Modeling Long–Running Transactions with Communicating Hierarchical Timed Automata

Ruggero Lanotte[1], Andrea Maggiolo-Schettini[2],
Paolo Milazzo[2], and Angelo Troina[2]

[1] Dip. di Scienze della Cultura, Politiche e dell'Informazione, Università dell'Insubria
[2] Dip. di Informatica, Università di Pisa

Abstract. Long-Running transactions consist of tasks which may be executed sequentially and in parallel, may contain sub-tasks, and may require to be completed before a deadline. These transactions are not atomic and, in case of executions which cannot be completed, a compensation mechanism must be provided.
In this paper we develop a model of Hierarchical Timed Automata suitable to describe the aspects mentioned. The automaton-theoretic approach allows the verification of properties by model checking. As a case study, we model and analyze an example of long–running transaction.

1 Introduction

The term *transaction* is commonly used in database systems to denote a logical unit of work designed for short-lived activities, usually lasting under a few seconds. These transactions are performed either completely or not at all: this means that if something goes wrong during the execution of the transaction, a roll–back activity is performed, which re–establishes the state of the system exactly as it was before the beginning of the transaction.

In order to permit the system to perform the roll–back activity, locks are acquired on the necessary resources at the beginning of a transaction and are released only at its end (in both the cases of completion and roll–back). The use of locks, which forbids others to access the resources, is justified by the short duration of the transaction. These transactions are called *ACID transactions*, because they satisfy the properties of Atomicity, Consistency, Isolation and Durability. Recent developments in distributed systems have created the need of a new notion of transaction, in which remote entities (possibly of different companies) may interact by performing complex activities (which may require also a human–interaction) that may take minutes, days or weeks. This increased length of time with respect to ACID transactions, forbids the use of locks on resources, and hence makes roll–back activities impossible. The alternative to roll-back activities in this kind of transactions is the use of compensations, which are activities explicitly programmed to remove the effects of the actions performed, and may require, for instance, the payment of some kind of penalty. This new kind of transactions are usually called *long–running transactions*, but they are also known as Sagas [7], web transactions [10], and extended transactions [9]. Although there is an interest for their support in distributed object–based middlewares [9], they are studied in particular in the context of orchestration languages for Web Services (such as BPEL4WS [8] and WSCI [12]).

Web Services are technologies that allow the distribution and the interoperability of heterogeneous software components (providing services) over the Internet. Orchestration languages allow the definition of complex services in terms of interactions among simpler services. Most orchestration languages offer several primitives for composing and handling services. Since the specifications of these languages mainly consist in informal textual description of their constructors, there is a strong interest in the formalization of their semantics (see [4–6, 10, 13]). Among these papers, [6, 10] give theoretical foundations to the fragments of orchestration languages describing long–running transactions. In particular, [6] identifies three main composition patterns for transactional activities with compensations, namely sequential composition, parallel composition, and nesting, and provides a formal semantics for them.

Communicating Hierarchical Machines (CHMs) [2], which are finite state machines endowed with the ability of refining states and of composing machines in parallel, seem to be a formalisms suitable to describe transactional activities and their composition patterns. Time is an important factor in the functioning of distributed systems, where communication may take time and deadlines may be used to counteract failure of remote components. Besides, transactions may have deadlines imposed by the requested QoS. Hence, to describe transactions a formalism is needed that also allows the representation of time constraints. After the seminal paper by Alur and Dill [1] many models of Timed Automata have been proposed and used to describe systems in which time cannot be abstracted. Furthermore, automata based formalisms are amenable to formal analysis, such as model checking.

In this paper we define the model of Communicating Hierarchical Transaction-based Timed Automata (CHTTAs). CHTTAs take from CHMs [2] the abilities of composing machines in parallel and hierarchically, but differ from CHMs insofar as they have two different terminal states (to describe different terminations of transactions) and provide different communication mechanisms. Moreover, CHTTAs have a notion of explicit time. We give a flattening procedure in order to obtain a timed automaton from a CHTTA, and prove the decidability of the reachability problem for CHTTAs. The class of flattened CHTTAs is a subclass of Timed Automata, hence our flattening procedure may be used in order to verify properties of CHTTAs with model checkers defined for timed automata (e.g. Kronos [14] and UPPAAL [3]).

We propose CHTTAs to describe transactional activities and define operations for composing CHTTAs which correspond to compositional patterns of transactional activities. In particular, among the patterns identified in [6], we focus on the sequential and parallel composition patterns for transactional activities. We give formal representations of these patterns in terms of CHTTAs and prove their correctness. As a case study, we model with CHTTAs a typical long–running transaction and verify some properties with the UPPAAL model checker [3].

2 Communicating Hierarchical Timed Automata

Let us assume a finite set of communication channels $\mathcal{C}$ with a subset $C_{Pub} \subseteq \mathcal{C}$. As usual, we denote with $a!$ the action of sending a signal on channel a and with $a?$ the action of receiving a signal on a.

Let us assume a finite set X of positive real variables called *clocks*. A *valuation* over X is a mapping $v : X \to \mathbb{R}^{\geq 0}$ assigning real values to clocks. Let V_X denote the set of all valuations over X. 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 $x \in X$.

The set of *constraints* over X, denoted $\Phi(X)$, is defined by the following grammar, where ϕ ranges over $\Phi(X)$, $x \in X$, $c \in \mathbb{Q}$ and $\sim \in \{<, \leq, =, \neq, >, \geq\}$:

$$\phi ::= x \sim c \,|\, \phi \wedge \phi \,|\, \neg\phi \,|\, \phi \vee \phi \,|\, true$$

We write $v \models \phi$ when *the valuation v satisfies the constraint ϕ*. Formally, $v \models x \sim c$ iff $v(x) \sim c$, $v \models \phi_1 \wedge \phi_2$ iff $v \models \phi_1$ and $v \models \phi_2$, $v \models \neg\phi$ iff $v \not\models \phi$, $v \models \phi_1 \vee \phi_2$ iff $v \models \phi_1$ or $v \models \phi_2$, and $v \models true$.

Let $B \subseteq X$; with $v[B]$ we denote the valuation resulting after resetting all clocks in B. More precisely, $v[B](x) = 0$ if $x \in B$, $v[B](x) = v(x)$, otherwise. Finally, with $\mathbf{0}$ we denote the valuation with all clocks reset to 0, namely $\mathbf{0}(x) = 0$ for all $x \in X$.

Definition 1. *A Transaction-based Timed Automaton (TTA) is a tuple $A = (\Sigma, X, S, Q, q_0, \delta)$, where:*

- *$\Sigma \subseteq \{a!, a? \mid a \in C\}$ is a finite set of labels;*
- *X is a finite set of clocks;*
- *S is a finite set of superstates;*
- *$Q = L \cup S \cup \{\odot, \otimes\}$, where L is a finite set of basic states and $\odot$ and $\otimes$ represent the special states commit and abort, respectively;*
- *$q_0 \in L$ is the initial state;*
- *$\delta \subseteq (L \times \Sigma \cup \{\tau\} \times \Phi(X) \times 2^X \times Q) \cup (S \times \{\boxdot, \boxtimes\} \times Q)$ is the set of transitions.*

Superstates are states that can be refined to automata (*hierarchical composition*). Note that from superstates in S only transitions with labels in $\{\boxdot, \boxtimes\}$ can be taken. We assume that $\odot$ and $\otimes$ are the final states of a TTA.

A TTA is said to be flat when it has no refinable states.

Definition 2 (Flat TTAs). *A TTA $A = (\Sigma, X, S, Q, q_0, \delta)$ is* flat *if $S = \emptyset$.*

Inspired by the definition of CHMs (see [2]) we now introduce CHTTAs as an extension of TTAs allowing superstate refinement and parallelism.

Definition 3. *Let $\Sigma_{Pub} = \{a!, a? \mid a \in C_{Pub}\}$ and $\mathcal{A} = \{A^1, \ldots, A^n\}$ be a finite set of TTAs, with $A^i = (\Sigma^i, X^i, S^i, Q^i, q_0^i, \delta^i)$ and such that there exists m ($m < n$) such that A^j is flat if and only if $j \geq m$. A* Communicating Hierarchical Transaction-based Timed Automaton *($CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$) is given by the following grammar:*

$$CHTTA_{\mathcal{A}}^{\Sigma_{Pub}} \;::=\; \langle A^i, \mu \rangle \quad | \quad CHTTA_{\mathcal{A}}^{\Sigma_{Pub}} || CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$$

where μ is a hierarchical composition function *$\mu : S^i \to CHTTA_{\{A^{i+1}, \ldots, A^n\}}^{\Sigma_{Pub}}$.*

Parallelism allows concurrent execution of automata. Hierarchical composition allows refining superstates. Automata executed in parallel may communicate by synchronizing transitions labeled with a sending and a receiving action on the same channel. Communication performed using non public channels are only allowed between components inside the same superstate or at top–level. Communication performed by using public channels have no restrictions.

Note that, by definition of $\mathcal{A}$ and μ, cyclic nesting is avoided. In the following, if it does not give rise to ambiguity, we may write CHTTA instead of $\text{CHTTA}_{\mathcal{A}}^{\Sigma_{Pub}}$. Finally, if A is a flat TTA, in $\langle A, \mu \rangle$ μ is an empty function.

Example 1. In Figure 1 we show an example of CHTTA. Superstates of the CHTTA are depicted as boxes and basic states as circles; initial states are represented as vertical segments. Transitions are labeled arrows in which labels τ and constraints *true* are omitted. Containment into boxes represents hierarchical composition, while parallel composition is represented by juxtapositions. The CHTTA in the figure is formally defined as $\langle(\emptyset, \emptyset, \{s_1\}, \{q_0, s_1, \odot, \otimes\}, q_0, \delta), \mu\rangle$ where $\delta = \{(q_0, \tau, true, \emptyset, s_1), (s_1, \boxdot, \odot), (s_1, \boxtimes, \otimes)\}$, and $\mu(s_1) = A_1 || A_2$. A_1 and A_2 are defined as $A_1 = \langle(\{a!, b?\}, \{x\}, \emptyset, \{q_0, q_1, \odot, \otimes\}, q_0, \delta_1)$ and $A_2 = \langle(\{a?, b!\}, \emptyset, \emptyset, \{q_0, q_2, \odot, \otimes\}, q_0, \delta_2)$, where $\delta_1 = \{(q_0, a!, true, \{x\}, q_1), (q_1, b?, x < 5, \emptyset, \odot), (q_1, \tau, x \geq 5, \emptyset, \otimes)\}$ and $\delta_2 = \{(q_0, a?, true, \emptyset, q_2), (q_2, b!, true, \emptyset, \odot)\}$.

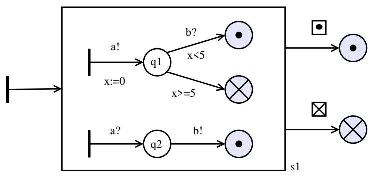

Fig. 1. Example of CHTTA.

2.1 Semantics of CHTTAs

Configurations of CHTTAs are pairs $tc = (c, \nu)$ where c, the *untimed configuration*, represents the currently active states, and ν, the *composed valuation*, represents the current clock valuations.

The configuration of a CHTTA without parallel components, when the currently active state is a basic state, is a pair (q, v) with q the currently active state, and v the automaton clock valuation. We represent with $q.c$ the configuration where q is a superstate and c is the untimed configuration of $\mu(q)$, and with $v.\nu$ the composed valuation where v is the clock valuation of the automaton having q as superstate and ν is the composed valuation of the clocks of $\mu(q)$. We denote with $c_1; c_2$ the untimed configuration of the parallel composition of two CHTTAs having c_1 and c_2 as untimed configurations. Analogously, we denote with $\nu_1; \nu_2$ the composed valuation of the parallel composition of two CHTTAs having ν_1 and ν_2 as composed valuations.

Formally, the set of configurations $Conf(A)$ of a CHTTA A is inductively defined as follows:

- if $A = \langle(\Sigma, X, S, Q, q_0, \delta), \mu\rangle$, then $Conf(A) = \{(Q \backslash S) \times V_X\} \cup \{(q.c, v.\nu) \mid q \in S \wedge v \in V_x \wedge (c, \nu) \in Conf(\mu(q))\}$;
- if $A = A_1||A_2$ then $Conf(A) = \{(c_1; c_2, \nu_1; \nu_2) \mid (c_1, \nu_1) \in Conf(A_1) \wedge (c_2, \nu_2) \in Conf(A_2)\}$.

For a composed valuation ν and a time value $t \in \mathbb{R}^{\geq 0}$, let $\nu + t$ denote the composed valuation such that $(v+t)(x) = v(x)+t$, for each valuation v occurring in ν.

The initial configuration of A, denoted $Init(A) \in Conf(A)$, is the configuration (c, ν) such that each state occurring in c is an initial state and each valuation occurring in ν is $\mathbf{0}$.

We give a semantics of CHTTAs in SOS style as a labeled transition system where states are pairs (A, tc) with $A \in \text{CHTTA}_{\mathcal{A}}^{\Sigma_{Pub}}$ and $tc \in Conf(A)$, and labels are in $\mathbb{R}^{>0} \cup \bigcup_i \Sigma^i \cup \{\tau\}$.

In order to simplify the SOS semantics for CHTTAs we introduce a notion of structural equivalence for pairs (A, tc), accounting for commutativity and associativity of parallelism. The relation $\approx$ is the least equivalence relation satisfying $(A_1||A_2, tc_1; tc_2) \approx (A_2||A_1, tc_2; tc_1)$ and $(A_1||(A_2||A_3), tc_1; (tc_2; tc_3)) \approx ((A_1||A_2)||A_3, (tc_1; tc_2); tc_3)$. Moreover, given an untimed parallel configuration $c = c_1; \ldots; c_n$ we use the following notations: $c \approx \odot$ if for $\forall i.c_i = \odot$, and $c \approx \otimes$ if for $\exists i.c_i = \otimes \wedge \forall i \neq j.c_j \in \{\odot, \otimes\}$.

Definition 4 (Semantics of CHTTAs). *Given $A \in CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$, the semantics of a A is the least labeled transition relation $\xrightarrow{\alpha}$ over $\{A\} \times Conf(A)$ closed with respect to structural equivalence and satisfying the rules in Figure 2.*

Rule (T) allows the elapsing of time for a generic CHTTA A. We note that the time t is the same for any TTA composing A.

Rules (C1) and (C2) describe the behavior of a flat TTA. From a configuration (q, v), the step is performed due to a transition (q, α, ϕ, B, q') such that the condition ϕ is satisfied by v. After the step, the flat TTA is in the configuration composed by state q' and where clocks in B are reset. If q' is a superstate (rule (C2)), then the CHTTA $\mu(q')$ becomes active inside q'.

The synchronization step is described by rule (P2). By definition of the relation $\approx$ also CHTTAs that are not neighborhood in the parallel composition can communicate.

Rules (C3) and (P1) allow expanding the step of a TTA which is a component of a CHTTA. Rule (C3) deals with the hierarchical composition and rule (P1) deals with the parallel composition. The label of the step is either τ or a public channel. Hence, thanks to rule (P2), communication between TTAs in parallel is allowed both for private and public channels, while for TTAs in different superstates the communication is allowed only if the channel is public. Moreover, we note that the step we are expanding cannot be a time step. Hence, time steps can be performed only by the root, implying that the time elapsed is the same for each TTA composing the CHTTA we are considering.

Each execution of a superstate terminates with either a commit or an abort state. Rules (Com1) and (Com2) deal with the case in which the commit of the

$$\frac{t \in \mathbb{R}^{>0}}{(A,(c,\nu)) \xrightarrow{t} (A,(c,\nu+t))} \quad \text{(T)}$$

$$\frac{(q,\alpha,\phi,B,q') \in \delta \quad v \models \phi \quad q' \notin S}{(\langle A,\mu\rangle,(q,v)) \xrightarrow{\alpha} (\langle A,\mu\rangle,(q',v[B]))} \quad \text{(C1)}$$

$$\frac{(q,\alpha,\phi,B,q') \in \delta \quad v \models \phi \quad q' \in S \quad Init(\mu(q')) = (c,\nu)}{(\langle A,\mu\rangle,(q,v)) \xrightarrow{\alpha} (\langle A,\mu\rangle,(q'.c,v[B].\nu))} \quad \text{(C2)}$$

$$\frac{(\mu(q),(c,\nu)) \xrightarrow{\alpha} (\mu(q),(c',\nu')) \quad \alpha \in \Sigma_{Pub} \cup \{\tau\}}{(\langle A,\mu\rangle,(q.c,v.\nu)) \xrightarrow{\alpha} (\langle A,\mu\rangle,(q.c',v.\nu'))} \quad \text{(C3)}$$

$$\frac{(A_1,(c_1,v)) \xrightarrow{\alpha} (A_1,(c_1',v')) \quad \alpha \in \Sigma_{Pub} \cup \{\tau\}}{(A_1||A_2,(c_1;c_2,v)) \xrightarrow{\alpha} (A_1||A_2,(c_1';c_2,v'))} \quad \text{(P1)}$$

$$\frac{(A_1,(c_1,v)) \xrightarrow{a!} (A_1,(c_1',v')) \quad (A_2,(c_2,v')) \xrightarrow{a?} (A_2,(c_2',v''))}{(A_1||A_2,(c_1;c_2,v)) \xrightarrow{\tau} (A_1||A_2,(c_1';c_2',v''))} \quad \text{(P2)}$$

$$\frac{c \approx \odot \quad (q,\boxdot,q') \in \delta \quad q' \notin S}{(\langle A,\mu\rangle,(q.c,v.\nu)) \xrightarrow{\tau} (\langle A,\mu\rangle,(q',v))} \quad \text{(Com1)}$$

$$\frac{c \approx \odot \quad (q,\boxdot,q') \in \delta \quad q' \in S \quad Init(\mu(q')) = (c',\nu')}{(\langle A,\mu\rangle,(q.c,v.\nu)) \xrightarrow{\tau} (\langle A,\mu\rangle,(q'.c',v.\nu'))} \quad \text{(Com2)}$$

$$\frac{c \approx \otimes \quad (q,\boxtimes,q') \in \delta \quad q' \notin S}{(\langle A,\mu\rangle,(q.c,v.\nu)) \xrightarrow{\tau} (\langle A,\mu\rangle,(q',v))} \quad \text{(Ab1)}$$

$$\frac{c \approx \otimes \quad (q,\boxtimes,q') \in \delta \quad q' \in S \quad Init(\mu(q')) = (c',\nu')}{(\langle A,\mu\rangle,(q.c,v.\nu)) \xrightarrow{\tau} (\langle A,\mu\rangle,(q'.c',v.\nu'))} \quad \text{(Ab2)}$$

Where we assume $A = (\Sigma, X, S, Q, q_0, \delta)$ except for rule (T) where A is a generic CHTTA.

Fig. 2. SOS semantics for CHTTAs.

superstate takes the TTA to a basic state or to a superstate, respectively, and rules (Ab1) and (Ab2) deal with the case in which the abort of the superstate takes the TTA to a basic state or to a superstate, respectively.

Given a string $w = \alpha_1 \ldots \alpha_m$, we will write $(A,(c,\nu)) \stackrel{w}{\Longrightarrow} (A,(c',\nu'))$ to denote the existence of a sequence of steps $(A,(c,\nu)) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_m} (A,(c',\nu'))$. We denote with $|w| = m$ the length of w and with $w[i] = \alpha_i$ the $i-$th label.

With $\mathcal{L}(A,\Sigma_V)$ we denote *the language accepted by a CHTTA A w.r.t. a set of visible actions $\Sigma_V \subseteq \Sigma_{Pub}$*. Namely, $\mathcal{L}(A,\Sigma_V) = \{w \in (\{\tau\} \cup \Sigma_V \cup \mathbb{R}^{>0})^* \mid (A, Init(A)) \stackrel{w}{\Longrightarrow} (A,(\odot,\nu')) \text{ or } (A, Init(A)) \stackrel{w}{\Longrightarrow} (A,(\otimes,\nu'))\}$.

The following proposition holds.

Proposition 1. *The class of flat TTAs is equivalent to the class of Timed Automata.*

3 Deciding Reachability for CHTTAs

Reachability is interesting for proving properties. For timed Automata the reachability problem is PSPACE-COMPLETE. In our case the problem is still decidable, but becomes EXPSPACE-COMPLETE.

Firstly, we give an algorithm for flattening a generic CHTTA, hence the reachability problem can be checked on the Timed Automaton resulting by the flattening. Due to the complexity of the flattening, the reachability problem for CHTTAs is EXPSPACE-COMPLETE. The increase of complexity is caused by the communication between different superstates, but it is not caused by the number of clocks.

3.1 Flattening CHTTAs

Let $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_n\}$ and ϕ be a formula in $\Phi(X)$. With $\phi[Y := X]$ we denote the formula where each clock y_i is replaced with x_i. Moreover, with $X_{i,j}$ we denote the renaming of clocks x in X with clocks $x^{i,j}$, more precisely $X_{i,j} = \{x_1^{i,j}, \dots, x_n^{i,j}\}$.

Given a $CHTTA$ A with $w(A)$ we denote the maximum width of the CHTTAs composing A. Namely:

$$w(\langle A_1, \mu_1\rangle \| \dots \| \langle A_m, \mu_m\rangle) = max\{m, w(\langle A_1, \mu_1\rangle), \dots, w(\langle A_m, \mu_m\rangle)\},$$

where $w(\langle A, \mu\rangle) = max\{w(\mu(q)) \mid q \in S\}$.

Moreover, $d(A)$ denotes the maximum depth of A. Namely:

$$d(\langle A_1, \mu_1\rangle \| \dots \| \langle A_m, \mu_m\rangle) = max\{d(\langle A_1, \mu_1\rangle), \dots, d(\langle A_m, \mu_m\rangle)\},$$

where $d(\langle A, \mu\rangle) = 1 + max\{d(\mu(q)) \mid q \in S\}$.

Definition 5. *Let $\mathcal{A} = \{A^1, \dots, A^n\}$, with $A^i = (\Sigma^i, X^i, S^i, Q^i, q_0^i, \delta^i)$, be a set of TTAs, and $A \in CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$. Given $\Sigma_V \subseteq \Sigma_{Pub}$, with $Flat(A, \Sigma_V)$ we denote the flat TTA $(\Sigma, X, \emptyset, Q, q_0, \delta)$ such that:*

- $\Sigma = \Sigma_V$;
- $X = \bigcup_{i \in [1, d(A)]} \bigcup_{j \in [1, w(A)]} X_{i,j}$;
- $Q = \{c \mid (c, \nu) \in Conf(A)\}$;
- $q_0 = c_0$ *such that* $Init(A) = (c_0, \nu)$ *is the initial configuration of* A;
- δ *is such that:*
  - $(c, \tau, true, \emptyset, c')$ *is in* δ *if there exists a step* $(A, (c, \nu)) \xrightarrow{\tau} (A, (c', \nu'))$ *triggered by either a commit or an abort transition;*
  - (c, α, ϕ, B, c') *is in* δ *if there exists a step* $(A, (c, \nu)) \xrightarrow{\alpha} (A, (c', \nu'))$, *with* $\alpha \in \Sigma_V$ *triggered by the transition* (q, α, ϕ, B, q') *of a TTA* A^i;
  - (c, τ, ϕ, B, c') *is in* δ *if there exists a step* $(A, (c, \nu)) \xrightarrow{\tau} (A, (c', \nu'))$ *triggered by the transition* $(q^1, a!, \phi^1, B^1, p^1)$ *of the TTA* A^i *at position* i_1, j_1 *and by the transition* $(q^2, a?, \phi^2, B^2, p^2)$ *of the TTA* A^j *at position* i_2, j_2 *such that* $\phi = (\phi_1[X^i := (X^i)_{i_1,j_1}]) \wedge (\phi_2[X^j := (X^j)_{i_2,j_2}])$ *and* $B = (B^1)_{i_1,j_1} \cup (B^2)_{i_2,j_2}$.

Proposition 2. *Let $\mathcal{A} = \{A^1, \dots, A^n\}$ and $A \in CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$ where each A^i has at most h states and k clocks. The reachability problem for A can be computed in $O(h^{w(A)^{d(A)}} \cdot 2^{k \cdot d(A) \cdot w(A)})$.*

Proof. $Flat(A, \Sigma_V)$ has at most $h^{w(A)^{d(A)}}$ states. Actually, the root is a parallel composition of at most $w(A)$ CHTTAs. Each of them has depth of at most $d(A) - 1$ and, by induction, has $h^{w(A)^{d(A)-1}}$ state configurations implying that the number of state configurations is $h^{w(A)^{d(A)}}$.

Given a Timed Automaton with d states and l clocks, the reachability problem can be solved in $d \cdot 2^l$ (see [1]). Hence, since $Flat(A, \Sigma_V)$ has $h^{w(A)^{d(A)}}$ states and at most $k \cdot d(A) \cdot w(A)$ clocks, the reachability problem for A can be computed in $h^{w(A)^{d(A)}} \cdot 2^{k \cdot d(A) \cdot w(A)}$. □

Therefore, the reachability problem for a CHTTA A is EXPSPACE-COMPLETE w.r.t. m, $w(A)$ and $d(A)$. As it happens for the reachability problem for Timed Automata (see [1]), the number of clocks does not influence the complexity.

Proposition 3. *Let $\mathcal{A} = \{A^1, \ldots, A^n\}$ and $A \in CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$, where each A^i has at most m states. The reachability problem for A is EXPSPACE-COMPLETE w.r.t. m, $w(A)$ and $d(A)$.*

Proof. In [2] it is proved that the reachability problem for CHMs is EXPSPACE–COMPLETE w.r.t. m, $w(A)$ and $d(A)$. The same holds for *untimed* CHTTAs, hence, the reachability problem for CHTTAs is at least EXPSPACE–COMPLETE w.r.t. m, $w(A)$ and $d(A)$. Therefore, if the reachability problem is PSPACE–COMPLETE for $Flat(A, \Sigma_V)$, then the thesis holds. But $Flat(A, \Sigma_V)$ has at most $k \cdot w(A) \cdot d(A)$ clocks, that is a polynomial number of clocks, and hence the thesis holds since the reachability problem for Timed Automata is PSPACE–COMPLETE [1]. □

4 Compositional Patterns for Long–Running Transactions

A *long–running transaction* is composed by atomic activities (called *subtransactions* or simply *activities*) that should be executed completely. Atomicity for activities means that they are either successfully executed (*committed*) or no effect is observed if their execution fails (*aborted*). Activities may be composed by other subtransactions.

Partial executions of a long–running transaction are not desirable, and, if they occur, they must be compensated for. Therefore, all the activities A_i in a long–running transaction have a compensating activity B_i that can be invoked to repair from the effects of a successful execution of A_i if some failure occurs later. Compensations are assumed to be transactions that always complete their execution successfully (they always commit and can never abort).

We assume that both activities and compensations are described as CHTTAs, and we denote with $A \Rsh B$ the association of compensation B with activity A.

Following the approach in [6], we identify some composition patterns for transactional activities with compensations. In particular, we focus on the sequential composition pattern and on the parallel composition one.

We denote with $A_1 \Rsh B_1 \cdot A_2 \Rsh B_2$ the sequential composition of two transactional activities with compensations, and we use the standard parallel composition of CHTTAs also to describe parallel composition of transactional activities with compensations. We show that the compositional patterns on transactional activities can be formulated as compositions of CHTTAs.

4.1 Sequential Transactions

Activities $A_1, \ldots, A_n$ composing a *sequential transaction* are assumed to be executed sequentially, namely, when activity A_i is committed, activity A_{i+1} starts its execution. Compensation activities $B_1, \ldots, B_n$ are associated with each activity A_i. Transactions of this kind must be guaranteed that either the entire sequence $A_1, \ldots, A_n$ is executed or the compensated sequence $A_1, \ldots, A_i, B_i, \ldots, B_1$ is executed for some $i < n$. The first case means that all activities in the sequence completed successfully, thus representing a successful commit of the whole transaction. The second case stands for the abort of activity A_{i+1}; hence, all the activities already completed $(A_1, \ldots, A_i)$ are recovered by executing the compensating activities $(B_i, \ldots, B_1)$.

In Figure 3 (a) we show the CHTTA $A = [\![A_1 \Rsh B_1 \cdot A_2 \Rsh B_2]\!]^S$ modeling the pattern of sequential transactions. We consider just two activities A_1, A_2 and compensations B_1, B_2. Note that, since the transaction is composed by only two activities, the compensation B_2 is not executed. This is because compensations are invoked only for activities that complete successfully, however, if activity A_2 commits, then the whole transaction successfully commits and no compensation needs to be invoked. The compensation B of the whole transactional activity A is defined as the sequential execution of the compensations B_2 and B_1 (see Figure 3 (b)).

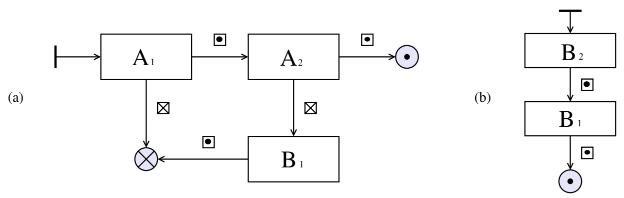

Fig. 3. Pattern for Sequential Transactions.

Definition 6. *Given* $A_1, A_2, B_1, B_2 \in CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$ *we define the sequential composition of activities* A_1, A_2 *with compensations* B_1, B_2 *as the* $CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$ $A = [\![A_1 \Rsh B_1 \cdot A_2 \Rsh B_2]\!]^S = \langle(\emptyset, \emptyset, \{s_1, s_2, s_3\}, \{s_1, s_2, s_3, q_0 \odot, \otimes\}, q_0, \delta), \mu\rangle$ *where* $\delta = \{(q_0, \tau, true, \emptyset, s_1), (s_1, \boxdot, s_2), (s_1, \boxtimes, \otimes), (s_2, \boxdot, \odot), (s_2, \boxtimes, s_3), (s_3, \boxdot, \otimes)\}$ *and* $\mu = \{(s_1, A_1), (s_2, A_2), (s_3, B_1)\}$. *The compound compensation of* A *is defined as the* $CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$ $B = [\![B_1 \cdot B_2]\!]_C^S = \langle(\emptyset, \emptyset, \{s_1, s_2\}, \{s_1, s_2, q_0, \odot, \otimes\}, q_0, \delta'), \mu'\rangle$ *with* $\delta' = \{(q_0, \tau, true, \emptyset, s_2), (s_2, \boxdot, s_1), (s_1, \boxdot, \odot)\}$ *and* $\mu' = \{(s_1, B_1), (s_2, B_2)\}$.

Considering only two activities in the sequential pattern is not a real limitation, since the case of n activities may be reduced by iteratively grouping the activities in pairs. Intuitively, $A = [\![A_1 \Lsh B_1 \cdot A_2 \Lsh B_2 \cdot A_3 \Lsh B_3]\!]^S = [\![A' \Lsh B \cdot A_3 \Lsh B_3]\!]^S$ where $A' = [\![A_1 \Lsh B_1 \cdot A_2 \Lsh B_2]\!]^S$ and B is the compensation for the whole sequential subtransaction A' (see Figure 4).

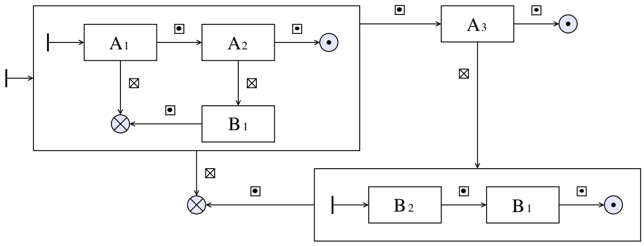

Fig. 4. Composing Sequential Transactions.

In order to prove the correctness of our definitions of compositional patterns, we introduce the notion of *wrapped* CHTTAs. Intuitively, for a CHTTA A, we call *wrapper* the automaton A^M which performs the special action $commit_A!$ before reaching the final commit state.

Given a CHTTA A, $A^M = \langle (\{commit_A!\}, \emptyset, \{s\}, Q, q_0, \delta), \mu \rangle$ is the *wrapped* CHTTA of A with set of states $Q = \{s, q_0, q_1, \odot, \otimes\}$, set of transitions $\delta = \{(q_0, \tau, true, \emptyset, s), (s, \boxdot, q_1), (s, \boxtimes, \otimes), (q_1, commit_A!, true, \emptyset, \odot)\}$ and $\mu(s) = A$. In Figure 5 we show the CHTTA A^M.

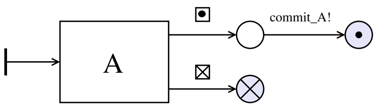

Fig. 5. A^M.

The next lemma derives immediately from the definition of A^M.

Lemma 1. *Given a CHTTA A, $(A, (c, \nu)) \stackrel{w}{\Longrightarrow} (A, (c', \nu'))$, with $c \not\approx \odot$ and $c \not\approx \otimes$ and either $c' \approx \odot$ or $c' \approx \otimes$ if and only if $(A^M, (s \cdot c, \epsilon \cdot \nu)) \stackrel{w'}{\Longrightarrow} (A^M, (s \cdot \hat{c}, \epsilon \cdot \hat{\nu}))$, where (given $\tilde{z} \in \{\mathbb{R}^{>0}\}^*$):*

$$\begin{cases} w' = \tilde{z} \cdot \tau \cdot w \cdot \tau \cdot commit_A! \;\; \text{and} \;\; \hat{c} = \odot & \text{if } c' \approx \odot \\ w' = \tilde{z} \cdot \tau \cdot w \cdot \tau \;\; \text{and} \;\; \hat{c} = \otimes & \text{if } c' \approx \otimes \end{cases}$$

Let us assume $\Sigma_V = \{commit_{A_1}!, commit_{B_1}!, \ldots, commit_{A_n}!, commit_{B_n}!\}$.

Theorem 1 (Correct Completion). *Given $A = [\![A_1^M \Lsh B_1^M \cdot \ldots \cdot A_n^M \Lsh B_n^M]\!]^S$, $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu))$ if and only if $w \in \mathcal{L}(A, \Sigma_V)$ and $w = \tilde{x}_1 \cdot commit_{A_1}! \cdot \ldots \cdot \tilde{x}_n \cdot commit_{A_n}! \cdot \tilde{x}_{n+1}$ where $\tilde{x}_i \in (\{\tau\} \cup \mathbb{R}^{>0})^*$.*

Proof. By definition of accepted language, it is obvious that if $w \in \mathcal{L}(A, \Sigma_V)$, then $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu))$. Hence we prove only the implication $\Rightarrow$.

We prove by induction on n that, given $c = c_1; \ldots; c_n$ with $c_i \not\approx \odot$, if $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu))$, then $w = \tilde{x}_1 \cdot commit_{A_1}! \cdot \ldots \cdot \tilde{x}_n \cdot commit_{A_n}! \cdot \tilde{x}_{n+1}$ where $\tilde{x}_i \in (\{\tau\} \cup \mathbb{R}^{>0})^*$.

If $n = 1$ then the thesis holds by Lemma 1. If $n > 1$, then $A_1^M \Rsh B_1^M \cdot \ldots \cdot A_n^M \Rsh B_n^M$ is synthesized as $A' \Rsh B' \cdot A_n^M \Rsh B_n^M$ where $A' = A_1^M \Rsh B_1^M \cdot \ldots \cdot A_{n-1}^M \Rsh B_{n-1}^M$ and B is the sequence of compensations $B_1, \ldots, B_{n-1}$.

By induction, if $(A', Init(A)) \stackrel{w'}{\Longrightarrow} (A', (\odot, \nu))$, then $w' = \tilde{y}_1 \cdot commit_{A_1}! \cdot \ldots \cdot \tilde{y}_{n-1} \cdot commit_{A_{n-1}}! \cdot \tilde{y}_n$ where $\tilde{y}_i \in (\{\tau\} \cup \mathbb{R}^{>0})^*$. Now, by Lemma 1, if $(A_n^M, (c, \nu) \stackrel{w''}{\Longrightarrow} (A_n^M, (\odot, \nu'))$, then $w'' = \tilde{z} \cdot commit_{A_n}! \cdot \tilde{z}'$, and, hence, for a fixed $w = w' \cdot w''$ and $\tilde{x}_1 = \tilde{y}_1, \ldots, \tilde{x}_{n-1} = \tilde{y}_{n-1}$ and $\tilde{x}_n = \tilde{y} \cdot \tilde{z}$ and $\tilde{x}_{n+1} = \tilde{z}'$ we have that $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu'))$ with $w = \tilde{x}_1 \cdot commit_{A_1}! \cdot \ldots \cdot \tilde{x}_n \cdot commit_{A_n}! \cdot \tilde{x}_{n+1}$ where $\tilde{x}_i \in (\{\tau\} \cup \mathbb{R}^{>0})^*$. □

Theorem 2 (Correct Compensation). *Given $A = [\![A_1^M \Rsh B_1^M \cdot \ldots \cdot A_n^M \Rsh B_n^M]\!]^S$, $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\otimes, \nu))$ if and only if, $w \in \mathcal{L}(A, \Sigma_V)$ and, for some $k \in [1, n]$, $w = \tilde{x}_1 \cdot commit_{A_1}! \cdot \ldots \cdot \tilde{x}_{k-1} \cdot commit_{A_{k-1}}! \cdot \tilde{x}'_{k-1} \cdot commit_{B_{k-1}}! \cdot \ldots \cdot \tilde{x}'_1 \cdot commit_{B_1}! \cdot \tilde{x}'$ where $\tilde{x}_i, \tilde{x}'_i \in (\{\tau\} \cup \mathbb{R}^{>0})^*$.*

Proof. By definition of accepted language, it is obvious that if $w \in \mathcal{L}(A, \Sigma_V)$, then $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\otimes, \nu))$. Hence we prove only the implication $\Rightarrow$.

We prove by induction on n that, if$(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\otimes, \nu))$, then $w \in \mathcal{L}(A)$ and, for some $k \in [1, n]$, $w = \tilde{x}_1 \cdot commit_{A_1}! \cdot \ldots \cdot \tilde{x}_{k-1} \cdot commit_{A_{k-1}}! \cdot \tilde{x}'_{k-1} \cdot commit_{B_{k-1}}! \cdot \ldots \cdot \tilde{x}'_1 \cdot commit_{B_1}! \cdot \tilde{x}'$ where $\tilde{x}_i, \tilde{x}'_i \in \{\{\tau\} \cup \mathbb{R}^{>0}\}^*$.

If $n = 1$ then the thesis holds by Lemma 1. If $n > 1$, then $A_1^M \Rsh B_1^M \cdot \ldots \cdot A_n^M \Rsh B_n^M$ is synthesized as $A' \Rsh B' \cdot A_n^M \Rsh B_n^M$ where $A' = A_1^M \Rsh B_1^M \cdot \ldots \cdot A_{n-1}^M \Rsh B_{n-1}^M$ and B is the sequence of compensations $B_1, \ldots, B_{n-1}$.

We have two cases. If $(A', Init(A')) \stackrel{w'}{\Longrightarrow} (A', (\otimes, \nu))$, then the thesis holds by induction. Otherwise, if $(A', Init(A')) \stackrel{w'}{\Longrightarrow} (A', (\odot, \nu))$, then by Theorem 1 $w' = \tilde{y}_1 \cdot commit_{A_1}! \cdot \ldots \cdot \tilde{y}_{n-1} \cdot commit_{A_{n-1}}! \cdot \tilde{y}_n$ where $\tilde{y}_i \in (\{\tau\} \cup \mathbb{R}^{>0})^*$. Now, by Lemma 1, if $(A_n^M, (c, \nu)) \stackrel{w''}{\Longrightarrow} (A_n^M, (\otimes, \nu'))$, then $w' = \tilde{z}$, and, $(B', (Init(B')) \stackrel{w'''}{\Longrightarrow} (B', (\odot, \nu''))$ with $w''' = \tilde{y}'_{n-1} \cdot commit_{B_{n-1}}! \cdot \ldots \cdot \tilde{y}'_1 \cdot commit_{B_1}! \cdot \tilde{y}'$. Therefore, for a fixed $w = w \cdot w' \cdot w'' \cdot w'''$; $\tilde{x}_1 = \tilde{y}_1, \ldots, \tilde{x}_{n-1} = \tilde{y}_{n-1}$; $\tilde{x}'_1 = \tilde{y}'_1, \ldots, \tilde{x}'_{n-2} = \tilde{y}'_{n-2}$ and $\tilde{x}'_{n-1} = \tilde{z} \cdot \tilde{y}'_{n-1}$ we have that $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\otimes, \nu'))$ and $w = \tilde{x}_1 \cdot commit_{A_1}! \cdot \ldots \cdot \tilde{x}_{n-1} \cdot commit_{A_{n-1}}! \cdot \tilde{x}'_{n-1} \cdot commit_{B_{n-1}}! \cdot \ldots \cdot \tilde{x}'_1 \cdot commit_{B_1}! \cdot \tilde{x}'$ where $\tilde{x}_i, \tilde{x}'_i \in \{\{\tau\} \cup \mathbb{R}^{>0}\}^*$. □

4.2 Parallel Transactions

If activities $A_1, \ldots, A_n$ composing a *parallel transaction* are executed concurrently, the whole transaction terminates when all the activities A_i complete their execution. Again, we assume compensation activities $B_1, \ldots B_n$. If all the activities terminate successfully then the whole transaction reaches a commit state. If some A_i aborts, then compensation activities should be invoked for the activities that completed successfully. In this latter case, the result of the whole transaction is “abort”.

The pattern for parallel transactions is shown in Figure 6. As for sequential transactions, we consider only two activities A_1, A_2 with compensations B_1, B_2 composed in parallel, thus resulting in the CHTTA $A = [\![A_1 \Rsh B_1 || A_2 \Rsh B_2]\!]^P$ of Figure 6. We remark that, by the semantics of CHTTAs, the parallel operator $||$ is assumed to be commutative and associative. In such a pattern, activities A_1 and A_2 are executed concurrently together with a *controller* that invokes compensations when one of the two activities commits and the other aborts.

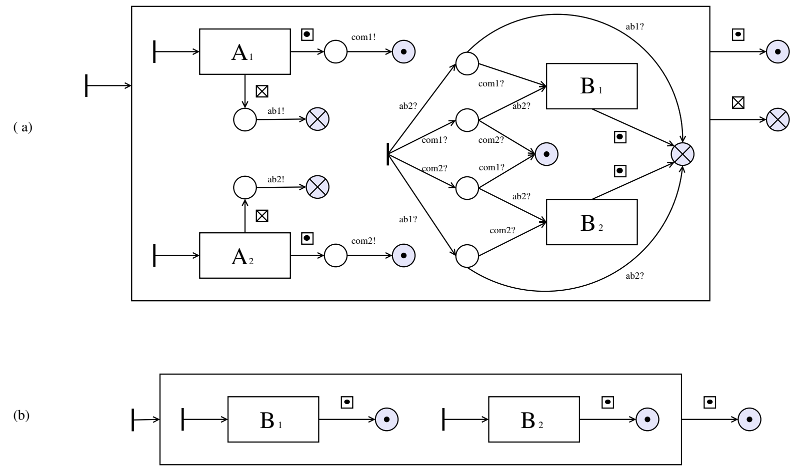

Fig. 6. Pattern for Parallel Transactions.

Definition 7. *Given* $A_1, A_2, B_1, B_2 \in CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$ *we define the parallel composition of activities* A_1 *and* A_2 *with compensations* B_1 *and* B_2 *as the* $CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$ $A = [\![A_1 \Rsh B_1 || A_2 \Rsh B_2]\!]^P = \langle(\emptyset, \emptyset, \{s\}, \{s, q_0 \odot, \otimes\}, q_0, \delta), \mu\rangle$ *with transitions* $\delta = \{(q_0, \tau, true, \emptyset, s), (s, \boxdot, \odot), (s, \boxtimes, \otimes)\}$, *and* $\mu(s) = A' || A'' || C$, *where* A' *and* A'' *are the two CHTTAs depicted in Figure 6 (a) contained in the superstate and referring to activities* A_1 *and* A_2, *and* C *is the* compensation controller *shown on the right part of the superstate. The compound compensation of* A *is defined as the* $CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$ $B = [\![B_1 || B_2]\!]_C^P = \langle(\emptyset, \emptyset, \{s\}, \{s, q_0, \odot, \otimes\}, q_0, \delta'), \mu'\rangle$ *with* $\delta' = \{(q_0, \tau, true, \emptyset, s), (s, \boxdot, \odot)\}$ *and* $\mu'(s) = B' || B''$, *where* B' *and* B'' *are the two CHTTAs in Figure 6 (b) referring to* B_1 *and* B_2 *respectively.*

As for sequential transactions, considering only two activities in the parallel pattern is not a limitation, since the case of n activities may be reduced by iteratively grouping the activities in pairs. For instance, $A = [\![A_1 \Rsh B_1 || A_2 \Rsh B_2 || A_3 \Rsh B_3]\!]^P = [\![A' \Rsh B || A_3 \Rsh B_3]\!]^P$ where $A' = [\![A_1 \Rsh B_1 || A_2 \Rsh B_2]\!]^P$ and B is the compensation for the whole parallel subtransaction A'. Given B_1 and B_2, we define the compensation B of A' as the concurrent execution of the compensations B_1 and B_2 (see Figure 6 (b)).

Theorem 3 (Correct Completion). *Given* $A = [\![A_1^M \Rsh B_1^M || \dots || A_n^M \Rsh B_n^M]\!]^P$, $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu))$ *if and only if,* $w \in \mathcal{L}(A, \Sigma_V)$ *and* $\forall i \in [1, n].\exists!$ $j \in [1, |w|]$. $w[j] = commit_{A_i}!$.

Proof. Since executions of activities A_i and compensations B_i do not interfere, we can assume that all compensations B_i are performed after the commit of activities A_i that terminate successfully.

Hence, the theorem can be reformulated as follows.

Given $A = [\![A_1^M \Rsh B_1^M || \dots || A_n^M \Rsh B_n^M]\!]^P$, $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu))$ if and only if $w \in \mathcal{L}(A, \Sigma_V)$ and there exists a permutation $(i_1, \dots, i_n)$ of $(1, \dots, n)$ such that $w = \tilde{x}_1 \cdot commit_{A_{i_1}}! \cdot \dots \cdot \tilde{x}_n \cdot commit_{A_{i_n}}! \cdot \tilde{x}_{n+1}$ where $\tilde{x}_i \in (\{\tau\} \cup \mathbb{R}^{>0})^*$.

By definition of accepted language, it is obvious that, if $w \in \mathcal{L}(A, \Sigma_V)$, then $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu))$. Hence, we prove only the implication $\Rightarrow$.

We prove by induction on n that, given $c = c_1; \dots; c_n$ with $c_i \not\approx \odot$, if $(A, (c, \nu)) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu'))$, then $w \in \mathcal{L}(A, \Sigma_V)$ and there exists a permutation $(i_1, \dots, i_n)$ of $(1, \dots, n)$ such that $w = \tilde{x}_1 \cdot commit_{A_{i_1}}! \cdot \dots \cdot \tilde{x}_n \cdot commit_{A_{i_n}}! \cdot \tilde{x}_{n+1}$ where $\tilde{x}_i \in \{\{\tau\} \cup \mathbb{R}^{>0}\}^*$.

If $n = 1$ then the thesis holds by Lemma 1. If $n > 1$, then, given a sequence $(A, (c, \nu)) \stackrel{w'}{\Longrightarrow} (A, (c'_1; \dots; c'_n, \nu'))$ such that $c'_k = \odot$ for some $1 \le k \le n$, by Lemma 1, $w' = \tilde{z} \cdot commit_{A_k} \cdot \tilde{z}'$.

Now, since A_{k+1}^M has committed and hence it does not participate in communications, we have that $(A, (c'_1; \dots; c'_n, \nu')) \stackrel{w''}{\Longrightarrow} (A, (\odot, \nu''))$ iff $(A', (c', \nu')) \stackrel{w''}{\Longrightarrow} (A', (\odot, \nu''))$, where $A' = [\![A_1^M \Rsh B_1^M || \dots || A_{k-1}^M \Rsh B_{k-1}^M || A_{k+1}^M \Rsh B_{k+1}^M || \dots || A_n^M \Rsh B_n^M]\!]^P$ and $c' = c'_1; \dots; c'_{k-1}; c'_{k+1}; \dots; c'_n$.

By induction, if $(A', (c', \nu')) \stackrel{w''}{\Longrightarrow} (A', (\odot, \nu''))$, then there exists a permutation $(j_1, \dots, j_{n-1})$ of $(1, \dots, n) \setminus \{k\}$ such that $w'' = \tilde{y}_1 \cdot commit_{A_{j_1}}! \cdot \dots \cdot \tilde{y}_{n-1} \cdot commit_{A_{j_{n-1}}}! \cdot \tilde{y}_n$ where $\tilde{y}_i \in \{\{\tau\} \cup \mathbb{R}^{>0}\}^*$.

Hence, for a fixed $w = w' \cdot w''$ and $(i_1, \dots, i_n) = (k, j_1, \dots, j_{n-1})$ and $\tilde{x}_1 = \tilde{z}$, $\tilde{x}_2 = \tilde{z}' \cdot \tilde{y}_1$, $\tilde{x}_3 = \tilde{y}_2, \dots, \tilde{x}_{n+1} = \tilde{y}_n$ and $\tilde{x}_n = \tilde{y} \cdot \tilde{z}$ and $\tilde{x}_{n+1} = \tilde{z}'$ we have that $(A, (c, \nu)) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu'))$ and $w = \tilde{x}_1 \cdot commit_{A_{i_1}}! \cdot \dots \cdot \tilde{x}_n \cdot commit_{A_{i_n}}! \cdot \tilde{x}_{n+1}$, where $\tilde{x}_i \in (\{\tau\} \cup \mathbb{R}^{>0})^*$. Since $Init(A) = (c_1; \dots; c_n)$ satisfies the condition $c_i \not\approx \odot$, the thesis holds. □

Theorem 4 (Correct Compensation). *Given* $A = [\![A_1^M \Rsh B_1^M || \dots || A_n^M \Rsh B_n^M]\!]^P$, $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\otimes, v))$ *if and only if* $w \in \mathcal{L}(A)$ *and, there exists* $Committed \subset \{A_1, \dots, A_n\}$ *such that* $\forall A_i \notin Committed$ $w[j] \neq commit_{A_i}!$ *and* $\forall A_i \in Committed$ $\exists!\, j \in [1, |w|[$ *such that* $w[j] = commit_{A_i}! \wedge \exists!\, k \in]j, |w|]$ *such that* $w[k] = commit_{B_i}!$.

Proof. The theorem can be reformulated as follows.

Given $A = [\![A_1^M \Rsh B_1^M || \dots || A_n^M \Rsh B_n^M]\!]^P$, $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\otimes, v))$ if and only if $w \in \mathcal{L}(A)$ and there exists a proper subset D of $\{1, \dots, n\}$ such that $w = \tilde{x}_1 \cdot commit_{X_1}! \cdot \dots \cdot \tilde{x}_{|D|} \cdot commit_{X_{|D|}}! \cdot \tilde{x}_{|D|+1} \cdot commit_{X_{|D|+1}}! \cdot \dots \cdot \tilde{x}_{2|D|} \cdot commit_{X_{2|D|}}! \cdot \tilde{x}_{2|D|+1}$ where:

- if $X_i = A_i$, then $i \in D$;
- for any $i \in D$ there exist k and h in $[1, 2|D|]$ with $k < h$ and such that $A_i = X_k$ and $B_i = X_h$;
- for any i, $\tilde{x}_1, \ldots, \tilde{x}_{2|D|+1} \in (\{\tau\} \cup \mathbb{R}^{>0})^*$.

By definition of accepted language, it is obvious that, if $w \in \mathcal{L}(A, \Sigma_V)$, then $(A, Init(A)) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu))$. Hence we prove only the implication $\Rightarrow$.

We prove by induction on the size of D that,

Let c be a configuration of A; we say that c *is well defined* if it holds that A_i is in the state $\odot$ iff B_i is in a state reachable after reading $comm_{A_i}$. With $commit(c)$ we denote the sum of the A_i and B_j that have commit.

Given two well defined configurations c_1 and c_2 we write that $c_1 < c_2$ if $commit(c_1) > commit(c_2)$.

Given a well defined configuration c, we prove by induction on the relation $<$ that if $(A, (c, \nu)) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu'))$, then there exists a proper subset D of $\{i \mid A_i \text{ in } c \text{ is not in the state } \odot\}$ such that $w = \tilde{x}_1 \cdot commit_{X_1}! \cdot \ldots \cdot \tilde{x}_m \cdot commit_{X_m}! \cdot \tilde{x}_{m+1}$ where:

- if $X_i = A_i$, then $i \in D$;
- if $X_i = B_i$ and $i \notin D$, then A_i in c is in the state $\odot$;
- for any $i \in D$ there exist k and h in $[1, m]$ with $k < h$ and such that $A_i = X_k$ and $B_i = X_h$;
- for any i, $\tilde{x}_1, \ldots, \tilde{x}_{m+1} \in (\{\tau\} \cup \mathbb{R}^{>0})^*$.

The base case $c = \odot$ is trivial.

Let us consider now the induction step. By Lemma 1, let c' be a configuration such that $(A, (c, \nu)) \stackrel{w}{\Longrightarrow} (A, (c', \nu'))$ and $w = \tilde{z} \cdot commit_X! \cdot \tilde{z}'$ with $\tilde{z}, \tilde{z}' \in (\{\tau\} \cup \mathbb{R}^{>0})^*$.

By Lemma 1 and since c is well defined, there exists j such that it holds one of the following two cases:

- $X = A_j$ and A_j in c is not in the state $\odot$;
- $X = B_j$ and A_j in c is in the state $\odot$.

In both cases we have that c' is well defined and $c' < c$, hence, by induction, if $(A, (c', \nu')) \stackrel{w}{\Longrightarrow} (A, (\odot, \nu''))$, then there exists a proper subset D' of $\{i \mid A_i \text{ in } c' \text{ is not in the state } \odot\}$ such that $w = \tilde{y}_1 \cdot commit_{X_1}! \cdot \ldots \cdot \tilde{y}_m \cdot commit_{X_{m'}}! \cdot \tilde{y}_{m'+1}$ where:

- if $X_i = A_i$, then $i \in D'$;
- if $X_i = B_i$ and $i \notin D'$, then A_i in c' is in the state $\odot$;
- for any $i \in D$ there exist k and h in $[1, m']$ with $k < h$ and such that $A_i = X_k$ and $B_i = X_h$;
- for any i, $\tilde{y}_1, \ldots, \tilde{y}_{m'+1} \in (\{\tau\} \cup \mathbb{R}^{>0})^*$.

Therefore, for a fixed $D = D' \cup \{j\}$, and $\tilde{x}_1 = \tilde{z}$ and $\tilde{x}_2 = \tilde{z}' \cdot \tilde{y}_1$ and $\tilde{x}_3 = \tilde{y}_2, \ldots, \tilde{x}_{m+1} = \tilde{y}_{m'+1}$ we have that $(A, (c, \nu)) \stackrel{w}{\Longrightarrow} (A, (\otimes, \nu'))$. Now, since $Init(A)$ is well defined, the thesis holds. $\square$

4.3 Long–Running Transactions

Sequential and parallel transactions may be composed in order to define complex transactions. Hence, resorting to the patterns of sequential and parallel transactions, we give the definition of long–running transactions.

Definition 8 (Long–running Transaction). *Given activities* $A_1, \ldots, A_n \in CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$ *and compensations* $B_1, \ldots, B_n \in CHTTA_{\mathcal{A}}^{\Sigma_{Pub}}$*, a* long–running transaction *is given by the following grammar:*

$$T \ ::= \ A_i \Rsh B_i \quad | \quad T \cdot T \quad | \quad T || T.$$

Now, we need to introduce an *encoding* function $[\![\cdot]\!] \rightarrow A \Rsh B$ that takes in input a long–running transaction and returns the CHTTAs A and B where A is the compound CHTTA modeling the transaction and B its compensation. We define the function $[\![\cdot]\!]$ recursively as follows:

- $[\![A_i \Rsh B_i]\!] = A_i \Rsh B_i$,
- $[\![T_1 \cdot T_2]\!] = [\![A_1 \Rsh B_1 \cdot A_2 \Rsh B_2]\!]^S \Rsh [\![B_1 \cdot B_2]\!]_C^S$, where $A_i \Rsh B_i = [\![T_i]\!]$ for $i \in [1, 2]$,
- $[\![T_1 || T_2]\!] = [\![A_1 \Rsh B_1 || A_2 \Rsh B_2]\!]^P \Rsh [\![B_1 || B_2]\!]_C^P$, where $A_i \Rsh B_i = [\![T_i]\!]$ for $i \in [1, 2]$.

Since the building blocks of the encoding function are the patterns of sequential and parallel transactions, the correctness of $[\![\cdot]\!]$ is given by Theorems 1– 4.

Given a long–running transaction T, we define the *top–level* of T (denoted $top(T)$) as the CHTTA A such that $[\![T]\!] = A \Rsh B$.

Modeling long–running transactions with CHTTAs allows verifying properties by model checking. In fact, given a long–running transaction T obtained as in Definition 8, and a set of visible actions Σ_V, we may flatten the CHTTA $top(T)$ according to Definition 5, and then verify properties of the transaction by model checking on the timed automaton $Flat(top(T), \Sigma_V)$.

5 Case Study: A Double Request

We model a typical all–or–nothing scenario in which a client performs two concurrent requests to two different servers, waits for replies, and sends back acknowledgements either to both servers (if it receives both replies) or to none of them (if it receives at most one reply). A similar scenario in a realistic context is given in [11], where a typical e–commerce application is described in which a customer of an on–line shop orders two products which are provided by two different stores. In that case, acknowledgements are sent (and products are bought) only if both products are available, instead, in our case, acknowledgements are sent only if replies are received before given times.

A single request/reply activity performed by the client is described by the transaction given in Figure 7 (a). We denote such a transaction with $A_i \Rsh B_i$. The client sends the request to the server by synchronizing on channel *req_i* and waits for the reply as a synchronization on channel *rep_i*. The time deadline for the reply is T_i. This is expressed as a constraint on the value of clock x_i which is set

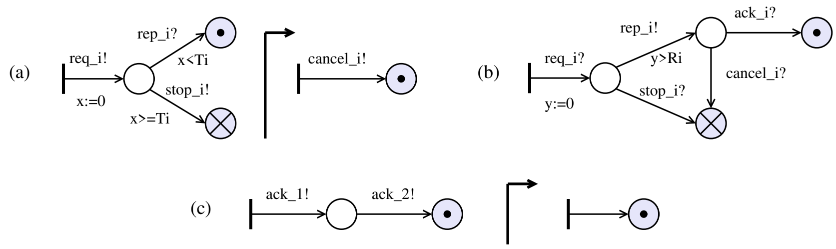

Fig. 7. A Double Request.

to zero when the request is sent. If the reply is received in time, the transaction commits, otherwise a stop message is sent to the server as a synchronization on channel *stop_i*, and the transaction is aborted. The compensation of this transaction consists in a synchronization on channel *cancel_i*, which corresponds to sending an undo message to the server.

A server is modeled by the automaton given in Figure 7 (b). We denote such an automaton with S_i. The server receives a request and sends the reply by synchronizing on the proper channels, and it spends a time between these two synchronizations which is greater than R_i. This amount of time models the time spent by the server to satisfy the request of the client. Then, the server reaches a state in which it waits for either an acknowledge or an undo message from the client. These two communications are modeled as synchronizations on channels *ack_i* and *cancel_i*, respectively, and lead to commit and abort of the server activity, respectively.

The activity of sending acknowledgments to two servers S_1 and S_2 is modeled by the transaction given in Figure 7 (c). We denote such a transaction with $A_{ack} \Rsh B_{ack}$. Finally, the whole client transaction in which two requests are sent to two different servers and the corresponding acknowledgments are sent if both requests are satisfied, is modeled by the long–running transaction $T = (A_1 \Rsh B_1 || A_2 \Rsh B_2) \cdot A_{ack} \Rsh B_{ack}$ and the whole system in which both the client and the two servers are modeled is $SYSTEM = T||S_1||S_2$.

To verify properties of this system, we consider the CHTTA $top(T)$, and then we compute the flat TTA $T' = Flat(top(T), \Sigma_V)$, where $\Sigma_V = \{a!, a? \mid a \in \{req_i, rep_i, stop_i, cancel_i, ack_i\}\}$. Now, since S_1 and S_2 are both flat, we have that $T'||S_1||S_2$ can be used as an input for the UPPAAL model checker. In order to reduce the size of the model we remove unnecessary τ transactions, and in order to avoid the execution of paths containing an infinite sequence of timed transitions we include time invariants in the states of the automaton modeling the client.

In Table 1 we show the results of the model checking. We have verified eight properties, and each property has been verified three times: once by setting both timeouts T_1 and T_2 greater than R_1 and R_2, respectively, once by setting $T_1 < R_1$ and $T_2 > R_2$, and once by setting both T_1 and T_2 smaller than R_1 and R_2, respectively.

Properties are expressed as logical formulas using the operators accepted by the UPPAAL model checker. A logical formula may have one of the following

	$T_1 > R_1$ $T_2 > R_2$	$T_1 < R_1$ $T_2 > R_2$	$T_1 < R_1$ $T_2 < R_2$
1. $A\Diamond(T.\odot \vee T.\otimes)$	true	true	true
2. $(A_1.\otimes \vee A_2.\otimes) \leadsto T.\otimes$	true	true	true
3. $(A_1.\odot \wedge A_2.\odot) \leadsto T.\odot$	true	true	true
4. $T.\odot \leadsto (S_1.\odot \wedge S_1.\odot)$	true	true	true
5. $x_1 \geq T_1 \leadsto T.\otimes$	true	true	true
6. $x_2 \geq T_2 \leadsto T.\otimes$	true	true	true
7. $E\Diamond T.\odot$	true	false	false
8. $E\Diamond T.\otimes$	true	true	true

Table 1. Results of the model checking.

forms: $E\Diamond\phi, E\Box\phi, A\Diamond\phi, A\Box\phi$ and $\phi \leadsto \psi$, where ϕ and ψ are state formulas, namely conditions which could be satisfied by a state. In particular: $E\Diamond\phi$ represents reachability: it asks whether ϕ is satisfied by some reachable state; $E\Box\phi$ says that there should exists a maximal path such that ϕ is always true; $A\Diamond\phi$ says that ϕ is eventually satisfied in all paths; $A\Box\phi$ expresses that ϕ should be true in all reachable states; finally, $\phi \leadsto \psi$ means that whenever ϕ is satisfied, then eventually (in the continuation of the path) ψ will be satisfied.

Properties 1–3 express the correctness of the encoding of long–running transactions into automata. These properties must be satisfied for any setting of the parameters. In particular, property 1 says that either the commit or the abort states of the transaction (denoted $T.\odot$ and $T.\otimes$, respectively) must be eventually reached. Property 2 requires that if at least one of the abort states of the parallel activities A_1 and A_2 is reached, then the whole transaction must reach its abort state, and property 3 requires that if both parallel activities A_1 and A_2 reach their commit states, then the whole transaction must reach its commit state.

Properties 4–7 express the correctness of the modeling of the scenario. As before, these properties must be satisfied for any setting of the parameters. Property 4 says that if the transaction reaches a commit state, then eventually both servers must reach their commit states. Properties 5 and 6, instead, say that if one of the two clocks of the parallel activities A_1 and A_2 becomes greater than its deadline, then the whole transaction must reach its abort state.

Finally, properties 8 and 9 express that the commit and abort states of the transaction can be reached, for different settings of the parameters. In particular, the commit state can be reached only if both the timeouts T_1 and T_2 are greater than the times R_1 and R_2 spent by the two servers. The abort state, instead, can be reached with any setting of the parameters. This is true because R_1 and R_2 are lower bounds, hence a server may spend more time than its minimum time, and may exceed the corresponding deadline in the transaction.

6 Conclusions

We studied some pattern for the composition of activities in long–running transactions. In particular we focused our attention on the sequential and parallel pattern. In [6] another pattern is identified allowing to deal with *nested transactions*. Intuitively, a nested transaction is composed by a hierarchy of subtransactions as activities. In the nested pattern, the top–level transaction completes its activity when all its sub–transactions terminate. When a transaction aborts, all its subtransactions should abort, and the committed subtransactions should be compensated. Nevertheless, a top–level transaction can commit even though some of its subtransactions have aborted. In [6] the compensation pattern for nested transactions is defined by resorting to a stack where the compensations of each subtransactions are stored when the related activities commit. If, at some point, the supertransactions needs to be compensated, compensations of the subtransactions are invoked from the stack.

With the model of CHTTAs given in this paper, we may represent the pattern of nested transaction by defining a compensation controller which should be put in parallel with the top–level transaction. While the patterns for sequential and parallel transactions are expressed in a rather natural way by CHTTAs, it is not so for the latter mechanism. Hence, we plan to enrich the model of CHTTAs with a notion of memory to store compensations of committed subtransactions.

References

1. R. Alur and D. L. Dill. “A Theory of Timed Automata”. Theoretical Computer Science, volume 126, pages 183–235, 1994.
2. R. Alur, S. Kannan, and M. Yannakakis. “Communicating Hierarchical State Machines”. ICALP’99, LNCS 1644, pages 169–178, 1999.
3. T. Amnell, G. Behrmann, J. Bengtsson, P. R. D’Argenio, A. David, A. Fehnker, T. Hune, B. Jeannet, K. G. Larsen, M. O. Moeller, P. Pettersson, C. Weise, and W. Yi. “Uppaal-now, next and future”. LNCS 2067, pages 99–124, 2000.
4. B. Benatallah and R. Himadi. “A Petri Net–Based Model for Web Service Composition”. ADC’03, Australian Computer Society, pages 191–200, 2003.
5. A. Brogi, C. Canal, E. Pimentel, and A. Vallecillo. “Formalizing Web Services Choreographies”. WS–FM’04, ENTCS 105, pages 73-94, 2004.
6. R. Bruni, H. Melgratti, and U. Montanari. “Theoretical Foundations for Compensations in Flow Composition Languages”. POPL’05, ACM Press, pages 209–220, 2005.
7. H. Garcia–Molina and K. Salem. “Sagas”. SIGMOD’87, ACM Press, pages 249–259, 1987.
8. BPEL Specifications: `www-128.ibm.com/developerworks/library/ws-bpel/`.
9. I. Houston, M.C. Little, I. Robinson, S. K. Shrivastava, and S. M. Wheater. “The CORBA Activity Service Framework for Supporting Extended Transactions”. Software — Practice and Experience, volume 33, number 4, pp. 351–373, 2003.
10. C. Laneve and G. Zavattaro. “Foundations of Web Transactions”. FOSSACS’05, LNCS 3441, pp. 282–298, 2005.
11. M. Mazzara and S. Govoni. “A Case Study of Web Services Orchestration.”. COORDINATION’05, LNCS 3454, pp. 1–16, 2005.
12. WSCI Specification. Version 1.0. Available at `http://www.w3.org/TR/wsci/`.
13. M. Viroli. “Towards a Formal Foundation to Orchestration Languages”. WS–FM’04, ENTCS 105, pages 51–71, 2004.
14. S. Yovine. “Kronos: A verification tool for real-time systems”. International Journal on Software Tools for Technology Transfer, volume 1, pages 123–133, 1997.