \neq i$ , namely component  $i^{th}$  knows  $\tau$  because it received it from component  $j^{th}$ ;
- or *triple*( $i, i, \tau$ ), namely component  $i^{th}$  knows  $\tau$  because it stored it in its knowledge by an  $\epsilon$ -step.

Formally, given a formula  $\phi$ , with  $\phi(i)$  we denote the formula  $\phi$  where each formula of the form  $\tau \in \tilde{\mathcal{K}}$  is replaced with  $\left( \bigvee_{j \in [1, m]} \text{triple}(i, j, \tau) \in \tilde{\mathcal{K}} \vee \text{triple}(j, i, \tau) \in \tilde{\mathcal{K}} \right)$ .

We now formalize the construction of the cartesian product. Given the SDMTA  $\mathcal{A} = (C, \Omega, A_1, \dots, A_m)$ , where  $A_i = (\Sigma^i, X^i, \Upsilon^i, Q^i, q_0^i, \delta^i)$  for  $i = 1, \dots, m$ , we construct  $\mathcal{A}' = (C, \Omega', A)$ , where  $\Omega' = \Omega \cup \{\text{triple}\}$  and  $A = (\Sigma, X, \Upsilon, Q, q_0, \delta)$ . The set  $\Sigma = \emptyset$  does not contain any channel label, and the sets  $X = \bigcup_i X^i$ ,  $\Upsilon = \bigcup_i \Upsilon^i$ . The set of states  $Q = Q^1 \times \dots \times Q^m$  is given by the cartesian product of the set of states of each  $A^i$  with  $q_0 = (q_0^1, \dots, q_0^m)$  the initial state. The set of transitions  $\delta$  is defined as follows.

In  $\mathcal{A}'$  we introduce a transition

$$((q_1, \dots, q, \dots, q_m), \epsilon(\text{triple}(i, i, \tau)), \phi(i), (q_1, \dots, q', \dots, q_m))$$

if there exists a transition  $(q, \epsilon(\tau), \phi, q')$  of the  $i^{\text{th}}$  component of  $\mathcal{A}$ .

In  $\mathcal{A}'$  we introduce a transition

$$((q_1, \dots, q, \dots, q'', \dots, q_m), \epsilon(\text{triple}(i, j, \tau)), \phi(i) \wedge \phi'(j) \wedge \bar{\phi}, (q_1, \dots, q', \dots, q''', \dots, q_m))$$

if there exist two transitions  $(q, \alpha_1, \phi, q')$  and  $(q'', \alpha_2, \phi', q''')$  of the  $i^{\text{th}}$  and  $j^{\text{th}}$  components, respectively, such that  $\{\alpha_1, \alpha_2\} = \{a!(\tau), a?(\tau')\}$  and  $\bar{\phi}$  expresses the set of instances  $I$  and  $I'$  such that  $I(\tau) = I'(\tau')$ .

Obviously, this construction does not change the value of  $\text{natVar}$  of the system.

We prove now the correctness of our construction. Given a configuration  $s = (s_1, \dots, s_m)$  of  $\mathcal{A}$  with  $s_i = (q_i, I_i, v_i, \mathcal{K}_i)$  and a configuration  $s' = (q', I', v', \mathcal{K}')$  of  $\mathcal{A}'$ ,  $s'$  simulates  $s$  if it holds that:

1.  $q' = (q_1, \dots, q_m)$ ;
2.  $I'(\mu) = I_i(\mu)$  if  $\mu \in \Upsilon^i$ ;
3.  $v'(x) = v_i$  if  $x \in X^i$ ;
4. For any  $\text{triple}(i, j, \tau) \in \mathcal{K}'$ , it holds that  $\tau \in \mathcal{K}_i$  and  $\tau \in \mathcal{K}_j$ ;
5. For any  $\tau \in \mathcal{K}_i$ , there exists  $j$  such that  $\text{triple}(i, j, \tau) \in \mathcal{K}'$  or  $\text{triple}(j, i, \tau) \in \mathcal{K}'$ .

We prove now that, given  $\bar{s}$  simulating  $s$ , if  $s \rightarrow s'$  is a step of  $\mathcal{A}$ , then  $\bar{s} \rightarrow \bar{s}'$  is a step of  $\mathcal{A}'$  and  $\bar{s}'$  simulates  $s'$ .

Consider the configurations  $s = (s_1, \dots, s_m)$  with  $s_i = (q_i, I_i, v_i, \mathcal{K}_i)$ , and  $s' = (s'_1, \dots, s'_m)$  with  $s'_i = (q'_i, I'_i, v'_i, \mathcal{K}'_i)$ , and let  $\bar{s} = (q, I, v, \mathcal{K})$  and  $\bar{s}' = (q', I', v', \mathcal{K}')$ .

If  $\rightarrow$  is a time step, then, for any  $i$ , it holds that  $q'_i = q_i$ ,  $I'_i = I_i$ ,  $\mathcal{K}'_i = \mathcal{K}_i$  and there exists  $t$  such that  $v'_i = v_i + t$ . Hence, in  $\mathcal{A}'$  we can perform the same time steps and hence  $\bar{s}'$  simulates  $s'$ .

If  $\rightarrow$  is an  $\epsilon$ -transition step, then there exist  $i$  and a transition  $e = (q_i, \epsilon(\tau), \phi, q_i)$  such that  $(I_i \oplus I'_i), (v_i \oplus v'_i), \mathcal{K}_i \models \phi, \mathcal{K}' = \mathcal{K} \cup \{I_i(\tau)\}$ . Moreover, for any  $j \neq i$ , it holds that  $I'_j = I_j$ ,  $v'_j = v_j$  and  $\mathcal{K}'_j = \mathcal{K}_j$ .

Therefore, there exists a transition

$$((q_1, \dots, q, \dots, q_m), \epsilon(\text{triple}(i, i, \tau)), \phi(i), (q_1, \dots, q', \dots, q_m))$$

of  $\mathcal{A}'$ .

By 2, 3 and 5 and since  $\phi(i)$  is  $\phi$  where each formula of the form  $\tau \in \tilde{\mathcal{K}}$  is replaced with  $\bigvee_{j \in [1, m]} (\text{triple}(i, j, \tau) \in \tilde{\mathcal{K}} \vee \text{triple}(j, i, \tau) \in \tilde{\mathcal{K}})$ , we have that  $(I \oplus I'), (v \oplus v'), \mathcal{K} \models \phi(i)$ . Therefore the transition is performable and  $\mathcal{K}' = \mathcal{K} \cup \{\text{triple}(i, i, I(\tau))\}$ .

This implies that  $\bar{s}'$  simulates  $s'$ .

The last case in which  $\rightarrow$  is a communication step can be proved similarly.

Hence we have that, if  $s \rightarrow s'$  is a step of  $\mathcal{A}$ , then  $\bar{s} \rightarrow \bar{s}'$  is a step of  $\mathcal{A}'$  and  $\bar{s}'$  simulates  $s'$ .

Symmetrically one can prove that if  $\bar{s} \rightarrow \bar{s}'$  is a step of  $\mathcal{A}'$ , then  $s \rightarrow s'$  is a step of  $\mathcal{A}$  and  $\bar{s}'$  simulates  $s'$ .

Hence, since the initial configuration of  $\mathcal{A}'$  simulates the initial configuration of  $\mathcal{A}$ , by induction on the length of the runs we have the thesis. Actually  $s$  is reachable by  $\mathcal{A}$  iff  $\bar{s}$  is reachable by  $\mathcal{A}'$  where  $\bar{s}$  simulates  $s$ .  $\square$

A term  $\tau$  is  $n$ -simple for a set of message variables  $\Upsilon$  if  $\tau = f(\mu_1, \dots, \mu_n)$  with  $\mu_i \in \Upsilon$ . As an example,  $f(\mu_1, \mu_2)$  is 2-simple but non 1-simple, while  $f(\mu_1, f(\mu_2, \mu))$  and  $f(1, \mu_2)$  are not  $n$ -simple, for any  $n$ .

Now, we show that an  $n$ -SDMTA can always be transformed into an  $n$ -standard form that preserves reachability of states.

**Definition 2** An  $n$ -SDMTA  $\mathcal{A}$  is in  $n$ -standard form iff  $\mathcal{A} = (C, \Omega, (\Sigma, X, \Upsilon, Q, q_0, \delta))$  is composed by only one DMTA such that:

- $C = \emptyset$ ;
- for any  $(g, \epsilon(\tau), \phi, q') \in \delta$ , it holds that  $\tau$  is  $n$ -simple for  $\Upsilon$  and each term  $\tau'$  appearing in  $\phi$  is  $n$ -simple for  $\Upsilon$ .

For an SDMTA in  $n$ -standard form, message variables can assume only natural number values ( $C = \emptyset$ ). Finally, the conditions on formulas and on terms appearing in the transitions guarantee that an SDMTA in  $n$ -standard form only operates on  $n$ -simple terms. Thus, every term appearing in an  $n$ -SDMTA in  $n$ -standard form has the form  $f(\mu_1, \dots, \mu_n)$  for some  $f \in \Omega$  and  $\mu \in \Upsilon$ , with  $I(\mu) \in \mathbb{N}$  for all  $I$ .

We prove now that any  $n$ -SDMTA can be translated into an  $n$ -SDMTA in  $n$ -standard form.

**Proposition 2** Given an  $n$ -SDMTA  $\mathcal{A}$  and a state  $\bar{q} = (q_1, \dots, q_m)$  of  $\mathcal{A}$ , an  $n$ -SDMTA  $\mathcal{A}'$  in  $n$ -standard form with a state  $\hat{q}$  can be constructed such that  $\bar{q}$  is reachable in  $\mathcal{A}$  iff  $\hat{q}$  is reachable in  $\mathcal{A}'$ .

**Proof.** By Proposition 1 we can assume that  $\mathcal{A} = (C, \Omega, (\Sigma, X, \Upsilon, Q, q_0, \delta))$ .

We construct an  $n$ -SDMTA  $\mathcal{A}' = (C', \Omega', (\Sigma, X, \Upsilon, Q', \bar{q}_0, \delta'))$  in  $n$ -standard form.

The construction needs three steps.

The first step deletes each occurrence of natural values from the terms appearing in the SDMTA. The second step ensures that all message variable can assume only natural values. The third step transform each term in a  $n$ -simple term.

### Step 1

Now, we delete the natural numbers appearing in the terms.

Given a term  $\tau$ , with  $Nat(\tau)$  we denote the set of natural numbers appearing in  $\tau$ . If  $\mathcal{K}$  is a knowledge, with  $Nat(\mathcal{K})$  we denote the set  $\bigcup_{\tau \in \mathcal{K}} Nat(\tau)$ .

Now, we delete the natural numbers in  $Nat(\mathcal{A})$ . If  $Nat(\mathcal{A}) = \{n_1, \dots, n_k\}$ , then we consider  $c_1, \dots, c_k$  new constants and we substitute each  $n_i$  with  $c_i$ . Moreover, we replace each occurrence of  $n_i$  with  $c_i$ , and  $\mu \in \mathbb{N}$  with  $\mu \in \mathbb{N} \vee \mu \in \{c_1, \dots, c_k\}$ , for any  $\mu \in \Upsilon \cup \Upsilon'$ . Moreover, in each formula we add the formula  $\bigwedge_{\mu \in \Upsilon \cup \Upsilon'} \mu \notin \{n_1, \dots, n_k\}$ .

We prove that  $I, v, \mathcal{K} \models \phi$  iff  $I', v', \mathcal{K}' \models \phi'$  where:

- $I(\mu) = n_i$  iff  $I'(\mu) = c_i$
- $I(\mu) \notin Nat(\mathcal{A})$  implies  $I'(\mu) = I(\mu)$
- $v' = v$
- Let  $\tau$  be a term and  $\tau'$  is  $\tau$  where each  $n_i$  is substituted with  $c_i$ . It holds that  $\tau \in \mathcal{K}$  iff  $\tau' \in \mathcal{K}'$ .

The formula  $\bigwedge_{\mu \in \Upsilon \cup \Upsilon'} \mu \notin \{n_1, \dots, n_k\}$  ensures that the terms added to the knowledge of  $\mathcal{A}'$  and the instance  $I$  do not contain values in  $\{n_1, \dots, n_k\}$ .

Therefore the property holds since  $\mu \in \mathbb{N}$  is substituted with  $\mu \in \mathbb{N} \vee \mu \in \{c_1, \dots, c_k\}$  and we have substituted each occurrence of  $n_i$  in a term with  $c_i$ . Actually, this guarantees that  $I, v, \mathcal{K} \models \mu \in \mathbb{N}$  iff  $I', v', \mathcal{K}' \models \mu \in \mathbb{N} \vee \mu \in \{c_1, \dots, c_k\}$ , and,  $I, v, \mathcal{K} \models \tau \in \tilde{\mathcal{K}}$  iff  $I', v', \mathcal{K}' \models \tau' \in \tilde{\mathcal{K}}$  where  $\tau'$  is  $\tau$  where each  $n_i$  is substituted with  $c_i$ .

This property, implies the correctness of the construction by induction on the length of the run. Actually, the property holds for the initial configuration, and by definition

of step and by construction, this property still holds also after a step.

### Step 2

Now, we must guarantee that all variables can assume only natural values. Hence, for any  $\mu$ , if  $\{I(\mu) \notin \mathbb{N} \mid I \models \phi\}$  is not empty, then let  $H = \{I(\mu) \notin \mathbb{N} \mid I \models \phi\}$ ; we can substitute  $\phi$  with  $\mu \in \mathbb{N} \wedge ((\phi) \vee (\bigvee_{c \in H} \phi[c/\mu]))$ .

It is trivial that, if  $I(\mu) \notin \mathbb{N}$ , then  $I(\tau)$  is equal to  $I'(\tau[I(\mu)/\mu])$ , for any  $I'$  such that  $I'(\bar{\mu}) = I(\bar{\mu})$  if  $\bar{\mu} \neq \mu$  and  $I'(\mu) \in \mathbb{N}$  otherwise. Moreover, this implies that  $I, v, \mathcal{K} \models \phi$  iff  $I', v, \mathcal{K} \models \mu \in \mathbb{N} \wedge ((\phi) \vee \phi[I(\mu)/\mu])$ .

### Step 3

Finally, we want all terms to be in the form  $f(\mu_1, \dots, \mu_n)$ .

Let  $T$  be the set of terms appearing in the SDMTA after the previous transformations. We can partition  $T$  in  $T_1, \dots, T_h$  such that, for any  $\tau, \tau' \in T$ , it holds that  $\tau \simeq \tau'$  iff  $\tau, \tau' \in T_i$  for some  $i$ .

We then take  $\mathcal{O}' = \{f_1, \dots, f_h\}$ , and, for any  $\tau \in T_i$ , we replace  $\tau$  with the simple term  $f_i(\mu_{i_1}, \dots, \mu_{i_k})$ , where  $\mu_{i_1}, \dots, \mu_{i_k}$  is the sequence of variables occurring in  $\tau$  from left to right without repetition.

Now we prove that  $I, v, \mathcal{K} \models \phi$  iff  $I', v', \mathcal{K}' \models \phi'$  where:

1.  $I = I'$ .
2.  $v = v'$ .
3. for any  $\tau \in T_i$  and instance  $\bar{I}$ , we have that  $I(\tau) \in \mathcal{K}$  iff  $\bar{I}(f_i(\mu_{i_1}, \dots, \mu_{i_k})) \in \mathcal{K}'$  where  $\mu_{i_1}, \dots, \mu_{i_k}$  is the sequence of variables occurring in  $\tau$  from left to right without repetition.
4.  $\phi'$  is  $\phi$  where, for any  $\tau \in T_i$ , we replace  $\tau$  with the simple term  $f_i(\mu_{i_1}, \dots, \mu_{i_k})$ , where  $\mu_{i_1}, \dots, \mu_{i_k}$  is the sequence of variables occurring in  $\tau$  from left to right without repetition.

Since  $I = I'$  and  $v = v'$ , the only case we must check is  $I, v, \mathcal{K} \models \tau \in \tilde{\mathcal{K}}$  iff  $I', v', \mathcal{K}' \models f_i(\mu_{i_1}, \dots, \mu_{i_k}) \in \tilde{\mathcal{K}}$ . But this holds by item 3.

This implies the correctness of the construction by induction on the length of the run. Actually, since this property holds for the initial configuration and, by definition of step and by item 4, this property still holds also after a step.

Now, since  $\mathcal{A}$  is a  $n$ -SDMTA, all terms are in the form  $f(\mu_1, \dots, \mu_h)$  with  $h \leq n$  (note that constants can be viewed as the term  $f(\mu_1, \dots, \mu_h)$  with  $h = 0$ ). If  $h$  is strictly less than  $n$  ( $h < n$ ), it is sufficient to add  $n - h$  new variables  $\bar{\mu}_1, \dots, \bar{\mu}_{n-h}$  to the system to replace  $f(\mu_1, \dots, \mu_h)$  with  $f(\mu_1, \dots, \mu_h, \bar{\mu}_1, \dots, \bar{\mu}_{n-h})$ , and to add in each transition the condition  $\bigwedge_{j \in [1, n-h]} \bar{\mu}'_j = \bar{\mu}_j$  (namely  $\bar{\mu}_1, \dots, \bar{\mu}_{n-h}$  have a fixed value given non deterministically at the beginning).

With such a construction defined in four steps, we guarantee that a state  $q \in Q$  can be reached in  $\mathcal{A}$  iff the same can be reached also in  $\mathcal{A}'$ .

Again, this construction does not change the value of  $\text{natVar}$  of the system.  $\square$

## 3.4 Decidability of Reachability for 1-SDMTAs

We recall the definitions of clock equivalence [4]. Clock equivalence is a finite index equivalence relation allowing to group sets of valuations.

Let  $A$  be a DMTA and  $X_A$  be the greatest constant appearing in a condition involving clocks and appearing in  $A$ . Let us consider the equivalence relation  $\approx$  over clock valuations containing precisely the pairs  $(v, v')$  such that:

- for each clock  $x$ , either  $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ , or both  $v(x)$  and  $v'(x)$  are greater than  $X_A$ ;
- for each pair of clocks  $x$  and  $y$  with  $v(x) \leq X_A$  and  $v(y) \leq X_A$  it holds that  $\text{fract}(v(x)) \leq \text{fract}(v(y))$  iff  $\text{fract}(v'(x)) \leq \text{fract}(v'(y))$  (where  $\text{fract}(\cdot)$  is the fractional part);
- for each clock  $x$  with  $v(x) \leq X_A$ ,  $\text{fract}(v(x)) = 0$  iff  $\text{fract}(v'(x)) = 0$ .

Let  $[v]$  denote the equivalence class  $\{v' \mid v \approx v'\}$ . The set of equivalence classes  $\{[v] \mid v \text{ is a valuation}\}$  is finite, and with  $V$  we denote its cardinality.

Known properties of the equivalence classes are summarized in the following theorem [4].

**Theorem 1** *Given a DMTA  $A$ , let  $c < X_A$  and  $v$  and  $v'$  be two valuations;  $v \in [v']$  implies that  $v \models x \sim c$  iff  $v' \models x \sim c$ , where  $\sim \in \{<, \leq, =, \geq, >\}$ . Moreover, given a class  $[v]$ , the set  $\{[v + t] \mid t \in \mathbb{R}^{\geq 0}\}$  is computable.*

In order to check the reachability problem, in [4], Timed Automata are reduced to finite state machines. In our case, however, we cannot simulate an unbounded knowledge with a finite state machine. Hence, we prove the decidability of the reachability problem for 1-SDMTAs by reducing the problem to the reachability problem for Vector Addition Systems (VASs).

We briefly recall the model of VASs. Given two vectors  $w, w' \in \mathbb{Z}^n$ , with  $w_i$  we denote the  $i^{\text{th}}$  component of  $w$  and with  $w + w'$  we denote the vector  $w''$  such that  $w''_i = w_i + w'_i$ , for any  $i \in [1, n]$ .

**Definition 3** A *Vector Addition System (VAS) of dimension  $n$*  is a tuple  $\mathcal{S} = (Q, \delta)$  such that  $Q$  is a finite set of states and  $\delta \subseteq Q \times \mathbb{Z}^n \times Q$ . A *configuration* is a pair  $(q, w)$  with  $q \in Q$  and  $w \in \mathbb{N}^n$ . There exists a step from configuration  $(q, w)$  to configuration  $(q', w')$  (denoted with  $(q, w) \rightarrow (q', w')$ ) if  $(q, w'', q') \in \delta$  such that  $w' = w + w''$ .

Note that while vectors  $w$  appearing in configurations can only assume positive natural values ( $w \in \mathbb{N}^n$ ), vectors  $w' \in \delta$  may assume any integer value ( $w' \in \mathbb{Z}^n$ ).

The following theorem is proved in [16].

**Theorem 2** *Given two configurations  $(q, w)$  and  $(q', w')$  of a VAS  $\mathcal{S}$ , the problem of checking whether there exists a sequence of steps  $(q, w) \rightarrow \dots \rightarrow (q', w')$  is decidable and EXPSPACE hard.*

Before proving the reduction result from 1-SDMTAs to VASs, we briefly discuss about the complexity of this problem. Since the number of clock zones of a region automaton is exponential w.r.t. the number of clocks of a timed automaton, the VAS corresponding to a 1-SDMTA has a size that is at least exponential w.r.t. the number of clocks. Moreover, to simulate message variables, structured messages and knowledge, the VAS needs a number of states that is exponential w.r.t. the number of function symbols in  $\Omega$  and message variables in  $\mathcal{T}$ . Hence, the VAS resulting from a 1-SDMTA will have an exponential size w.r.t. the size of the 1-SDMTA.

**Theorem 3** *Given an 1-SDMTA  $\mathcal{A}$  and a state  $\bar{q}$  of  $\mathcal{A}$ , there exists a VAS  $\mathcal{S}$  of exponential size w.r.t.  $\mathcal{A}$  and two states  $q'$  and  $q''$  of  $\mathcal{S}$  such that  $\mathcal{A}$  reaches  $\bar{q}$  iff there exists a sequence of steps  $(q', (0, \dots, 0)) \rightarrow \dots \rightarrow (q'', (0, \dots, 0))$  of  $\mathcal{S}$ .*

Before proving the theorem, we consider an extension of VASs. If  $\mathcal{S} = (Q, \delta)$  we assume  $\delta \subseteq Q \times 2^{[1, n]} \times \mathbb{Z}^n \times Q$ . There exists a step from the configuration  $(q, w)$  to the configuration  $(q', w')$  if  $(q, In, w'', q') \in \delta$  such that  $w' = w + w''$  and  $w'_i > 0$  for any  $i \in In$ .

It is easy to see that, given an *extended*-VAS, we can construct an equivalent VAS with size polynomial w.r.t. the original one. Actually, it is sufficient to substitute any transition  $(q, In, w, q')$  with two transitions  $(q, w - \bar{w}, \hat{q}), (\hat{q}, \bar{w}, q')$ , where  $\hat{q}$  is a new state and  $\bar{w}_i = 1$  if  $i \in In$  and  $\bar{w}_i = 0$  otherwise. Note that, by definition, the values of  $w$ , for any configuration  $(q, w)$ , could not be smaller than 0. Therefore, the two transitions introduced above guarantee that the values corresponding to indexes in  $In$  will be kept greater than 0.

**Proof.** The proof can be divided into two main parts. In the first part we introduce the construction of an extended-VAS for a given 1-SDMTA, in the second part we show that for a sequence of steps performed by the 1-SDMTA, there is an analogous step in the corresponding VAS leading to a special state  $q_T$ .

**Part I:** Thanks to Propositions 2, we can suppose that  $\mathcal{A}$  is in  $n$ -standard form.

Let  $\Omega = \{f_1, \dots, f_k\}$ ,  $Q$ ,  $\mathcal{Y}$  and  $X$  be the set of function symbols, the set of states, the set of message variables, and the set of clocks of  $\mathcal{A}$ , respectively.

A state of the VAS  $\mathcal{S}$  is constructed as a tuple  $(q, [v], (A_1, T_1), \dots, (A_p, T_p))$  such that  $q \in Q$ ,  $v$  is a valuation on  $X$ , and  $\emptyset \neq A_i \subseteq \mathcal{Y}$  and  $T_i \subseteq \Omega$ , for any  $i$ , and  $\{A_1, \dots, A_p\}$  is a partition of  $\mathcal{Y}$ .

The idea is that  $(A_1, T_1), \dots, (A_p, T_p)$  express the instance  $I$  such that:

- $I(\mu_1) = I(\mu_2)$  iff  $\mu_1, \mu_2 \in A_i$ , for some  $i$ ;
- $f(\mu) \in \tilde{\mathcal{K}}$  is *true* iff  $\mu \in A_i$  and  $f \in T_i$ , for some  $i$ .

The dimension of  $\mathcal{S}$ , denoted by  $U = 2^{|\Omega|} - 1$ , is given by the number of possible sets  $T \subseteq \Omega$  such that  $T \neq \emptyset$ .

Since an instance that does not use any constant appearing in the knowledge is always definable, we do not consider the empty set.

Let  $g$  be a bijective function assigning a coordinate of the vector to each set  $T \subseteq \Omega$  such that  $T \neq \emptyset$ .

The configuration  $((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w)$  of  $\mathcal{S}$  expresses the configuration  $(q', v', I, \mathcal{K})$  of  $\mathcal{A}$  such that  $q = q'$ ,  $v' \in [v]$ ,  $(A_1, T_1), \dots, (A_p, T_p)$  define the instance  $I$  as shown before, and  $w \in \mathbb{N}^U$  represents the knowledge  $\mathcal{K}$ . More precisely, let  $T \subseteq \Omega$ ; if the  $g(T)$ -th coordinate of  $w$  is equal to  $h$ , then there exist  $c_1, \dots, c_h$  distinct values in  $\mathbb{N}$  such that, for any  $j$  and for any  $f \in T$ , it holds that  $f(c_j)$  is in the knowledge.

Summing up, we will write that a configuration  $((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w)$  represents a configuration  $(q', v', I, \mathcal{K})$  of  $\mathcal{A}$  if

- $q = q'$ ;
- $v' \in [v]$ ;
- $I(\mu_1) = I(\mu_2)$  iff  $\mu_1, \mu_2 \in A_i$ , for some  $i$ ;
- $f(\mu) \in \tilde{\mathcal{K}}$  is *true* iff  $\mu \in A_i$  and  $f \in T_i$ , for some  $i$ ;

- for any  $T \subseteq \Omega$ , it holds that, if  $w_{g(T)} = h$ , then there exist  $c_1, \dots, c_h$  distinct values in  $\mathbb{N}$  such that, for any  $j$  and for any  $f \in T$ , it holds that  $f(c_j)$  is in the knowledge.

The set of transitions of  $\mathcal{S}$  is constructed as follows.

To model time steps of  $\mathcal{A}$ , we add in  $\mathcal{S}$  transitions of the form  $(\bar{q}, In, w, \bar{q}')$ , where  $In = \emptyset$ ,  $w = (0, \dots, 0)$ ,  $\bar{q} = (q, [v], (A_1, T_1), \dots, (A_p, T_p))$ , for some  $q, [v], A_i$  and  $T_i$ , and  $\bar{q}' = (q, [v+t], (A_1, T_1), \dots, (A_p, T_p))$  for some  $t \in \mathbb{R}^{\geq 0}$ .

Moreover, there is a transition in  $\mathcal{S}$  from the state  $(q, [v], (A_1, T_1), \dots, (A_p, T_p))$  to the state  $(q', [v'], (A'_1, T'_1), \dots, (A'_r, T'_r))$  with label  $(In, w)$  if there exists a transition  $(q, \epsilon(\bar{f}(\bar{\mu}), \phi, q')$  of  $\mathcal{A}$  such that the following conditions hold:

- There exists  $(A''_1, T''_1), \dots, (A''_z, T''_z)$  representing the relations between actual and new values of message variables, where:
  - for any  $h$ ,  $T''_h \subseteq \Omega$ ;
  - $\{A''_1, \dots, A''_z\}$  is a partition of  $\mathcal{T} \cup \mathcal{T}'$ ;
  - for any  $h \in [1, p]$ , there exists  $k$  such that  $A_h = A''_k \cap \mathcal{T}$  and  $T_h = T''_k$ ; namely, the sequence  $(A''_1, T''_1), \dots, (A''_z, T''_z)$  preserves the relationships linking the actual values of the variables and the conditions that are satisfied by the actual instance and knowledge;
  - for any  $h \in [1, r]$ , there exists  $k$  such that  $A'_h = \{\mu \mid \mu' \in A''_k \cap \mathcal{T}'\}$  and if  $\bar{\mu} \in A''_h$  then  $T'_h = T''_k \cup \{\bar{f}\}$  otherwise if  $\bar{\mu} \notin A''_h$  then  $T'_h = T''_k$  (recall  $\bar{f}(\bar{\mu})$  is the term injected into the knowledge), namely, the sequence  $(A''_1, T''_1), \dots, (A''_z, T''_z)$  preserves the relationships linking the new values of the variables and the conditions that will be satisfied by the target instance and knowledge;
- $\phi$  is true when evaluating:
  - each clock  $x$  with  $(v \oplus v')(x)$ ,
  - each  $\mu \in \mathbb{N}$  as *true*,
  - $\mu_1 = \mu_2$  as *true* iff, for some  $h$ , it holds that  $\mu_1, \mu_2 \in A''_h$ ,
  - $f(\mu) \in \tilde{\mathcal{K}}$  is *true* iff for some  $h$ , it holds that  $\mu \in A''_h$  and  $f \in T''_h$ ;
hence, condition  $\phi$  holds for the valuations  $v$  and  $v'$  and when considering instances and conditions expressed by  $(A''_1, T''_1), \dots, (A''_z, T''_z)$ .
- $In = \{g(T_i) \mid T_i \neq \emptyset \text{ and } i = 1, \dots, r\}$ . This condition expresses the presence in the knowledge of the terms  $f(\mu)$  where  $f \in T_i$  and  $\mu \in A_i$ , namely ensures that  $f \in T_i$  iff  $f(\mu)$  is in the knowledge. Note that this implies that the starting state is coherent with the knowledge. Therefore the coherence of a state with the knowledge is guaranteed in the successive step. Moreover, an instance that does not use any constant appearing in the knowledge is always definable, hence, it is not necessary to check  $T_i$  such that  $T_i = \emptyset$ .
- Let  $h$  be such that  $\bar{\mu} \in A_h$ ; if  $\bar{f} \in T_h$ , then  $w = (0, \dots, 0)$ , otherwise
$$w_j = \begin{cases} -1 & \text{if } g^{-1}(j) = T_h \\ 1 & \text{if } g^{-1}(j) = \{\bar{f}\} \cup T_h \\ 0 & \text{otherwise.} \end{cases}$$

This a condition expresses the fact that, if  $\bar{f}(\bar{\mu})$  is in the knowledge (namely,  $\bar{f} \in T_h$  and  $\bar{\mu} \in A_h$ ), then the knowledge does not change (namely,  $w = (0, \dots, 0)$ ). Otherwise, the insertion of  $\bar{f}(\bar{\mu})$  creates new relations among the terms in the knowledge. Actually, the natural numbers appearing in  $\bar{f}(\bar{\mu})$  can appear in other terms in the knowledge. The terms that become related when  $\bar{f}(\bar{\mu})$  is inserted are those expressed by  $T_h$  with  $\bar{\mu} \in A_h$ . Hence, the case  $w_j = -1$  means that the set  $T_h$ , represented by coordinate  $j$  (namely,  $g^{-1}(j) = T_h$ ) that has relations with

$\bar{f}(\bar{\mu})$ , cannot still be activated by any instance. Actually,  $\bar{f}(\bar{\mu})$  does not introduce new relations. The case  $w_j = 1$  means that the set of terms expressed by  $\{\bar{f}\} \cup T_h$ , represented by coordinate  $j$  and expressing the new relations, can be activated by some instance.

Hence, we have a transition in  $\mathcal{S}$  for any step of  $\mathcal{A}$ . Each transition updates the values of the vector in the configuration by following the changes of the knowledge due to the step of  $\mathcal{A}$ . Note that, if a transition of  $\mathcal{S}$  models a time step of  $\mathcal{A}$ , the knowledge is left unchanged.

Our construction checks for the coherence of the state with the knowledge in the successive step. Actually, as explained before, the set  $In$  checks the coherence of the source state and not of the target state. So, we add a new state  $q_T$  to  $\mathcal{S}$  and transitions with label  $(In, (0, \dots, 0))$  and target  $q_T$  from each state  $((\bar{q}, [v], (A_1, T_1), \dots, (A_m, T_m)))$  with  $In = \{g(T_i) \mid i = 1, \dots, m\}$ . So when arriving at the state  $q_T$  all states have been checked for coherence.

**Part II:** We prove that, given a sequence of steps of  $\mathcal{A}$ , there exists a sequence of steps of the extended VAS ending with  $q_T$  where every crossed state of  $\mathcal{A}$  is represented by the states crossed by the VAS. We prove this fact by induction on the length of the sequence of steps. The base case is obvious, so we consider the inductive step.

We consider a run  $\sigma = (q, v, I, \mathcal{K}) \rightarrow (q', v', I', \mathcal{K}') \rightarrow \dots$

By induction, the thesis holds for the sequence  $\sigma' = (q', v', I', \mathcal{K}') \rightarrow \dots$  and there exists a sequence of steps  $((q', [v'], (A'_1, T'_1), \dots, (A'_r, T'_r)), w') \rightarrow \dots (q_T, w'')$ , for some  $w''$ . Hence we prove that the thesis holds for the first step of  $\sigma$ . More precisely, we prove that  $(q, v, I, \mathcal{K}) \rightarrow (q', v', I', \mathcal{K}')$  implies that there exists a step in  $\mathcal{S}$  of the form  $((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w) \rightarrow ((q', [v'], (A'_1, T'_1), \dots, (A'_r, T'_r)), w')$ , for some  $((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w)$  representing  $(q, v, I, \mathcal{K})$  (note that by induction  $((q', [v'], (A'_1, T'_1), \dots, (A'_r, T'_r)), w')$  represents  $(q', v', I', \mathcal{K}')$ ).

The step performed by  $\mathcal{A}$  may be of two types: it could be a time step or it could be an  $\epsilon$ -transition step.

We start with the case of a time step. If  $(q, v, I, \mathcal{K}) \rightarrow_t (q', v', I', \mathcal{K}')$  then we have that  $q = q'$ ,  $v' = (v + t)$ ,  $I' = I$ , and  $\mathcal{K}' = \mathcal{K}$ .

Let  $((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w)$  represent the configuration  $(q, v, I, \mathcal{K})$ .

By construction of  $\mathcal{S}$  we have a transition

$$(((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w), In, (0, \dots, 0), ((q, [v+t], (A_1, T_1), \dots, (A_p, T_p)), w))$$

with  $In = \{g(T_1), \dots, g(T_p)\}$ . Since the state  $((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w)$  represents  $(q, v, I, \mathcal{K})$ , then, for any  $i$ , it holds that  $w_{g(T_i)} > 0$ , therefore the step can be taken.

We now prove the case of an  $\epsilon$ -transition step. If  $(q, v, I, \mathcal{K}) \rightarrow_\tau (q', v', I', \mathcal{K}')$  then there exists a transition  $e = (q, \epsilon(\bar{f}(\bar{\mu})), \phi, q')$  such that the following holds:  $I(\tau) = \bar{f}(\bar{\mu})$ ,  $(I \oplus I')$ ,  $(v \oplus v')$ ,  $\mathcal{K} \models \phi$ ,  $\mathcal{K}' = \mathcal{K} \cup \{\bar{f}(\bar{\mu})\}$ .

Let the state  $((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w)$  represent  $(q, v, I, \mathcal{K})$ , and let the state  $((q', [v'], (A'_1, T'_1), \dots, (A'_r, T'_r)), w')$  represent  $(q', v', I', \mathcal{K}')$ . It is now sufficient to prove that there exists a transition from state  $(q, [v], (A_1, T_1), \dots, (A_p, T_p))$  to state  $(q', [v'], (A'_1, T'_1), \dots, (A'_r, T'_r))$  with label  $(In, w'')$  such that  $w_i > 0$  if  $i \in In \cap [1, n]$  and  $w' = w + w''$ . This holds due to the following facts:

1. We prove the relations between the old and the new values of message variables:

Let  $\{A_1'', \dots, A_z''\}$  be a partition of  $\mathcal{Y} \cup \mathcal{Y}'$  representing the instances  $I$  and  $I'$ , namely, for any  $\mu_1, \mu_2 \in \mathcal{Y} \cup \mathcal{Y}'$  it holds that  $(I \oplus I')(\mu_1) = (I \oplus I')(\mu_2)$  iff  $\mu_1, \mu_2 \in A_i''$ , for some  $i$ .

For any  $h$ , let  $T_h'' = \{f \mid f((I \oplus I')(\mu)) \in \mathcal{K} \text{ with } \mu \in A_h''\}$ .

Now, by construction, we have that, for any  $h \in [1, p]$ , there exists  $k$  such that  $A_h = A_k'' \cap \mathcal{Y}$  and  $T_h = T_h''$ . Moreover, for any  $h \in [1, r]$ , there exists  $k$  such that  $A_h' = \{\mu \mid \mu' \in A_k'' \cap \mathcal{Y}'\}$  and, if  $\bar{\mu} \in A_h''$  then  $T_h' = T_k'' \cup \{\bar{f}\}$  otherwise if  $\bar{\mu} \notin A_h''$  then  $T_h' = T_k''$  (we recall that  $\bar{f}(\bar{\mu})$  is the term injected in the knowledge by the  $\epsilon$ -transition).

2. We prove that the guard on the transition is satisfied:

Since  $(I \oplus I'), (v \oplus v'), \mathcal{K} \models \phi$  and since  $((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w)$  represents  $(q, v, I, \mathcal{K})$ , we have that  $\phi$  is true when evaluating:

- each clock  $x$  as  $(v \oplus v')(x)$ ;
- each  $\mu \in \mathbb{N}$  as *true*
- $\mu_1 = \mu_2$  as *true* iff, for some  $h$ , it holds that  $\mu_1, \mu_2 \in A_h''$ ;
- $f(\mu) \in \tilde{\mathcal{K}}$  as *true* iff for some  $h$ , it holds that  $f \in T_h''$  and  $\mu \in A_h''$ .

3. We prove that the  $In$  requirement is satisfied:

since the state  $((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w)$  represents  $(q, v, I, \mathcal{K})$ , it holds that  $w_{g(T_j')} > 0$ , for  $j = 1, \dots, r$ .

4. We prove that  $w' = w + w''$ :

Let  $I(\bar{\mu}) = c$  and  $h$  be such that  $\bar{\mu} \in A_h$ .

If  $\bar{f}(c) \in \mathcal{K}$ , then  $\mathcal{K}' = \mathcal{K}$  and so  $w' = w$ . Hence  $w''$  must be equal to  $(0, \dots, 0)$ .

But since  $\bar{f}(c) \in \mathcal{K}$ , then  $\bar{f} \in T_h$  and so, by construction,  $w'' = (0, \dots, 0)$ .

If  $\bar{f}(c) \notin \mathcal{K}$ , then  $\mathcal{K}' = \{\bar{f}(c)\} \cup \mathcal{K}$ .

Now, to prove that  $w' = w + w''$ , we must prove that, for any  $T \subseteq \Omega$ , it holds that  $w'_{g(T)} = w_{g(T)} + w''_{g(T)}$ .

Hence, let  $T' = \{f' \in \Omega \mid f'(c) \in \mathcal{K}\}$ . For any  $T \subseteq \Omega$ , we have three cases:

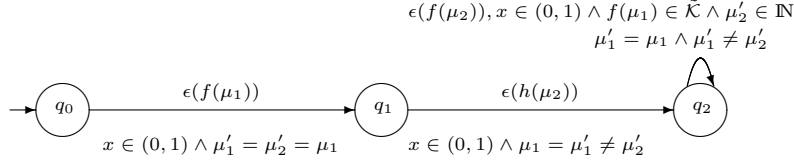
- (a) If  $T \neq T'$  and  $T \neq T' \cup \{\bar{f}\}$ , then, since  $\mathcal{K}' = \{\bar{f}(c)\} \cup \mathcal{K}$ , we have by construction that  $w_{g(T)} = w'_{g(T)}$  and  $w'', w''_{g(T)} = 0$ . Hence, it holds that  $w'_{g(T)} = w_{g(T)} + w''_{g(T)}$ .
- (b) If  $T = T' \cup \{\bar{f}\}$ , then, since  $\mathcal{K}' = \{\bar{f}(c)\} \cup \mathcal{K}$ , we have by construction that  $w_{g(T)} + 1 = w'_{g(T)}$  and  $w'', w''_{g(T)} = 1$ . Thus  $w'_{g(T)} = w_{g(T)} + w''_{g(T)}$ .
- (c) If  $T = T'$ , then, since  $\mathcal{K}' = \{\bar{f}(c)\} \cup \mathcal{K}$ , we have by construction that  $w_{g(T)} - 1 = w'_{g(T)}$  and  $w'', w''_{g(T)} = -1$ . Hence, it holds that  $w'_{g(T)} = w_{g(T)} + w''_{g(T)}$ .

Therefore, we have proved that  $(q, v, I, \mathcal{K}) \rightarrow (q', v', I', \mathcal{K}')$  implies that there exists a step  $((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w) \rightarrow ((q', [v'], (A'_1, T'_1), \dots, (A'_r, T'_r)), w')$  of  $\mathcal{S}$  for some state  $((q, [v], (A_1, T_1), \dots, (A_p, T_p)), w)$  representing  $(q, v, I, \mathcal{K})$  and some state  $((q', [v'], (A'_1, T'_1), \dots, (A'_r, T'_r)), w')$  representing  $(q', v', I', \mathcal{K}')$ . The inverse implication can be proved similarly.

Given the correctness of our construction, in order to complete the proof, we add transitions  $e_1, \dots, e_U$  such that, for each  $i$ ,  $e_i = (q_T, \emptyset, w^i, q_T)$ , where  $w_j^i = -1$  if  $j = i$  and  $w_j^i = 0$  otherwise. The state  $q_T$  represents the state  $\bar{q}$  of  $\mathcal{A}$ , and these new transitions serve the purpose of setting the  $w$  vector to  $(0, \dots, 0)$ . The initial state is  $(q_0, \bigwedge_{x \in X} x = 0, \{\mathcal{T}, \emptyset\})$ .  $\square$

Finally, we have the following result.

**Corollary 1** *Given an 1-SDMTA  $\mathcal{A}$ , it is decidable whether a state  $q$  of  $\mathcal{A}$  is reachable.*



**Fig. 2** Example Reachability.

Such a result is mainly a technical interest, from an applicative point of view, however, since the reduction to VASs is exponential in the size of the 1-SDMTA model, and the reachability problem for VASs is itself EXPSpace hard, the complexity of our decision procedure is a great limit to our methodology. Actually, with such a technique, large 1-SDMTAs are clearly out of the scope of automated verification techniques.

*Example 5* Consider the simple SDMTA of Figure 2 composed by a DMTA  $\mathcal{A}$ .

The first two transitions add to the knowledge  $\{f(0), h(0)\}$  since  $\mu_1$  and  $\mu_2$  are 0 at the beginning.

The loop on  $q_2$  inserts in the knowledge  $f(c')$ , where  $c'$  is the value assigned to  $\mu_2$  by the preceding step. The variable  $\mu_1$  preserves its value in each step.

In Figure 2 we have depicted the VAS simulating  $\mathcal{A}$ .

Now before discussing the states of the VAS, we recall their construction. A state is a tuple  $(q, [v], (A_1, T_1), \dots, (A_k, T_k))$  where  $q$  is the actual state,  $[v]$  is the equivalence class of the valuation we are considering, and,  $(A_i, T_i)$  expresses the instance and the set of basic formulae  $\tau \in \tilde{\mathcal{K}}$  holding. More precisely,  $\mu, \mu' \in A_i$  means that the instance assigns to  $\mu, \mu'$  the same value. Moreover, if  $f \in T_i$  and  $\mu \in A_i$  means that  $f(\mu)$  is in the knowledge. Hence a generic pair  $(A_i, T_i)$  expresses exactly the conjunction of the following constraints:

- $\bigwedge_{\mu \in A_i} \bigwedge_{\mu' \in A_i} \mu = \mu'$
- $\bigwedge_{\mu \in A_i} \bigwedge_{\mu' \notin A_i} \mu \neq \mu'$
- $\bigwedge_{\mu \in A_i} \bigwedge_{g \in T_i} g(\mu) \in \tilde{\mathcal{K}}$ ;
- $\bigwedge_{\mu \in A_i} \bigwedge_{g \notin T_i} \neg g(\mu) \in \tilde{\mathcal{K}}$ .

Therefore, we have considered the four reachable states:

- $(q_0, x = 0, (\{\mu_1, \mu_2\}, \emptyset))$  expressing that the system is in the state  $q_0$  with  $x$  equal to 0. The pair  $(\{\mu_1, \mu_2\}, \emptyset)$  expresses that  $\mu_1$  is equal to  $\mu_2$  (expressed by the set  $\{\mu_1, \mu_2\}$ ), and, that  $f(\mu_1), f(\mu_2), h(\mu_1)$  and  $h(\mu_2)$  are not in the knowledge.
- $(q_1, x \in (0, 1), (\{\mu_1, \mu_2\}, \{f\}))$  expressing that the system is in the state  $q_1$  with  $x$  assuming a value in the interval  $(0, 1)$ . The pair  $(\{\mu_1, \mu_2\}, \{f\})$  expresses that  $\mu_1$  is equal to  $\mu_2$ , that  $f(\mu_1)$  and  $f(\mu_2)$  (that are the same term since  $\mu_1 = \mu_2$ ) are in the knowledge, and that  $h(\mu_1)$  and  $h(\mu_2)$  are not in the knowledge.
- $(q_2, x \in (0, 1), (\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \emptyset))$  expressing that the system is in the state  $q_2$  with  $x$  assuming a value in the interval  $(0, 1)$ . The pairs  $(\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \emptyset)$  express that  $\mu_1$  is different from  $\mu_2$ , that  $f(\mu_1)$  and  $h(\mu_1)$  are in the knowledge, and that  $f(\mu_2)$  and  $h(\mu_2)$  are not in the knowledge.

- $(q_2, x \in (0, 1), (\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \{f\}))$  expressing that the system is in the state  $q_2$  with  $x$  assuming a value in the interval  $(0, 1)$ . The pairs  $(\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \{f\})$  express that  $\mu_1$  is different from  $\mu_2$ , that  $f(\mu_1), f(\mu_2)$  and  $h(\mu_1)$  are in the knowledge, and that  $h(\mu_2)$  is not in the knowledge.

We assign to the set  $\{f\}$  the first position (hence the first coordinate of the vector), to the set  $\{h\}$  the second position (hence the second coordinate of the vector), and, to the set  $\{f, h\}$  the third position (hence the third coordinate of the vector). We recall that:

- the first coordinate of the vector corresponding to the case  $\{f\}$  represents the number of the possible term  $f(c)$  in the knowledge such that  $h(c)$  is not in the knowledge;
- the second coordinate of the vector corresponding to the case  $\{h\}$  represents the number of the possible term  $h(c)$  in the knowledge such that  $f(c)$  is not in the knowledge;
- the third coordinate of the vector corresponding to the case  $\{f, h\}$  represents the number of the possible terms  $f(c)$  and  $h(c)$  such that both  $f(c)$  and  $h(c)$  are in the knowledge.

Therefore, a triple  $(n_1, n_2, n_3)$  means that there exist  $c_1, \dots, c_{n_1+n_2+n_3}$  pairwise different natural number values such that:

- $f(c_i) \in \tilde{\mathcal{K}} \wedge \neg h(c_i) \in \tilde{\mathcal{K}}$  holds iff  $i \in [1, n_1]$ ;
- $h(c_i) \in \tilde{\mathcal{K}} \wedge \neg f(c_i) \in \tilde{\mathcal{K}}$  holds iff  $i \in [n_1 + 1, n_1 + n_2]$ ;
- $f(c_i) \in \tilde{\mathcal{K}} \wedge h(c_i) \in \tilde{\mathcal{K}}$  holds iff  $i \in [n_1 + n_2 + 1, n_1 + n_2 + n_3]$ .

We consider now the transitions of the VAS. We explain the exact construction of the two transitions starting from the state  $(q_2, x \in (0, 1), (\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \emptyset))$  of the VAS. First of all we shall consider all the possible self-loop transitions exiting from state  $(q_2, x \in (0, 1), (\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \emptyset))$ . The only transition in  $\mathcal{A}$  that can generate it is:

$$\epsilon(f(\mu_2)), x \in (0, 1) \wedge f(\mu_1) \in \tilde{\mathcal{K}} \wedge \mu'_2 \in \mathbb{N} \wedge \mu'_1 = \mu_1 \wedge \mu'_1 \neq \mu'_2.$$

As explained in the construction of Theorem 3 first of all we must find a sequence of tuples  $(A_1, T_1) \dots (A_n, T_n)$  such that:

- $A_1, \dots, A_n$  is a partition of  $\{\mu_1, \mu_2, \mu'_1, \mu'_2\}$  and  $T_i \subseteq \{f, h\}$ ;
- the sequence of tuples  $(A_1, T_1) \dots (A_n, T_n)$  restricted to  $\{\mu_1, \mu_2\}$  is equal to the sequence  $(\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \emptyset)$  of the starting state.
- the sequence of tuples  $(A_1, T_1) \dots (A_n, T_n)$  restricted to  $\{\mu'_1, \mu'_2\}$  and by adding  $\{f\}$  to the set related to  $\mu'_1$  (or  $\mu'_2$ ) if  $\mu'_1 = \mu_2$  (or  $\mu'_2 = \mu_2$ ) is equal to the sequence  $(\{\mu'_1\}, \{f, h\}), (\{\mu'_2\}, \emptyset)$  of the target state.
- The condition  $x \in (0, 1) \wedge f(\mu_1) \in \tilde{\mathcal{K}} \wedge \mu'_2 \in \mathbb{N} \wedge \mu'_1 = \mu_1 \wedge \mu'_1 \neq \mu'_2$  of the transition of  $\mathcal{A}$  must be true in the sequence and state considered.

Note that the sequences  $(A_1, T_1) \dots (A_n, T_n)$  are finitely many since  $A_1, \dots, A_n$  must be a partition of  $\{\mu_1, \mu_2, \mu'_1, \mu'_2\}$ . The only sequence satisfying these conditions is the sequence  $(\{\mu_1, \mu'_1\}, \{f, h\}), (\{\mu_2\}, \emptyset), (\{\mu'_2\}, \emptyset)$ .<sup>1</sup>

<sup>1</sup> Note that the sequence  $(\{\mu_1, \mu'_1\}, \{f, h\}), (\{\mu_2, \mu'_2\}, \emptyset)$  is not admitted since, if  $\mu'_2 = \mu_2$ , then in the target state we must have the pair  $(\{\mu'_2\}, \{f\})$ . Actually, after the transition,  $f(\mu_2)$  is in the knowledge.

Now we must compute the label of the transition of the VAS. Firstly, we must construct the set  $In$  for this transition of the extended VAS. As defined in the Theorem 3,  $In$  is the set of coordinates representing the sets  $T_i \neq \emptyset$ . Hence, the source state has the tuple  $(\{\mu_1\}, \{f, h\}, (\{\mu_2\}, \emptyset))$ . The coordinate we have reserved for the set  $\{f, h\}$  is the third (as defined before). Hence,  $In = \{3\}$ .

Finally, we must construct the vector  $w$  for updating the knowledge due to the insertion of term  $f(\mu_2)$ . By Theorem 3, the pair of the starting state considering  $\mu_2$  is  $(\{\mu_2\}, \emptyset)$ . Hence  $f(\mu_2)$  is not in the knowledge (otherwise the pair of the starting state should be  $(\{\mu_2\}, \{f\})$ ). So the coordinate corresponding to  $\{f\}$  (namely the first one) must be increased. The coordinate corresponding to  $\{h\}$  (namely the second one) obviously must not be increased. The coordinate corresponding to  $\{f, h\}$  (namely the third one) must not be increased; actually, for increasing this coordinate we must have that also  $h(\mu_2)$  must be in the knowledge, but this does not hold.

The second transition we can create from  $(q_2, x \in (0, 1), (\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \emptyset))$  is to the state  $(q_2, x \in (0, 1), (\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \{f\}))$ . The only transition in  $\mathcal{A}$  that can generate it is:

$$\epsilon(f(\mu_2)), x \in (0, 1) \wedge f(\mu_1) \in \tilde{\mathcal{K}} \wedge \mu'_2 \in \mathbb{N} \wedge \mu'_1 = \mu_1 \wedge \mu'_1 \neq \mu'_2.$$

Now, we must find a sequence of tuples  $(A_1, T_1) \dots (A_n, T_n)$  such that:

- $A_1, \dots, A_n$  is a partition of  $\{\mu_1, \mu_2, \mu'_1, \mu'_2\}$  and  $T_i \subseteq \{f, h\}$ ;
- the sequence of tuples  $(A_1, T_1) \dots (A_n, T_n)$  restricted to  $\{\mu_1, \mu_2\}$  is equal to the sequence  $(\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \emptyset)$  of the starting state.
- the sequence of tuples  $(A_1, T_1) \dots (A_n, T_n)$  restricted to  $\{\mu'_1, \mu'_2\}$  and by adding  $\{f\}$  to the set related to  $\mu'_1$  (or  $\mu'_2$ ) if  $\mu'_1 = \mu_2$  (or  $\mu'_2 = \mu_2$ ) is equal to the sequence  $(\{\mu'_1\}, \{f, h\}), (\{\mu'_2\}, \{f\})$  of the target state.
- The condition  $x \in (0, 1) \wedge f(\mu_1) \in \tilde{\mathcal{K}} \wedge \mu'_2 \in \mathbb{N} \wedge \mu'_1 = \mu_1 \wedge \mu'_1 \neq \mu'_2$  of the transition of  $\mathcal{A}$  must be true in the sequence and state considered.

The two sequences satisfying these conditions are  $(\{\mu_1, \mu'_1\}, \{f, h\}), (\{\mu_2, \mu'_2\}, \emptyset)$  and  $(\{\mu_1, \mu'_1\}, \{f, h\}), (\{\mu_2\}, \emptyset), (\{\mu'_2\}, \{f\})$ .

The first sequence represents the fact that the new value of  $\mu_2$  is equal to the old one. Hence  $f$  occurs in the knowledge with this value since it is already inserted. The second sequence represents the fact that the new value of  $\mu_2$  is different from the old one but  $f$  occurs with this value in the knowledge for an old insertion.

Obviously, the two sequences (potentially) could generate two transitions (we will see that they generate only one transition).

Let us consider the first sequence  $(\{\mu_1, \mu'_1\}, \{f, h\}), (\{\mu_2, \mu'_2\}, \emptyset)$ . We must construct the set  $In$ , the target state has the tuple  $(\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \{f\})$ . The coordinate we have reserved for the set  $\{f, h\}$  is the third (as defined before). Hence,  $In = \{3\}$ .

Finally we must construct the vector  $w$  for updating the knowledge due to the insertion of term  $f(\mu_2)$ . By Theorem 3, the pair of the starting state considering  $\mu_2$  is  $(\{\mu_2\}, \emptyset)$ . Hence  $f(\mu_2)$  is not in the knowledge (otherwise the pair of the starting state should be  $(\{\mu_2\}, \{f\})$ ). So the coordinate corresponding to  $\{f\}$  (namely the first one) must be increased. The coordinate corresponding to  $\{h\}$  (namely the second one) obviously must not be increased. The coordinate corresponding to  $\{f, h\}$  (namely the third one) must not be increased; actually, for increasing this coordinate we must have that also  $h(\mu_2)$  must be in the knowledge, but this does not hold.

The second sequence  $(\{\mu_1, \mu'_1\}, \{f, h\}), (\{\mu_2\}, \emptyset), (\{\mu'_2\}, \{f\})$  generates the same  $In$  and  $w$  of  $(\{\mu_1, \mu'_1\}, \{f, h\}), (\{\mu_2, \mu'_2\}, \emptyset)$  (and therefore the same transition). Hence there exists only one transition from the state  $(q_2, x \in (0, 1), (\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \emptyset))$  to the state  $(q_2, x \in (0, 1), (\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \{f\}))$ .

This construction can be repeated for all the transitions. Hence we now only explain the origin of all the transitions of the VAS:

- The transition  $\epsilon(f(\mu_1)), x \in (0, 1) \wedge \mu'_1 = \mu'_2 = \mu_1$  of  $\mathcal{A}$  allows to have a transition from  $(q_0, x = 0, (\{\mu_1, \mu_2\}, \emptyset))$  to  $(q_1, x \in (0, 1), (\{\mu_1, \mu_2\}, \{f\}))$  with label  $\{\}, (1, 0, 0)$ .

This is because the original transition add  $f(\mu_1)$  to knowledge. The first coordinate actually is 1 since there exists  $c \in \mathbb{N}$  such that  $f(c)$  is in the knowledge. The condition  $\mu'_1 = \mu'_2 = \mu_1$  ensures that the new values of  $\mu_1$  and  $\mu_2$  are equal to the old values. Therefore  $f(\mu_1) \in \tilde{\mathcal{K}}$  and  $f(\mu_2) \in \tilde{\mathcal{K}}$  are true. Actually the new value is equal to the old one, hence we see the term already added. Moreover,  $h(\mu_1) \in \tilde{\mathcal{K}}$  and  $Know(h(\mu_2))$  are false. This is expressed by the pair  $(\{\mu_1, \mu_2\}, \{f\})$ .

- The transition  $\epsilon(h(\mu_2)), x \in (0, 1) \wedge \mu_1 = \mu'_1 \neq \mu'_2$  of  $\mathcal{A}$  allows to have a transition from  $(q_1, x \in (0, 1), (\{\mu_1, \mu_2\}, \{f\}))$  to  $(q_2, x \in (0, 1), (\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \emptyset))$  with label  $\{1\}, (-1, 1, 0)$ . The reasons are similar to the preceding case.

The index  $\{1\}$  ensures that the pair  $(\{\mu_1, \mu_2\}, \{f\})$  of the source state is coherent with the knowledge. Moreover, the transition decrements the coordinate of  $\{f\}$  (that becomes 0) and increments that of  $\{h, f\}$  (that becomes 1). Actually, before the step we can create an instance for which  $f(\mu_1)$  is in the knowledge and  $h(\mu_1)$  is not, but after the step this is not possible. Actually, after the step we can create an instance for which  $f(\mu_1)$  and  $h(\mu_1)$  are in the knowledge.

- The transition  $\epsilon(f(\mu_2)), x \in (0, 1) \wedge f(\mu_1) \in \tilde{\mathcal{K}} \wedge \mu'_2 \in \mathbb{N} \wedge \mu'_1 = \mu_1 \wedge \mu'_1 \neq \mu'_2$  of  $\mathcal{A}$  creates four transitions of the VAS.

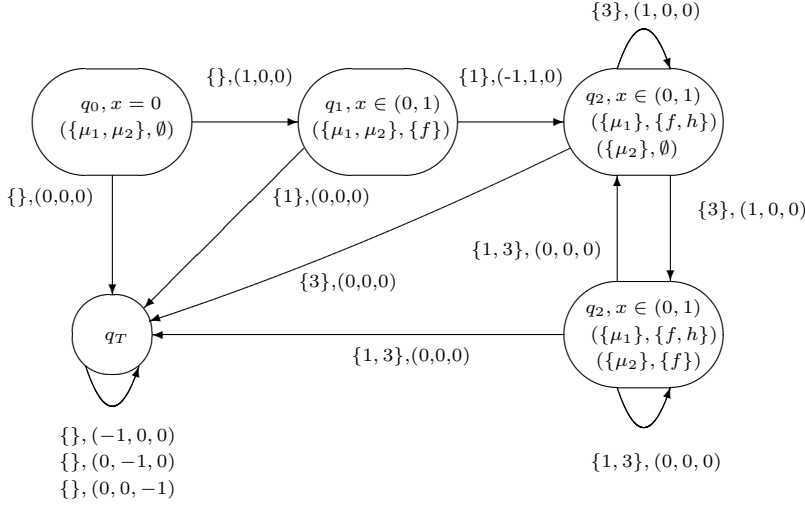
Actually,  $\mu'_2 \in \mathbb{N}$  assigns non deterministically a value to  $\mu_2$ . The condition  $\mu'_1 = \mu_1 \wedge \mu'_1 \neq \mu'_2$  ensures that the new value  $\mu_2$  is different from the value of  $\mu_1$  that remains invariant. Hence, in the two states labeled with  $q_3$ ,  $\mu_1 \in \tilde{\mathcal{K}}$  and  $h(\mu_1) \in \tilde{\mathcal{K}}$  are true.

Now, we have two states since the new value assigned to  $\mu_2$  could be equal to some value  $c$  such that either  $f(c) \in \tilde{\mathcal{K}} \wedge \neg h(c) \in \tilde{\mathcal{K}}$  holds or  $f(c) \in \tilde{\mathcal{K}} \wedge h(c) \in \tilde{\mathcal{K}}$  holds. Actually, we have two cases  $(\{\mu_2\}, \{f\})$  and  $(\{\mu_2\}, \emptyset)$ . Therefore, when  $(\{\mu_2\}, \{f\})$  holds, the transition adds a term that is in the knowledge. Hence the knowledge is not incremented and, therefore, the transition is labeled with  $\{1, 3\}, (0, 0, 0)$  (obviously,  $\{1, 3\}$  ensures that  $(\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \{f\})$  is coherent with the knowledge).

Otherwise, when  $(\{\mu_2\}, \{\})$  holds, the transition adds a term that is not in the knowledge. Hence the knowledge is incremented and, therefore, the transition is labeled with  $\{3\}, (1, 0, 0)$  (obviously,  $\{3\}$  ensures that  $(\{\mu_1\}, \{f, h\})$  is coherent with the knowledge).

From state  $(q_1, x \in (0, 1), (\{\mu_1, \mu_2\}, \{f\}))$  we have a transition reaching only  $(q_2, x \in (0, 1), (\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \emptyset))$  since we cannot have an instance such that  $f(\mu_2) \in \tilde{\mathcal{K}} \wedge \neg h(\mu_2) \in \tilde{\mathcal{K}}$ .

Finally we introduce the state  $q_T$  representing the ending of the computation. As proved in Theorem 3, from a state representing a state of  $\mathcal{A}$  a transition starts leading to  $q_T$  and checking the validity of the tuple of the starting state. As an example, from  $(q_2, x \in (0, 1), (\{\mu_1\}, \{f, h\}), (\{\mu_2\}, \{f\}))$  a transition starts leading to  $q_T$  with label



**Fig. 3** Example Reachability VAS.

$\{1, 3\}$ , since the coordinates associated with  $\{f\}$  and  $\{f, h\}$  are 1 and 3, respectively. From state  $q_T$  only transitions with the purpose of putting the coordinates to 0 are allowed.

Therefore, we reach configurations of the form  $(s, (0, 1, n))$ , for some state  $s$ , representing the fact that the knowledge is equal to  $\{f(c_0), h(c_0), f(c_1), \dots, f(c_n)\}$  with  $c_i \in \mathbb{N}$  and  $c_i \neq c_j$ , for any  $i \neq j$ .

### 3.5 Expressiveness of SDMTAs and Comparisons with Other Models

In this section we discuss the expressive power of SDMTAs. To this purpose we introduce a notion of accepted run with respect to a set of states and a notion of language. We suppose that  $\mathcal{A}$  has an associated set  $F \subseteq \bigcup_{i \in [1, m]} Q_i$  of *final states*.

Given a run  $r = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_l} s_l$ , with  $word(r)$  we denote the string  $a_1 \dots a_l$  such that, for any  $i$ , either  $\alpha_i = a_i$  and  $\alpha_i \in \mathbb{R}^{\geq 0}$  or there exists  $\tau$  such that  $\alpha_i = a_i(\tau)$ . We say that  $r$  is *accepted by  $\mathcal{A}$*  if the states appearing in  $s_l$  are contained in  $F$ . With  $\mathcal{L}(\mathcal{A})$  we denote the set  $\{word(r) \mid r \text{ is accepted by } \mathcal{A}\}$ .

Given a string  $z = a_1 \dots a_m$ , with  $untime(z)$  we denote the string  $a'_1 \dots a'_m$  such that  $a'_i = \epsilon$  if  $a_i \in \mathbb{R}^{\geq 0}$  and  $a'_i = a_i$  otherwise.

Given a language  $L$ , with  $Untime(L)$  we denote the set  $\{untime(z) \mid z \in L\}$ .

The following Proposition states that the class of 2-SDMTAs (and hence each class of  $n$ -SDMTAs with  $n \geq 2$ ) is able to simulate a Turing machine.

**Proposition 3** *For any Turing Machine  $\mathcal{M}$ , there exists a 2-SDMTA  $\mathcal{A}$  such that  $Untime(\mathcal{L}(\mathcal{A})) = \mathcal{L}(\mathcal{M})$ .*

**Proof.** Let us discuss how to model the tape of a Turing Machine with symbols in  $\Sigma$ . We can use a function of the form  $node(p_1, a, p_2)$ , where  $p_1$  and  $p_2$  are natural numbers

and  $c$  is a symbol. The natural numbers  $p_1$  and  $p_2$  can be used as pointers and the null pointer is modeled by 0. We use two special symbols *start* and *end* for marking the beginning and the end of the sequence. Hence a sequence  $a_1 \dots a_k$  is modeled with the terms

$$\text{node}(0, \text{start}, p_0), \text{node}(p_0, a_1, p_1), \dots, \text{node}(p_{k-1}, a_k, p_k), \text{node}(p_k, \text{end}, 0)$$

in the knowledge.

The presence of the term  $\text{deleted}(p_1, a, p_2)$  in the knowledge means that the node  $\text{node}(p_1, a, p_2)$  has been changed. This is because the pointers  $p_1$  and  $p_2$  can be used only once.

To operate on the sequence, it is sufficient to store the value  $p_h$  of the actual node in a message variable  $\mu_1$  and the actual symbol in a message variable  $\mu_2$ . Hence, one can easily move on the tape with a sequence of steps. As an example, to move to the left is sufficient to use the following condition

$$\mu_1 \neq \text{start} \wedge \mu'_1 \in \mathbb{N} \wedge \mu'_2 \in \Sigma \wedge \bigvee_{\mu'_2 \in \Sigma} (\text{node}(\mu', \mu'_2, \mu) \in \tilde{\mathcal{K}} \wedge \text{deleted}(\mu', \mu'_2, \mu) \in \tilde{\mathcal{K}}).$$

The condition assigns non deterministically a new pointer value to  $\mu_1$  and a new symbol value to  $\mu_2$  such that the new value of  $\mu_1$  is the previous active pointer (namely the previous pointer that has not been deleted).

Given the sequence  $\text{node}(0, \text{start}, p_0) \dots \text{node}(p_{k-1}, a_k, p_k) \dots \text{node}(p_k, \text{end}, 0)$ , in order to change  $a_h$  into  $a'$ , it is sufficient to mark the nodes  $(p_{h-2}, a_{h-1}, p_{h-1})$  and  $(p_{h-1}, a_h, p_h)$  as deleted (by adding  $\text{deleted}(p_{h-2}, a_{h-1}, p_{h-1})$  and  $\text{deleted}(p_{h-1}, a_h, p_h)$  in the knowledge) and to insert two new nodes  $(p_{h-2}, a_{h-1}, p')$  and  $(p', a', p_h)$ , where  $p'$  is a fresh pointer (namely,  $\neg \text{node}(p_{h-2}, b, p') \in \tilde{\mathcal{K}} \wedge \text{node}(p', b, p_h) \in \tilde{\mathcal{K}}$ , for any symbol  $b$ ).

Similarly, we can add a node at the end of the sequence.  $\square$

In the previous section we proved that the reachability problem for the class of 1-SDMTAs is decidable. This implies that the class of 1-SDMTAs is not Turing complete. However, we prove that 1-SDMTAs recognize also non regular languages.

**Proposition 4** *There exists an 1-SDMTA  $\mathcal{A}$  such that  $\text{Untime}(\mathcal{L}(\mathcal{A})) = \{a^k b^h c^m \mid k \geq h \geq m\}$ .*

**Proof.** In figure 4 we give  $\mathcal{A}$  such that  $\text{Untime}(\mathcal{L}(\mathcal{A})) = \{a^k b^h c^m \mid k \geq h \geq m\}$ .

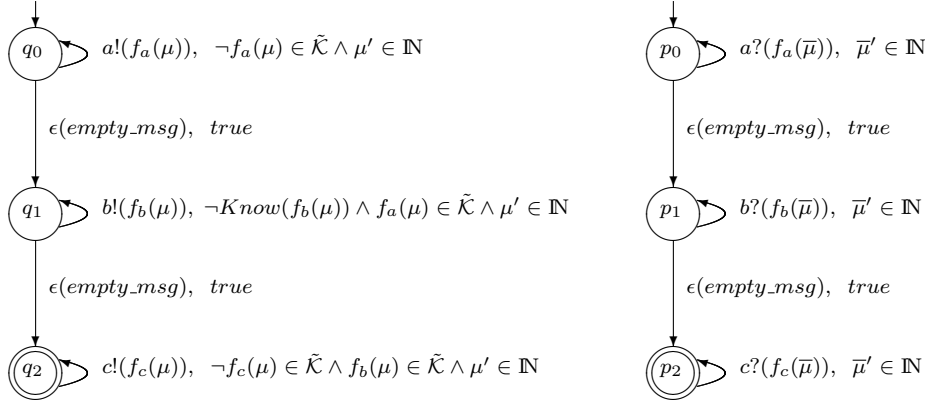
Note that the second component just receives the messages sent by the first one.

In state  $q_0$ , the first component generates  $a^k$ . After the execution of the  $\epsilon$ -transition from  $q_0$  to  $q_1$  the knowledge becomes  $\{f_a(c_1), \dots, f_a(c_k)\}$  for some natural numbers  $c_i$  such that  $c_i \neq c_j$  if  $i \neq j$ .

In state  $q_1$ ,  $\mathcal{A}$  generates  $b^h$ . We have that  $h \leq k$  since we require that  $f_a(\mu) \in \tilde{\mathcal{K}}$ . After executing the  $\epsilon$ -transition from  $q_1$  to  $q_2$ , the knowledge contains  $\{f_b(c'_1), \dots, f_b(c'_h)\}$  for some natural numbers  $c'_i$  such that  $c'_i \neq c'_j$  if  $i \neq j$ .

Finally, in state  $q_2$ ,  $\mathcal{A}$  generates  $c^m$ . We have that  $m \leq h$  since we require that  $\text{Know}(f_b(\mu))$ .  $\square$

Now, the language  $\{a^k b^h c^m \mid k \geq h \geq m\}$  is not regular, while in [4] it is proven that, if  $L$  is a language recognized by a Timed Automaton, then  $\text{Untime}(L)$  is a regular language. Therefore, by Proposition 4, and since a Timed Automaton can be easily translated into a 0-SDMTA and viceversa, we have the following result.



**Fig. 4** 1-SDMTA for  $\{a^k b^h c^m \mid k \geq h \geq m\}$ .

**Corollary 2** 1-SMDTAs are more expressive than 0-SDMTAs which are equivalent to Timed Automata.

As a consequence, also the model presented in [21], which is equivalent to the class of Timed Automata, is less expressive than 1-SDMTAs.

Let us now consider Timed Petri Nets (see [7] for a survey). In the original definition of Petri nets there is no notion of accepted language, hence, we compare our model of SDMTAs and Timed Petri Nets w.r.t. decidability of the reachability problem.

In *Timed Petri Nets* (TPNs) (see [6]) a transition can be fired if its enabling duration lies in its interval and time can elapse only if it does not disable some transition: firing an enabled transition may depend on other enabled transitions even if they do not share any input or output place, this restricts a lot the applicability of partial order methods in this model. Moreover, because of such an "urgency" requirement, all significant problems become undecidable for unbounded TPNs. On the other hand it is proved that the bounded version is bisimilar to Timed Automata and hence to 0-SDMTAs.

Timed-arc Petri Nets (see [24]) associate with each arc an interval (or bag of intervals). Each token has an age. This age is initially set to a value belonging to the interval of the arc which has produced it or set to zero if it belongs to the initial marking. Afterwards, ages of tokens evolve synchronously with time. A transition may be fired if tokens with age belonging to the intervals of its input arcs may be found in the current configuration. Old tokens cannot be used anymore for firing a transition but they remain in the place. The reachability problem is undecidable also for this class (see [24]).

If one is interested in an untimed version of SDMTAs, we can compare our model w.r.t. Petri Nets. Actually, Petri Nets are equivalent to VASs. A VAS can simulate the untimed version of a SDMTA (this comes from the proof of Theorem 3). Conversely, a VAS  $\mathcal{S}$  can be easily simulated by an SDMTA. Intuitively, for each coordinate  $i$  of

the vector  $w$  of  $\mathcal{S}$  we can define two functions  $in_i$  and  $del_i$ . The value  $w_i$  is equal to the occurrences of  $in_i$  minus the occurrences of  $del_i$  within the knowledge  $\mathcal{K}$ , more precisely:

$$w_i = |\{in_i(c) \mid c \in \mathbb{N} \text{ and } in_i(c) \in \mathcal{K}\}| - |\{del_i(c) \mid c \in \mathbb{N} \text{ and } in(c) \in \mathcal{K}\}|.$$

Hence, to increment the value of the  $i^{th}$  coordinate, it is sufficient to add a new fresh term  $in_i(c)$  (i.e.  $in_i(c) \notin \mathcal{K}$ ), while, to decrement the value of the  $i^{th}$  coordinate, it is sufficient to add a term  $del_i(c)$  that deletes an occurrence of  $in_i(c)$  (i.e.  $in_i(c) \in \mathcal{K}$  and  $del_i(c) \notin \mathcal{K}$ ). To check whether  $w_i \geq 0$ , it is sufficient to check whether there exists  $c$  such that  $in_i(c) \in \mathcal{K}$  and  $del_i(c) \notin \mathcal{K}$ .

#### 4 Modelling Cryptographic Protocols with SDMTAs

Security protocols are sensitive to the passage of time since time aspects can influence the flow of messages during the execution of a protocol.

On the one hand, timeouts may be integrated within the protocol and retransmissions or other behaviours can be considered. In this case, the specification of the protocol should model these implementation details. On the other hand, time information can be used within a protocol in order to enrich the information contained in a message. This kind of application may be very useful, for example, when generating *timestamps*. Finally, one should also be aware that the timing of the message flow may be exploited by an adversary to violate the security of the protocol.

In this section we define an instantiation of SDMTAs to the case of cryptographic protocols (CSDMTAs). We formalize intruder's capabilities and we give the definitions of secrecy and authentication properties for cryptographic protocols within this framework. Then we study the Yahalom protocol by modelling each principal of the protocol and the intruder with DMTAs. The execution of the protocol is modeled by the resulting CSDMTA.

Consider  $C = \{A, M, K\}$ , where  $A = \{a, b, \dots\}$  is a set of agent names,  $M = \{m_1, m_2, \dots\}$  is a set of basic messages (i.e. a set of plaintext messages represented by bitstrings of a fixed length) and  $K = \{k_1, k_2, \dots\}$  is a set of keys. Given a finite set of message variables  $\mathcal{Y}$ , we assume  $\Omega = \{Pair, Enc, Nonce\}$ , with domains  $\mathcal{T}(\mathcal{Y}) \times \mathcal{T}(\mathcal{Y})$ ,  $\mathcal{T}(\mathcal{Y}) \times K$  and  $(A \cup \mathcal{Y}) \times (\mathcal{Y} \cup \mathbb{N})$ , respectively.  $Pair(\tau_1, \tau_2)$  denotes the concatenation of the terms  $\tau_1$  and  $\tau_2$ ,  $Enc(m_1, k_1)$  denotes the encryption of message  $m_1$  with the key  $k_1$ , and  $Nonce(a, 100)$  denotes a nonce of the agent  $a$  with value 100.  $Nonce(\mu_1, \mu_2)$ , where  $\mu_1, \mu_2 \in \mathcal{Y}$  are variables, may be instantiated by  $Nonce(a, w)$  for some  $a \in A$  and  $w \in \mathbb{N}$ .  $Enc(\mu, k)$ , where  $\mu$  is a variable can be instantiated by  $Enc(m, k)$  for some  $m \in M$ , by  $Enc(a, k)$  for some  $a \in A$ , by  $Enc(k, k)$  for some  $k \in K$  or by  $Enc(w, k)$  for some  $w \in \mathbb{N}$ . We remark that  $\mu_1, \mu_2, \mu$  cannot be complex terms, i.e.  $Enc(Nonce(a, 1), k)$  or  $Enc(Enc(m', k'), k)$  are not instances of  $Enc(\mu, k)$ .

**Definition 4** A Cryptographic SDMTA (CSDMTA) is an SDMTA expressed by  $\mathcal{AC} = (C, \Omega, A_1, \dots, A_m, I)$  where DMTAs  $A_i$  model the principals in the protocol, and  $I$  is the DMTA modelling the intruder.

In the following, we give a formalization of a classical Dolev–Yao style intruder [13]. That is, the intruder has a complete control over the network (considered on public

channels) and he can derive new messages from the initial knowledge and the messages received from the honest principals during the protocol runs. To derive a new message, the intruder can compose and decompose pairs of message and encrypt and decrypt messages, in case he knows the key.

**Definition 5** Given a CSDMTA  $\mathcal{AC} = (C, \Omega, A_1, \dots, A_m, I)$ , where DMTAs  $A_i = (\Sigma^i, X^i, \mathcal{Y}^i, Q^i, q_0^i, \delta^i)$  model the principals in a cryptographic protocol, we say that  $I = (\Sigma, X, \mathcal{Y}, Q, q_0, \delta)$  is the intruder for  $\mathcal{AC}$  iff:

- $\Sigma = \cup_{i=1}^m \Sigma^i$ ;
- $X = \{x\}$ ;
- $Q = \{q_0\}$ ;
- $(q_0, \epsilon(), x' = 0, q_0) \in \delta'$ ;
- for all  $i \in [1, m]$ , if  $(q, a!(\tau), \phi, q') \in \delta^i$  or  $(q, a?(\tau), \phi, q') \in \delta^i$ , then  $(q_0, a?(\tau'), x' = 0, q_0)$  and  $(q_0, a!(\tau'), \tau' \in \mathcal{K} \wedge x' = 0, q_0)$  are in  $\delta$ , where  $\tau \simeq \tau'$ ;
- (Extracting a term from a pair) for all  $i \in [1, m]$ , if  $Pair(\tau_1, \tau_2)$  is a subformula of  $\tau$  with  $(q, a!(\tau), \phi, q') \in \delta^i$ , then both  $(q_0, \epsilon(\tau'_1), Pair(\tau'_1, \tau'_2) \in \mathcal{K} \wedge x \geq t_u \wedge x' = 0, q_0)$  and  $(q_0, \epsilon(\tau'_2), Pair(\tau'_1, \tau'_2) \in \mathcal{K} \wedge x \geq t_u \wedge x' = 0, q_0)$  are in  $\delta$ , where  $\tau_1 \simeq \tau'_1$ ,  $\tau_2 \simeq \tau'_2$  and  $t_u$  is a constant;
- (Pairing two terms) for all  $i \in [1, m]$ , if  $Pair(\tau_1, \tau_2)$  is a subformula of  $\tau$  with  $(q, a?(\tau), \phi, q') \in \delta^i$ , then  $(q_0, \epsilon(Pair(\tau'_1, \tau'_2)), \tau'_1 \in \mathcal{K} \wedge \tau'_2 \in \mathcal{K} \wedge x \geq t_p \wedge x' = 0, q_0) \in \delta$ , where  $\tau_1 \simeq \tau'_1 \wedge \tau_2 \simeq \tau'_2$  and  $t_p$  is a constant;
- (Deciphering an encrypted term) for all  $i \in [1, m]$ , if  $Enc(\tau_1, \tau_2)$  is a subformula of  $\tau$  with  $(q, a!(\tau), \phi, q') \in \delta^i$ , then  $(q_0, \epsilon(\tau'_1), Enc(\tau'_1, \tau'_2) \in \mathcal{K} \wedge \tau'_2 \in \mathcal{K} \wedge x \geq t_d \wedge x' = 0, q_0) \in \delta$ , where  $\tau_1 \simeq \tau'_1$ ,  $\tau_2 \simeq \tau'_2$  and  $t_d$  is a constant;
- (Encrypting a term) for all  $i \in [1, m]$ , if  $Enc(\tau_1, \tau_2)$  is a subformula of  $\tau$  with  $(q, a?(\tau), \phi, q') \in \delta^i$ , then  $(q_0, \epsilon(Enc(\tau'_1, \tau'_2)), \tau'_1 \in \mathcal{K} \wedge \tau'_2 \in \mathcal{K} \wedge x \geq t_e \wedge x' = 0, q_0) \in \delta$ , where  $\tau_1 \simeq \tau'_1$ ,  $\tau_2 \simeq \tau'_2$  and  $t_e$  is a constant.

With such a definition of a single state intruder, we assume that principals of the protocol use only public channels, therefore the intruder may intercept each message exchanged within the network and spread any information. This is modeled by taking  $\Sigma$  as the set containing all channel names appearing in the automata  $A_i$  and by adding an input transition to the intruder for any output transitions of any  $A_i$ . Conversely, an output transition is added to the intruder for any input transition of any  $A_i$ . Moreover, we give the intruder classical Dolev-Yao capabilities such as extraction of a term from a pair, pairing of two terms, and encryption and decryption of terms, assuming that the key is known. Note that, according to some given algorithms, this operations may require some amount of time to be performed. Therefore, for each of those transitions, we require that the value of the clock  $x$  is equal or greater than the constants  $t_u$ ,  $t_p$ ,  $t_d$  and  $t_e$  representing, respectively, the time needed for extracting a term from a pair, for pairing two terms, for deciphering an encrypted term and for encrypting a term.

Finally, with the empty  $\epsilon$ -transition the intruder may nondeterministically instantiate variables by modifying their actual instances. Note that by performing such a transition no messages are added to the knowledge.

#### 4.1 Security Properties

Many security properties may be defined in order to analyze cryptographic protocols (among them, *secrecy*, *authentication*, *integrity*, *fairness*, *non-repudiation*, etc.).

In this paper we focus on the definition of secrecy and authentication properties for our framework. Intuitively, a term is secret to a principal if it never appears within its knowledge (see [1,14]), while a principal  $A$  truly authenticates to a principal  $B$  if whenever  $B$  thinks of communicate with  $A$ ,  $B$  is really communicating with  $A$ .

For specifying and verifying the secrecy property, we require that a term  $\tau$ , that should be kept secret to a set of principals  $P$ , does never appear within the knowledge of a principal in the set  $P$ .

**Definition 6** Given a CSDMTA  $\mathcal{AC} = (C, \Omega, A_1, \dots, A_m, I)$ , a term  $\tau$  with  $\text{Var}(\tau) = \emptyset$  and a set of principals  $P \subseteq \{A_1, \dots, A_m, I\}$ , we say that  $\tau$  is  $P$ -secret for any principal in  $P$  iff there exists no run  $\sigma = s^0 \rightarrow_{\alpha_1} s^1 \rightarrow_{\alpha_2} \dots \rightarrow_{\alpha_l} s^l$  of  $\mathcal{AC}$ , with  $s^i = (s_{A_1}, \dots, s_{A_m}, s_I)$  for some  $i \in [0, l]$  and  $s_j = (q, I, v, \mathcal{K})$  for some  $j \in P$  such that  $\tau \in \mathcal{K}$ .

**Proposition 5** Given a CSDMTA  $\mathcal{AC} = (C, \Omega, A_1, \dots, A_m, I)$ , it is decidable whether a term  $\tau$  with  $\text{Var}(\tau) = \emptyset$  is  $P$ -secret for a set of principals  $P \subseteq \{A_1, \dots, A_m, I\}$ .

**Proof.** We build the CSDMTA  $\mathcal{AC}' = (C, \Omega, A'_1, \dots, A'_m, I')$  such that for each DMTA  $A \notin P$ ,  $A' = A$  and for each  $A = (\Sigma, X, \mathcal{T}, Q, q_0, \delta) \in P$ ,  $A' = (\Sigma, X, \mathcal{T}, Q \cup q_s, q_0, \delta \cup \delta_s)$ , where  $q_s \notin Q$  and  $\delta_s = \delta \cup \{(q, \epsilon(\tau), \tau \in \mathcal{K}, q_s) \mid q \in Q\}$ . With such a construction, for each DMTA in the set  $P$ , we have new transitions reaching the special state  $q_s$  whenever the term to be kept secret appears in the knowledge of the DMTA (in any state of the DMTA we might fire the transition guarded by  $\tau \in \mathcal{K}$ ). Therefore, we can say that the term  $\tau$  is  $P$ -secret for  $\mathcal{AC}$  if none of the DMTAs in  $P$  reaches the special state  $q_s$ . Since checking state reachability is decidable (Corollary 1), also  $P$ -secrecy for the CSDMTA  $\mathcal{AC}'$  is decidable. By construction,  $\tau$  is  $P$ -secret for  $\mathcal{AC}'$  if and only if  $\tau$  is  $P$ -secret for  $\mathcal{AC}$ .  $\square$

In the literature several authentication flaws are defined (see, for example, [11,1,20]). The authentication failure that we will define in our framework is inspired by Lowe [20]. For specifying and verifying the authentication property, we require that whenever the principal  $B$  ends a communication protocol with the principal  $A$ , then  $A$  had previously started the communication protocol.

In order to formally define the authentication property for the model of CSDMTAs, we may assume that, given a CSDMTA  $\mathcal{AC} = (C, \Omega, A_1, \dots, A_m, I)$  and two principals  $A_i, A_j$ , whenever  $A_i$  starts a communication protocol with  $A_j$ , it reaches the state  $q_{S_{ij}}$ , and that whenever  $A_j$  ends the protocol with  $A_i$ , it reaches the state  $q_{E_{ij}}$ . This property can be transposed within our model by requiring that for each run of a CSDMTA, whenever the state  $q_{E_{ij}}$  is reached, then the state  $q_{S_{ij}}$  was previously crossed. Intuitively, this means that any closing event  $q_{E_{ij}}$  is preceded by a starting event  $q_{S_{ij}}$ . In fact, if  $A_j$  reaches the state  $q_{E_{ij}}$  without  $A_i$  reaching state  $q_{S_{ij}}$ , then it means that another principal, simulating to be  $A_i$ , started and performed a complete session of the communication protocol with  $A_j$ .

**Definition 7** Given a CSDMTA  $\mathcal{AC} = (C, \Omega, A_1, \dots, A_m, I)$ , and two principals  $A_i, A_j$ , we say that the principal  $A_i$  truly authenticates to principal  $A_j$ , if there exists no run  $\sigma = s^0 \rightarrow_{\alpha_1} s^1 \rightarrow_{\alpha_2} \dots \rightarrow_{\alpha_l} s^l$  of  $\mathcal{AC}$ , with configurations  $s^k = (s_{A_1}^k, \dots, s_{A_m}^k, s_I^k)$  for  $k \in [0, l]$ , such that  $s_{A_j}^p = (q_{E_{ij}}, I^p, v^p, \mathcal{K}^p)$  and  $s_{A_j}^r = s_{A_j}^0$  or  $s_{A_j}^r = (q_{E_{ij}}, I^r, v^r, \mathcal{K}^r)$  for some  $p, r$  with  $r < p$  and  $s_{A_i}^q \neq (q_{S_{ij}}, I^q, v^q, \mathcal{K}^q)$  for all  $q \in [r, p]$ .

**Proposition 6** Given a CSDMTA  $\mathcal{AC} = (C, \Omega, A_1, \dots, A_m, I)$ , it is decidable whether a principal  $A_i$  truly authenticates to principal  $A_j$ .

**Proof.** We build the CSDMTA  $\mathcal{AC}' = (C, \Omega, A_1, \dots, A'_i, \dots, A'_j, \dots, A_m, O, I)$ . On the one hand, given  $A_i = (\Sigma, X, \Upsilon, Q, q_0, \delta)$ , we have  $A'_i = (\Sigma, X, \Upsilon, Q \cup q_{run}, q_0, \delta')$ , where  $q_{run} \notin Q$  and  $\delta'$  is obtained by replacing each transition  $(q_{S_{ij}}, \alpha, \phi, q) \in \delta$  with the two transitions  $(q_{S_{ij}}, run!(), \bigwedge_{x \in X} x' = x \wedge \bigwedge_{\mu \in \Upsilon} \mu' = \mu, q_{run})$  and  $(q_{run}, \alpha, \phi, q)$ . On the other hand, given  $A_j = (\Sigma, X, \Upsilon, Q, q_0, \delta)$ , we have  $A'_j = (\Sigma, X, \Upsilon, Q \cup q_{com}, q_0, \delta')$ , where  $q_{com} \notin Q$  and  $\delta'$  is obtained by replacing each transition  $(q_{E_{ij}}, \alpha, \phi, q) \in \delta$  with the two transitions  $(q_{E_{ij}}, commit!(), \bigwedge_{x \in X} x' = x \wedge \bigwedge_{\mu \in \Upsilon} \mu' = \mu, q_{com})$  and  $(q_{com}, \alpha, \phi, q)$ . We assume that *run* and *commit* are special unused labels. Therefore, every time the DMTA  $A_i$  starts a run of the communication protocol it performs a transition with label *run*, while every time the DMTA  $A_j$  commits the ending of the communication it performs a transition with label *commit*.

Now, the new principal  $O$  has the task of observing the *run* and the *commit* transitions performed by  $A_i$  and  $A_j$ . More formally, the principal  $O$  is defined as  $O = (\{com, run\}, \emptyset, \emptyset, \{o_0, o_1, o_{AutF}\}, o_0, true, \emptyset, \delta_0)$  where the set of transitions  $\delta_0$  is composed by  $(o_0, run?(), true, o_1)$ ,  $(o_0, commit?(), true, o_{AutF})$ ,  $(o_1, run?(), true, o_1)$  and  $(o_1, commit?(), true, o_0)$ .

Hence, if a *commit* transition is performed without being preceded by a *run* transition (thus violating the authentication property), the observer  $O$  will reach state  $o_{AutF}$ .

Therefore, we can say that, given the protocol modelled by  $\mathcal{AC}$ , the principal  $A_i$  truly authenticates to the principal  $A_j$  if the state  $o_{AutF}$  can not be reached in the CSDMTA  $\mathcal{AC}'$ . Thus, by the decidability of state reachability given in Corollary 1, also the authentication property is decidable.  $\square$

Now, we can use the framework of CSDMTAs to model and analyze a real cryptographic protocol in a timed setting.

#### 4.2 The Yahalom Protocol

The Yahalom protocol [9] is designed for the distribution of a fresh symmetric key shared between two users. The protocol resorts to a trusted server, and each user is assumed to share a symmetric key with the server. Here we consider a strengthened version of the Yahalom protocol proposed by Paulson in [22]. In the standard protocol notation, we denote with  $A \rightarrow B : Msg$  a message  $Msg$  sent by  $A$  and received by  $B$ , with  $\{Msg\}_K$  we denote the encryption of the message  $Msg$  under the key  $K$ . If  $A$ ,  $B$  and  $S$  are the principals of the protocols (with  $S$  the trusted server),  $Na$  and  $Nb$  are fresh nonces and  $Kas$ ,  $Kbs$  and  $Kab$  are symmetric keys shared by the principals in the subscript, the Yahalom protocol can be described as follows:

1.  $A \rightarrow B : A, Na$
2.  $B \rightarrow S : B, Nb, \{A, Na\}_{Kbs}$
3.  $S \rightarrow A : Nb, \{B, Kab, Na\}_{Kas}, \{A, B, Kab, Nb\}_{Kbs}$
4.  $A \rightarrow B : \{A, B, Kab, Nb\}_{Kbs}, \{Nb\}_{Kab}$

It is also assumed that principal  $A$  only knows elements in the set  $\{A, B, S, Kas\}$ , the knowledge of principal  $B$  is given by  $\{B, S, Kbs\}$ , the trusted server  $S$  knows  $\{A, B, Kas, Kbs\}$ .

User  $A$  starts the protocol by communicating to  $B$  its intention to share a new session key with it (step 1). User  $B$  generates a fresh nonce  $Nb$  and creates a new term containing the identity of  $A$  and its nonce  $Na$  encrypted with the key  $Kbs$  shared with

the trusted server. User  $B$  sends its identity, the fresh nonce  $Nb$  and the encrypted term to the server (step 2). Server  $S$  deciphers the encrypted term, obtains the identity of  $A$  and generates a new fresh key  $Kab$ . It also builds two terms encrypted with  $Kas$  and  $Kbs$ , and sends the whole message to  $A$  (step 3). Finally,  $A$  deciphers the first encryption and checks whether it contains nonce  $Na$ . If this is the case,  $A$  sends to  $B$  the second term encrypted with  $Kbs$  and mutually authenticates to  $B$  by sending nonce  $Nb$  encrypted with the fresh session key (step 4).

The basic requirements that this protocol must satisfy are the secrecy of the key  $Kab$  (in every session, the value of  $Kab$  must be known only by the participants playing the roles of  $A$ ,  $B$  and  $S$ ) and a proper authentication of the principal  $A$  to the principal  $B$ .

In a timed setting, the protocol can be adapted to take timeouts and retransmissions into account. In step 1, after sending its message,  $A$  may start a timer while waiting for the message of step 3. If the timeout occurs,  $A$  may retransmit its message (even the same nonce  $Na$  can be resent as it was already sent in clear). As some amount of time must pass while  $B$  and  $S$  receive and elaborate messages, if  $A$  receives an answer too early, this might signal some misbehaviour as, for example, the reception of a faked message from the intruder. Therefore, if we suppose that  $A$  knows the encryption and decryption times of  $B$  and  $S$ , we might adapt the Yahalom protocol to take time into account. In a formalism where time aspects are not considered this would not be possible.

#### 4.3 Specification of the Yahalom Protocol through SDMTAs

We model the protocol execution through a CSDMTA  $\mathcal{AC} = (C, \Omega, A, B, S, I)$ , where  $A$ ,  $B$  and  $S$  are the DMTAs representing the principal  $A$ , the principal  $B$  and the trusted server  $S$ , respectively, while  $I$  is the intruder as specified in Definition 5.

The principals in the protocol are specified by the following DMTAs, graphically represented in Figure 5:

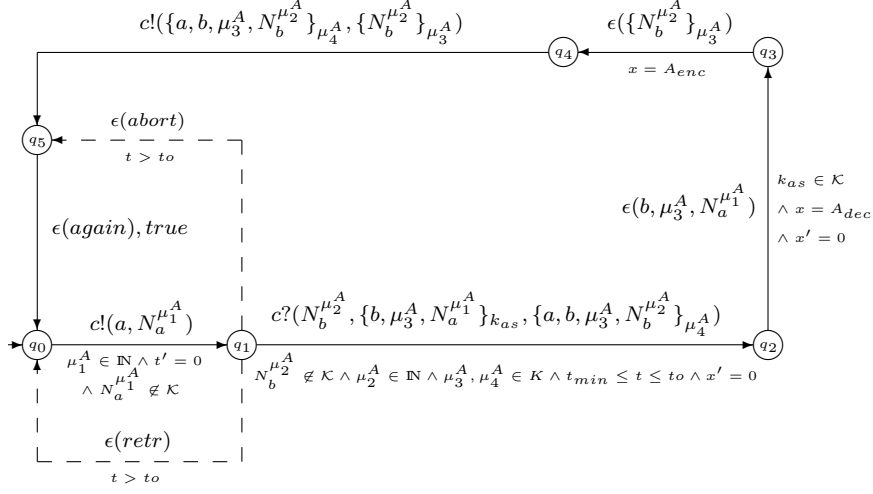
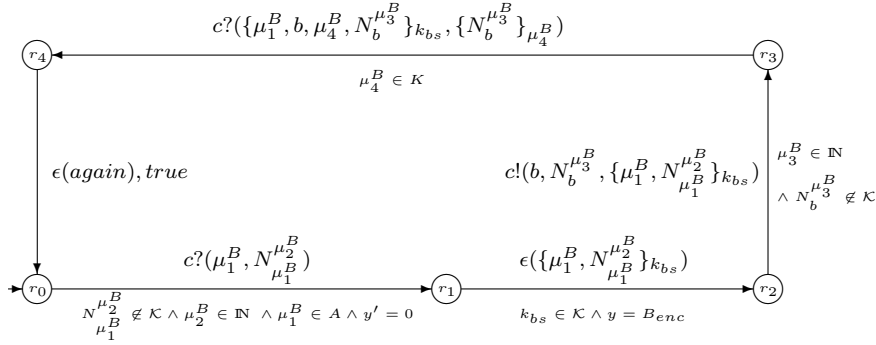
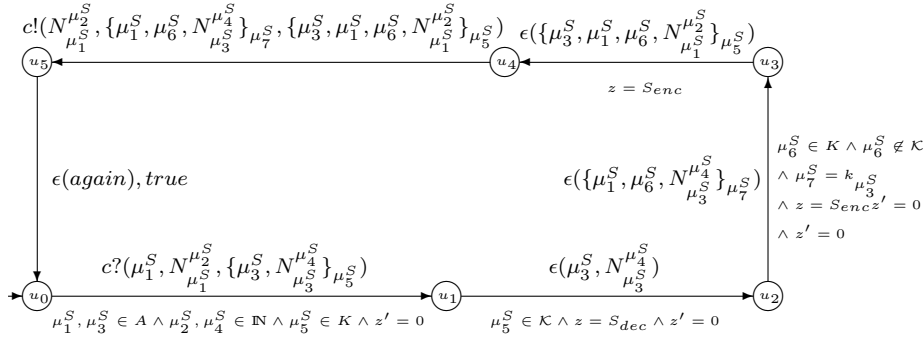
$$\begin{aligned} A &= (\{c\}, \{x, t\}, \mathcal{Y}^A, Q^A, q_0, \delta^A); \\ B &= (\{c\}, \{y\}, \mathcal{Y}^B, Q^B, r_0, \delta^B); \\ S &= (\{c\}, \{z\}, \mathcal{Y}^S, Q^S, u_0, \delta^S). \end{aligned}$$

We assume that all communications happen on the only public channel  $c$ . All principals  $i \in \{A, B, S\}$  have the set of variable messages  $\mathcal{Y}^i = \{\mu_1^i, \dots, \mu_6^i\}$ .

For readability, in Figure 5 we have not described the series of initial transition initializing the initial knowledges of the principals as follows:  $\{a, b, s, k_{as}\}$  for  $A$ ,  $\{b, s, k_{bs}\}$  for  $B$  and  $\{a, b, s, k_{as}, k_{bs}\}$  for  $C$ . Moreover, in Figure 5 we use the notation  $N_\mu^{\mu'}$  for  $Nonce(\mu, \mu')$ ;  $\tau, \tau'$  for  $Pair(\tau, \tau')$  and  $\{\tau\}_\mu$  for  $Enc(\tau, \mu)$ . We also suppose that when a message  $(\tau_1, \dots, \tau_n)$  is sent/received,  $\tau_1, \dots, \tau_n$  fall in the knowledge (this is simply implementable by a finite sequence of  $\epsilon$  transitions).

In the figure, for all principals, we omit the condition  $\forall \mu \in \mathcal{Y}^i \mu' = \mu$  from transition guards, but we assume that it holds for all transitions but the transitions with label  $\epsilon(\text{again}), \text{true}$ , where the guard  $\text{true}$  means that message variables assume a value non-deterministically.

Note that  $\mathcal{K}$  on a transition is intended to refer to the knowledge in the configuration from which the transition is performed.

**Principal A****Principal B****Principal S****Fig. 5** Specification with CSDMTAs of the principals of the Yahalom protocol.

If we do not consider the intruder, which is always capable of intercepting all messages, the guarantee that each message sent within the network will be received by the right participant is given by the reducibility of the sent terms.

The protocol is started by  $A$  that sends a message  $a, N_a^{\mu_1^A}$ , where  $N_a^{\mu_1^A}$  is a fresh unused nonce, and starts waiting for a reply from  $S$ . In the formula on this transition,  $N_a^{\mu_1^A}$  should not be in the knowledge of  $A$  (modelling the fact that it is freshly generated). When  $A$  sends the message to  $B$  it also resets a timer  $t$  to 0, and a timeout can be fixed (in the figure it is fixed to the value  $to$ ). Moreover, we assume that  $A$  knows the encryption/decryption time of  $B$  and  $S$ , and hence it does not expect to receive a message before time  $t_{min}$ .

After  $B$  receives the message from  $A$ , it checks whether the nonce is unused, generates a fresh nonce, and builds the encrypted terms it will send to  $S$ . Note that  $B_{enc}$  is a constant representing the time needed by  $B$  for encrypting a term.

When  $S$  receives the term from  $B$  it decipheres the encrypted term and builds the ciphered terms it should send to  $A$ . Again,  $S_{dec}$  and  $S_{enc}$  are constant values representing the time needed by  $S$  to decipher and cipher a term.

Note that the key chosen by  $S$  through the instantiation of the variable  $\mu_6^S$  is fresh (since it does not appear in the knowledge of  $S$ , it was not chosen before).

When  $S$  finishes elaborating the messages, it sends to  $A$  the message containing the new fresh key in the two subterms ciphered with  $k_{as}$  and  $k_{bs}$ , respectively.

If  $A$  receives such a message before the timeout has expired, it extracts the new fresh key and uses it to cipher the nonce sent by  $B$  to  $S$ , and then sends to  $B$  the ciphered nonce together with the term containing the fresh key built by  $S$  for  $B$ .

Some policies can be implemented when  $A$  does not receive an answer before the timeout expires. In Figure 5, we considered a couple of them (see the dashed transitions from  $q_1$ ). In one case,  $A$  may think that either  $B$  or  $S$  is "down" and aborts the execution of the protocol (dashed transition to state  $q_5$ ). In the other case,  $A$  may think that its first message was lost, and retransmits its first message (dashed transition to state  $q_0$ ). With our methodology, both policies are proven to be secure.

Given the specification of the Yahalom protocol through the model of CSDMTA, we can, in fact, prove the secrecy for the fresh generated key  $Kab$  (the intruder is not be able to deduce any information about the new session key). In particular we require that after the generation of the key (state  $u_3$  of  $S$ ) the key  $I(\mu_6^S)$  is  $\{I\}$ -secret for the CSDMTA  $\mathcal{AC}$  modeling the execution of the protocol.

We can also prove that the principal  $A$  truly authenticates to the principal  $B$  (during the execution of the communication protocol no other principal is able to simulate of being  $A$ ).

These properties can be verified through the reachability proof method, as we have shown in Example 5.

Let us consider a variant of the protocol in which a second timer is set by  $B$  after sending its message at step 2. In this case, if a timeout occurs, the retransmission decision is more delicate. It is not clear whether  $B$  should resend the original message, should send a new nonce or should abort. Intuitively, the nonce  $Nb$  could be reused. In fact, if we implement such a retransmission policy and we check again the secrecy and authentication properties, we obtain that the protocol it still secure, thus confirming that, in this case, retransmission of the same message by  $B$  is secure.

The intruder we have considered so far follows the classical Dolev–Yao assumptions. Since we are dealing with a quantitative framework, we might relax some of these

assumptions. We may consider, for example, a context of imperfect cryptography and allow the intruder to break an encrypted message after a certain amount of time has elapsed<sup>2</sup>. So, we can extend Definition 5, by taking into account the ability of the intruder to break a cryptographic message:

- (Breaking an encrypted message) for all  $i \in [1, m]$ , if  $Enc(\tau_1, \tau_2)$  is a subformula of  $\tau$  with  $(q, a!(\tau), \phi, q') \in \delta^i$ , then  $(q_0, \epsilon(\tau'_1), Enc(\tau'_1, \tau'_2)) \in \mathcal{K} \wedge x > t_b \wedge x' = 0, q_0) \in \delta$ , where  $\tau_1 \simeq \tau'_1$ ,  $\tau_2 \simeq \tau'_2$  and  $t_b$  is a constant modelling the robustness of the cryptographic algorithm.

Now, the intruder becomes able to break an encrypted message when a certain time delay passes by. Intuitively, the rule states that the intruder may spend an amount of time  $t_b$  to forge an encrypted message and get the plaintext. With such a rule, the timed context becomes more affecting, and the passage of time intrinsically interacts with the execution of the protocol. As it is easy to imagine, a secrecy or authentication property might hold only if the protocol ends before a certain amount of time has passed (the time needed by the intruder to forge a secret).

If we consider again our specification of the Yahalom protocol together with our improved intruder, we have that the key  $I(\mu_6^S)$  is not  $\{I\}$ -secret anymore for the CS-DMTA  $\mathcal{A}$  modelling the protocol. In particular, intercepting the message sent by principal  $S$  in state  $u_4$ , the intruder can use the new rule to forge the message and learn the secret  $I(\mu_6^S)$  after an amount of time  $t_b$  has elapsed.

Similar argumentations can be done about the authentication property: by forging the session key it will be able to build messages disguising its true identity.

As we have already suggested, to guarantee the secrecy or authentication property in this case, the protocol needs to be stopped (or reinitialised in order to produce a new fresh key) before the intruder forges the secret. In our case, we might add new transitions aborting the execution of the protocol (going in a final state) or restarting the protocol (going to the initial state) with guards on a new clock  $w$  checking that  $w = t_b$  (all other transitions should be guarded by  $w < t_b$ ).

## 5 Conclusions

We have defined systems of communicating timed automata endowed with structures to store information and functions to elaborate them, and we have discussed the expressive power of the model.

The decidability result implies that with 1-SDMTAs one may range not only on the uncountable state space induced by time aspects (made discrete through the usual region construction) but also on a denumerable range of discrete states induced by a variable that may take any natural number as a value. Therefore, the graph in which reachability is analyzed may be countably infinite.

SDMTAs may be applied, for example, in the field of security analysis. In Section 4 we proposed a variant of SDMTAs to analyse cryptographic protocols. In such a context, the reachability result for 1-SDMTAs could be used to define secrecy and authentication properties in a security setting where the knowledge of the adversary could be stored within the structures of our model.

---

<sup>2</sup> This amount of time should be modelled according to the robustness of the cryptographic algorithm under consideration.

Actually we have shown how this model can be used to specify cryptographic protocols and how we can verify that protocols satisfy secrecy and authentication properties.

However, since the reduction to VASs is exponential in the size of the 1-SDMTA, and the reachability problem for VASs is itself EXPSPACE hard, the complexity of our decision procedure is a limit of our methodology. Actually, with such a technique, large 1-SDMTAs are clearly out of the scope of automated verification techniques.

A future development of this paper may consist in the introduction of probability aspects within the framework of SDMTAs. Intuitively, the reachability result could be extended to a model with discrete probability distributions over the set of transitions exiting a state. This would permit to model and analyze also systems which exhibit probabilistic behaviour. On the one hand, it would be possible to quantify the security level of systems, calculating the probability of reaching states which are considered as insecure. On the other hand, also performance and reliability properties could be studied, motivating, again, the application of the model of SDMTAs for the analysis of distributed real-time data bases.

## References

1. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The Spi calculus. In *4th ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.
2. P. A. Abdallah and B. Jonsson. Verifying Networks of Timed Processes. Proc. of TACAS'98, volume 1384 of LNCS, pages 298–312, Springer, 1998.
3. J. Hoenicke and E. R. Olderog. CSP-OZ-DC: A Combination of Specification Techniques for Processes, Data and Time. *Nordic Journal of Computing*, 9(4):301–334, 2002.
4. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
5. G. Behrmann, A. David and K. G. Larsen. A Tutorial on Uppaal. *Springer LNCS*, 3185:200–236, 2004.
6. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Transactions in Software Engineering*, 17(3):259–273, 1991.
7. J. Wang. Timed Petri Nets: Theory and Application. Kluwer Academic Publishers, 1998.
8. A. Bouajjani, R. Echahed, and R. Robbana. On the Automatic Verification of Systems with Continuous Variables and Unbounded Discrete Data Structures. In Proc. Hybrid System II, *Springer LNCS*, 999:64–85, 1995.
9. M. Burrows, M. Abadi, and R. Needham. A logic for authentication. Technical Report 39, Digital Systems Research Center, Feb 1989.
10. R. Corin, S. Etalle, P. H. Hartel, and A. Mader. Timed analysis of security protocols. *Journal of Computer Security*, to appear. A preliminary report appeared as CTIT Technical Report TR-CTIT-05-14, University of Twente, The Netherlands, 2005.
11. C. J. F. Cremers, S. Mauw, and E. P. de Vink. Defining authentication in a trace model. In *1st International Workshop on Formal Aspects in Security and Trust (Fast 2003)*, pages 131–145, Pisa, IITT-CNR technical report, 2003.
12. Z. Dang. Pushdown time automata: a binary reachability characterization and safety verification. *Theoretical Computer Science*, 302: 93 – 121, 2003.
13. D. Dolev and A. C.-C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
14. R. Focardi, R. Gorrieri, and F. Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In *International Conference on Automata, Languages and Programming (ICALP'00)*, volume 1853 of LNCS, pages 354–372. Springer, 2000.
15. O. H. Ibarra and J. Su. Augmenting the Discrete Timed Automaton with Other Data Structures *Theoretical Computer Science*, 289: 191–204, 2002.
16. S. R. Kosaraju. Decidability of reachability in vector addition systems. In *14th Annual ACM Symp. on Theory of Computing*, pages 267–281. ACM Press, 1982.

- 
17. P. Krcal and W. Yi. Communicating Timed Automata: The More Synchronous, the More Difficult to Verify. *CAV'06*, volume 4144 of *LNCS*, pages 249–262, Springer, 2006.
  18. R. Lanotte. Expressive Power of Hybrid Systems with Real Variables, Integer Variables and Arrays. *Journal of Automata, Languages and Combinatorics*, accepted for publication.
  19. R. Lanotte, A. Maggiolo-Schettini, and A. Troina. Timed Automata with Data Structures for Distributed Systems Design and Analysis. In *3rd Int. Conference on Software Engineering and Formal Methods (SEFM'05)*, pages 44–53. IEEE Computer Society Press, 2005.
  20. G. Lowe. A hierarchy of authentication specifications. In *10th Int. Computer Security Foundations Workshop (CSFW'97)*, pages 31–44. IEEE Computer Society Press, 1997.
  21. M. Napoli, M. Parente, and A. Peron. Specification and verification of protocols with time constraints. *Electronic Notes in Theoretical Computer Science*, 99:205–227, 2004.
  22. L. C. Paulson. Relations between secrets: Two formal analyses of the Yahalom protocol. *Journal of Computer Security*, 9(3):197–216, 2001.
  23. The Object-Z specification language. *Kluwer Academic Publishers*, Norwell, 1999.
  24. V. Valero Ruiz, F. Cuartero Gomez, and D. de Frutos-Escrig. On non-decidability of reachability for timed-arc Petri nets. In *Proc. 8th Int. Work. Petri Nets and Performance Models (PNPM'03)*, pages 188–196. IEEE Computer Society Press, 1999.
  25. S. Yovine. Kronos: A verification tool for real-time systems. *International Journal on Software Tools for Technology Transfer*, 1:123–133, 1997.
  26. C. Zhou, C. A. R. Hoare, A. P. Ravn. A calculus of durations. *Information Processing Letters* 40(5):269–276, 1991.