Fundamenta Informaticae 42 (2000) 1–21
IOS Press

Timed Cooperating Automata*

Ruggero Lanotte and Andrea Maggiolo-Schettini†
Dipartimento di Informatica,
Università di Pisa,
Corso Italia 40, 56125 Pisa, Italy
E_mail: lanotte@di.unipi.it
E_mail: maggiolo@di.unipi.it

Adriano Peron‡
Dipartimento di Matematica e Informatica,
Università di Udine,
Via delle Scienze 206, 33100 Udine, Italy
E_mail: peron@dimi.uniud.it

Abstract. We propose Timed Cooperating Automata (TCAs), an extension of the model Cooperating Automata of Harel and Drusinsky, and we investigate some basic properties. In particular we consider variants of TCAs based on the presence or absence of internal activity, urgency and reactivity, and we compare the expressiveness of these variants with that of the classical model of Timed Automata (TAs) and its extensions with periodic clock constraints and with silent moves. We consider also closure and decidability properties of TCAs and start a study on succinctness of their variants with respect to that of TAs.

Keywords: Timed automata, Expressiveness, Closure properties, Succinctness.

*Research partially supported by CNR Progetto Strategico "Modelli e Metodi per la Matematica e l'Ingegneria", by MURST Progetto Cofinanziato "Tecniche Formali per la Specifica, l'Analisi, la Verifica, la Sintesi e la Trasformazione di Sistemi Software", and by MURST Progetto Cofinanziato TOSCA.
†Address for correspondence: Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, Italy
‡Address for correspondence: Dipartimento di Matematica e Informatica, Università di Udine, Via delle Scienze 206, 33100 Udine, Italy

1. Introduction

In [1] Alur and Dill propose Timed Automata (*TA*s) to model the behaviour of real-time systems over time. The behaviour of a system is described, in an abstract way, in terms of acceptance or non acceptance of timed infinite (ω)-sequence. (A timed sequence is a sequence of symbols of a given alphabet where each symbol is annotated with a time value taken from a dense time domain.) A Timed Automaton maintains an ongoing interaction with the environment by reacting to the sequence of environment prompts represented in the form of a timed (ω)-sequence. Reacting to an environment prompt means performing a transition. Timed Automata are *reactive*, since they must have the ability to properly respond to each prompt of the environment, and they are *urgent*, since a reaction is accomplished instantaneously without any unmotivated delay. Recently, two extensions of *TA* have been proposed, one (see [5]) permitting a stronger form of control over time (periodic clocks) -denoted TA_p-, and the other (see [3]) permitting silent moves -denoted TA_ϵ-. (Also restrictions of *TA*s have been considered in [2, 7].)

Timed Automata are inherently sequential and lack a way for explicitly representing parallelism and communication. As shown in [6], an advantage of representing parallelism and cooperation is the gain in succinctness, that is the inherent size of the automaton required to accept a given language. In [6], a model, called Cooperating Automaton (*CA*), is introduced, which represents parallelism and communication while remaining as close as possible to classical finite automata. Besides succinctness, another advantage of this model is that it comes close to the successful specification language Statecharts [8]. In fact, *CA*s correspond to Statecharts devoid of hierarchy and consisting of a single collection of orthogonal components, each of which is merely a finite automaton which evolves synchronously with respect to the other components.

In this paper we propose a model, called *Timed Cooperating Automaton* (*TCA*), which is a twofold extension of *CA*. First, we consider transitions guarded with suitable timing constraints. Second, we extend the notion of transition step, which traditionally consists of one only transition firing (as it is also the case of *TA*), by permitting transitions steps consisting of sequences of transitions. Sequences of transitions can be interpreted as an activity which is triggered by the environment but may go on independently and take a non-null time for its completion (*internal activity*). A reason for considering internal activity is that it is permitted in many specification formalisms (see e.g Esterel [4] and some variants of Statecharts [10, 12]).

In this paper, we study the impact of internal activity, urgency and reactivity over the expressive power of *TCA*s, and we compare the model and its variants, which we are proposing, with the mentioned models *TA*, TA_p and TA_ϵ. We show that both the presence of internal activity and (independently) the absence of urgency increase the expressive power of the formalism. As far as the comparison with *TA*, TA_p and TA_ϵ, we show that the absence of internal activity makes the expressive power of *TCA*s comparable to that of *TA*s. By permitting internal activity we have that *TCA*s have at least the same power of TA_ps, and by releasing urgency we have that *TCA*s have at least the same power of TA_ϵs. The role played by reactivity is not at the moment completely clear to us. In the paper we begin also a study of succinctness properties of the model presented with respect to *TA*s and we show that *TCA*s may give an exponential saving in the representation. It is our intention to use TCAs as an intermediate format (in the line of [11]) in which (Timed) Statecharts can be naturally translated and where interesting properties, such as refinement and retiming, can be investigated more easily than in the higher-level formalism. The present paper is a revised and extended version of [9]. The paper is organised as follows. In section 2 we introduce the model of Timed Cooperating Automata and its variants. In section 3, after recalling the definitions of TA, TA_p, and TA_ϵ, we compare the expressive power of the variants of Timed Cooperating Automata with the three classes of automata mentioned. In section 4 we discuss the issues of closure under boolean operations and of decidability of emptiness. In section 5 we deal with succinctness of TCAs.

2. Timed Cooperating Automata

A timed sequence is a sequence of symbols (from a finite alphabet Σ), where each symbol is annotated with a time value taken from a dense time domain $Time$ (non-negative rational numbers, for instance). In [1] a timed sequence is presented as a pair $\alpha = \langle \alpha_1, \alpha_2 \rangle$ consisting of an ω-sequence α_1 over Σ (the *untimed version of the sequence*) and an ω-sequence α_2 over $Time$, where it is intended that the i-th symbol of α_1 (also written $\alpha_1(i)$) occurs at time $\alpha_2(i)$. Timed sequences have to fulfill the two restrictions of *monotonicity* and *progress*:

monotonicity $\alpha_2(i) \leq \alpha_2(i+1)$, for all $i \geq 0$;
progress for every $\tau \in Time$ there is i such that $\alpha_2(i) \geq \tau$.

Monotonicity and progress permit to interpret a timed sequence as the description of an ongoing interaction betweeen the environment which prompts a system with symbols communicated instantaneously and a system which is able (or is not able) to react to such prompts.

In this work we extend the notion of timed sequence by considering sequences of elements of the powerset of Σ (sets of symbols can be communicated by the evironment instead of singletons). Moreover, we assume that symbols can be communicated continuously during intervals of time. To this purpose, we extend a timed sequence by an ω-sequence α_3 over $Time$, intending that signals in $\alpha_1(i)$ are communicated continuously during the interval $[\alpha_2(i), \alpha_2(i) + \alpha_3(i)]$.

Therefore, an *environment* is a triple

$$\alpha = \langle \alpha_1, \alpha_2, \alpha_3 \rangle, \text{ where:}$$

1. $\alpha_1 : \mathbb{N} \longrightarrow 2^\Sigma$ gives the set of events communicated at each interaction;
2. $\alpha_2 : \mathbb{N} \longrightarrow Time$ gives the time of each interaction and satisfies the requirements of monotonicity and progress;
3. $\alpha_3 : \mathbb{N} \longrightarrow Time$ gives the duration of communication during interaction and satisfies the requirement that communication intervals do not overlap (but possibly in their borders), namely $\alpha_2(i) + \alpha_3(i) \leq \alpha_2(i+1)$.

We assume that the interaction with the environment starts at time 0, and that, for any environment α, $\alpha_1(0) = \emptyset$, $\alpha_2(0) = 0$, and $\alpha_3(0) = 0$. One can interpret this first interaction as the swiching on of the system. The meaningful interaction with the environment is then given by $\alpha^+ = \langle \alpha_1^+, \alpha_2^+, \alpha_3^+ \rangle$, where α_i^+, for $i \in \{1,2,3\}$, is the restriction of α_i to positive natural numbers.

We briefly recall the idea of a Cooperating Automaton. A *CA* consists of n automata each with its own set of states, an initial state and a transition table. These automata work together in a synchronous manner, taking transitions according to the common input symbol being read, their internal states and a set of propositions over atomic formulas having the form "component i currently enters state q" (*condition formulas*). Condition formulas are interpreted to take on truth values according to the states of possibly all the components, thus permitting cooperation among them. In the extension of *CA* we are proposing, condition formulas impose constraints not only on the set of current states but also on the relative time of entering or exiting a state.

Let Π_Σ denote the collection of propositional formulas over Σ (*environment conditions*), inductively defined as follows:

1. $true, \Sigma \subseteq \Pi_\Sigma$;
2. $p_1 \wedge p_2$, $p_1 \vee p_2$, $\neg p_1$ are in Π_Σ for p_1 and p_2 in Π_Σ.

For an alphabet of state symbols Q, let Γ_Q denote the collection of *internal conditions* inductively defined as follows:

1. $true, q = \tau$, $q[\tau]$ and $q\{\tau\}$ are in Γ_Q with $q \in Q$ and $\tau \in Time$;
2. $p_1 \wedge p_2$, $p_1 \vee p_2$, $\neg p_1$ are in Γ_Q for p_1 and p_2 in Γ_Q.

As well as cooperating automata, a *TCA* is the parallel composition of sequential automata. In order to generalize the standard notion of step (consisting of one only transition for each sequential component) to a notion of step admitting sequences of transitions, we distinguish between *input states* (starting and ending an internal activity) and *non-input states* (intermediate states of an internal activity). The acceptance condition is a Büchi acceptance condition (see [13]).

Definition 2.1. A *Timed Cooperating Automaton* (*TCA*) is a tuple

$$M = \langle M_1, \dots, M_n, \mathcal{I}, \mathcal{F} \rangle, \text{ where:}$$

1. each *sequential automaton* M_i (with $1 \leq i \leq n$) is a triple $\langle Q_i, q_i^0, \delta_i \rangle$ such that
   (a) Q_i is a finite set of *states* ($Q_1, \dots, Q_n$ are required to be pairwise disjoint and Q is $\bigcup_{1 \leq i \leq n} Q_i$);
   (b) $q_i^0 \in Q_i$ is the *initial state*;
   (c) $\delta_i \subseteq Q_i \times \Pi_\Sigma \times \Gamma_Q \times Q_i$ is the *transition relation*;
2. the set $\mathcal{I}$ of *input states* is such that $\bigcup_{1 \leq i \leq n} \{q_i^0\} \subseteq \mathcal{I} \subseteq Q$;
3. $\mathcal{F} \subseteq \mathcal{I}$ is the set of *accepting states*.

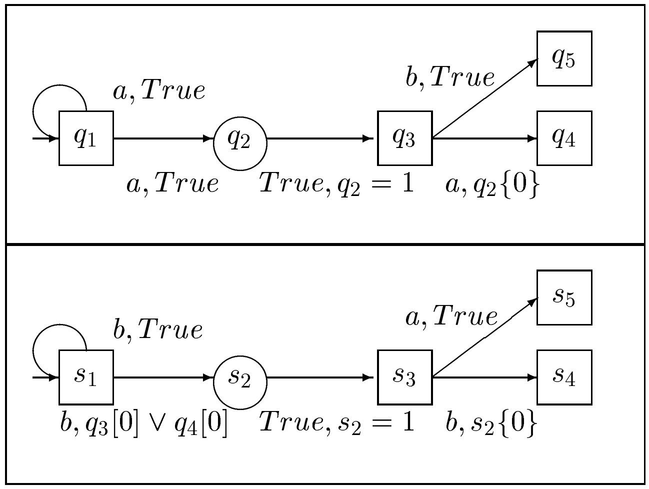

Figure 1. An example of TCA.

A TCA_I is a TCA with only input states, i.e. $\mathcal{I} = Q$. A *Sequential Timed Cooperating Automaton* ($STCA$) is a TCA consisting of only one sequential automaton. An example of TCA is given in Fig.1. This automaton consists of two sequential automata. Boxes are input states and circles represent states which are non-input states. Initial states are depicted as states associated with a dangling arc.

A *local configuration* for a sequential automaton M_i is a tuple

$$\mathcal{C}_i = \langle q_i, in_i, out_i, \sigma_i \rangle, \text{ where:}$$

1. $q_i \in Q_i$ is the *enabled state* of M_i;
2. $in_i : Q_i \longrightarrow Time$ is the *state enabling* (partial) function;
3. $out_i : Q_i \longrightarrow Time$ is the *state disabling* (partial) function;
4. $\sigma_i \in Time$ is the *local time*.

The enabling function $in_i(q)$ gives, whenever it is defined, the time when the state $q \in Q_i$ has most recently been enabled. If $in_i(q)$ is undefined, then q has never been enabled.
The disabling function $out_i(q)$ gives, whenever it is defined, the time when a state $q \in Q_i$ has most recently been disabled (whenever it has been previously enabled, i.e. whenever $in_i(q)$ is defined).
If $in_i(q)$ is defined and either $out_i(q)$ is undefined or $out_i(q) < in_i(q)$, then q is currently enabled. Viceversa, if $in_i(q)$ and $out_i(q)$ are both defined and $in_i(q) \leq out_i(q)$, then q is currently disabled.
A local configuration $\mathcal{C}_i = \langle q_i, in_i, out_i, \sigma_i \rangle$ is *an input configuration* if $q_i \in \mathcal{I}$, and it is *initial* if $q_i = q_i^0$, $\sigma_i = 0$, $in_i(q_i^0) = 0$ (with in_i undefined elsewhere) and out_i is undefined for any $q \in Q_i$. Such a configuration is denoted by $\mathcal{C}_i^0$.
A *(global) configuration* $\mathcal{C}$ is a tuple $\langle \mathcal{C}_1, \ldots, \mathcal{C}_n, \alpha, m \rangle$, where:

1. $\mathcal{C}_i$ is a local configuration, for $1 \leq i \leq n$;

2. α is an environment;
3. $m \in \mathbb{N}$ is the number indicating that $\mathcal{C}$ is concerned with the m-th interaction with the environment, namely $\langle \alpha_1(m), \alpha_2(m), \alpha_3(m) \rangle$.

Let $\mathcal{C} = \langle \mathcal{C}_1, \ldots, \mathcal{C}_n, \alpha, m \rangle$ be a configuration with $\mathcal{C}_i = \langle q_i, in_i, out_i, \sigma_i \rangle$. An atomic proposition $true, q[\tau]$, $q\{\tau\}$ and $q = \tau$ with $q \in Q_i$, is evaluated to $-$ (i.e. undefined) in $\mathcal{C}$ at time σ, if $\sigma_i > \sigma$.
An atomic proposition $q[\tau]$ with $q \in Q_i$ is evaluated to $True$ at time σ in $\mathcal{C}$ iff

1. $\sigma_i \leq \sigma$;
2. $q = q_i$ and $in_i(q) \leq \sigma - \tau$,

namely if state q is currently enabled and it has been enabled at least since a time τ.
An atomic proposition $q\{\tau\}$ with $q \in Q_i$ is evaluated to $True$ at time σ in $\mathcal{C}$ iff

1. $\sigma_i \leq \sigma$;
2. either $\sigma - \tau \leq out_i(q) \leq \sigma$ or $q = q_i$,

namely if state q is currently enabled or it has been disabled since at most a time τ.
An atomic proposition $q = \tau$ with $q \in Q_i$ is evaluated to $True$ at time σ in $\mathcal{C}$ iff

1. $\sigma_i \leq \sigma$;
2. $q = q_i$ and $in_i(q) = \sigma - \tau$,

namely if state q has been enabled exactly since a time τ.
In all of the remaining cases the above mentioned atomic propositions evaluate to $False$ in a configuration $\mathcal{C}$ at time σ. Given the evaluation of atomic formulas, a formula $\gamma \in \Gamma_Q$ is evaluated in a configuration $\mathcal{C}$ at a time σ by exploiting the standard semantics of connectives $\wedge$, $\vee$ and $\neg$ over the truth-values $True$, $False$ and $-$.
An environment proposition $\pi \in \Pi$ is evaluated in an environment α at time σ and at the m-th interaction with the environment by interpreting to $True$ all the symbols in the set

$$\{true\} \cup \{a : a \in \alpha_1(m), \alpha_2(m) \leq \sigma \leq \alpha_2(m) + \alpha_3(m)\},$$

and to $False$ all the other symbols. Given a configuration $\mathcal{C} = \langle \mathcal{C}_1, \ldots, \mathcal{C}_n, \alpha, m \rangle$, there is a *local derivation* from $\mathcal{C}_i = \langle q_i, in_i, out_i, \sigma_i \rangle$ to $\mathcal{C}'_i = \langle q'_i, in'_i, out'_i, \sigma'_i \rangle$, written $\mathcal{C}_i \rightarrow_{\mathcal{C}} \mathcal{C}'_i$, if

1. $\sigma'_i \geq \alpha_2(m)$;
2. $\langle q_i, \pi, \gamma, q'_i \rangle \in \delta_i$ for some π and γ;
3. $in'_i(q'_i) = \sigma'_i$ and $in'_i(q) = in_i(q)$ for $q'_i \neq q$;
4. $out'_i(q_i) = \sigma'_i$ and $out'_i(q) = out_i(q)$ for $q_i \neq q$;
5. π evaluates to $True$ in α at time σ'_i and interaction m;
6. γ evaluates to $True$ in $\mathcal{C}$ at time σ'_i.

We say that σ'_i is the time of the local derivation $\mathcal{C}_i \rightarrow_{\mathcal{C}} \mathcal{C}'_i$. The local derivation $\mathcal{C}_i \rightarrow_{\mathcal{C}} \mathcal{C}'_i$ is *urgent* if does not exist a local derivation $\mathcal{C}_i \rightarrow_{\mathcal{C}} \mathcal{C}''_i$ at time σ'' with $max\{\sigma_i, \alpha_2(m)\} \leq \sigma'' < \sigma'_i$.

The behaviour of a sequential component of the automaton consists of a sequence of local derivations. While the configuration of the considered component changes, the local configurations of all the other components are assumed to be fixed.

A *local step* from a configuration $\mathcal{C} = \langle \mathcal{C}_1, \ldots, \mathcal{C}_n, \alpha, m \rangle$ is a sequence of local derivations $\mathcal{C}_i \rightarrow_{\mathcal{C}} \mathcal{C}_i^1 \rightarrow_{\mathcal{C}^1} \ldots \rightarrow_{\mathcal{C}^{v-1}} \mathcal{C}_i^v \rightarrow_{\mathcal{C}^v} \mathcal{C}'_i$, written $\mathcal{C}_i \Rightarrow_{\mathcal{C}} \mathcal{C}'_i$, where

1. $\mathcal{C}_i$ and $\mathcal{C}'_i$ are input local configurations and $\mathcal{C}_i^j$ is not an input local configuration for any $1 \leq j \leq v$;
2. $\mathcal{C}^j$ is $\langle \mathcal{C}_1, \ldots, \mathcal{C}_{i-1}, \mathcal{C}_i^j, \mathcal{C}_{i+1}, \ldots, \mathcal{C}_n, \alpha, m \rangle$.

A local step is *urgent* if it consists of urgent local derivations. A local step $\mathcal{C}_i \Rightarrow_{\mathcal{C}} \mathcal{C}'_i$ with $\mathcal{C}_i = \mathcal{C}'_i$ is said to be a *ghost* local step.

The behaviour of an automaton is a concurrent composition of local steps which guarantees causality and maximality. The local step of a component is performed with respect to the local configurations of the other components which are either in the starting configurations or in the configurations reached in the same step, provided that a causal justification can be given.

Given two configurations $\mathcal{C} = \langle \mathcal{C}_1, \ldots, \mathcal{C}_n, \alpha, m \rangle$ and $\mathcal{C}' = \langle \mathcal{C}'_1, \ldots, \mathcal{C}'_n, \alpha, m+1 \rangle$, there is a *step* from $\mathcal{C}$ to $\mathcal{C}'$, written $\mathcal{C} \Rightarrow \mathcal{C}'$, whenever

1. for any $1 \leq i \leq n$, if $\mathcal{C}_i \neq \mathcal{C}'_i$ or $\mathcal{C}_i = \mathcal{C}'_i$ and there exists a ghost local step $\mathcal{C}_i \Rightarrow_{\mathcal{C}} \mathcal{C}_i$, then there exists a local step $\mathcal{C}_i \Rightarrow_{\overline{\mathcal{C}}_i} \mathcal{C}'_i$ with $\overline{\mathcal{C}}_i = \langle \mathcal{C}_1^\star, \ldots, \mathcal{C}_i, \ldots, \mathcal{C}_n^\star, \alpha, m \rangle$ and either $\mathcal{C}_j^\star = \mathcal{C}_j$ or $\mathcal{C}_j^\star = \mathcal{C}'_j$, with $1 \leq j \leq n$; in this case, we define a relation $\rightarrow_i \subseteq \{0, \ldots, n\} \times \{1, \ldots, n\}$ such that $\rightarrow_i = \{\langle j, i \rangle : \mathcal{C}_j^\star \neq \mathcal{C}_j, j \neq i\} \cup \{\langle 0, i \rangle : \mathcal{C}_j^\star = \mathcal{C}_j, j \neq i\}$. The relation $\bigcup_i \rightarrow_i$ is required to be a partial order with least element 0 (causal justification);
2. if $\mathcal{C}_i = \mathcal{C}'_i$, then either there is a ghost local step or there is no local step $\mathcal{C}_i \Rightarrow_{\mathcal{C}'} \mathcal{C}'_i$ having local time $\alpha_2(m) \leq \sigma'_i \leq \alpha_2(m+1)$ (maximality).

A step is *urgent* if it consists of urgent local steps only.

A step is *reactive* if each local step $\mathcal{C}_i \Rightarrow_{\overline{\mathcal{C}}_i} \mathcal{C}'_i$ has final local time $\sigma'_i \leq \alpha_2(m+1)$ and the step is maximal with respect to the local steps satisfying such a constraint (namely if the step terminates before the next interaction with the environment). To achieve reactivity, local steps which violate the reactivity condition, are aborted.

A *run* is an ω-sequence of global configurations $\mathcal{C}^0, \mathcal{C}^1, \ldots, \mathcal{C}^i, \ldots$ such that $\mathcal{C}^0$ is an initial configuration and $\mathcal{C}^j \Rightarrow \mathcal{C}^{j+1}$, for any $j \geq 0$.

A run is urgent (resp. reactive) if its steps are urgent (resp. reactive). Given a run $\mathcal{R}$, let $Inf(\mathcal{R})$ be the set of input states which occur infinitely often in $\mathcal{R}$. A run $\mathcal{R}$ is successful if $Inf(\mathcal{R}) \cap \mathcal{F} \neq \emptyset$ (Büchi acceptance condition). An environment α is *accepted* by a TCA (resp. TCA_U, TCA_R, TCA_{RU}) if there exists a successful run (resp. successful urgent run, successful reactive run, successful urgent and reactive run) starting from the initial configuration with environment α.

The *language accepted* by a TCA M, written $L(M)$, is the set $\{\alpha^+ : \alpha \text{ is accepted by } M\}$.

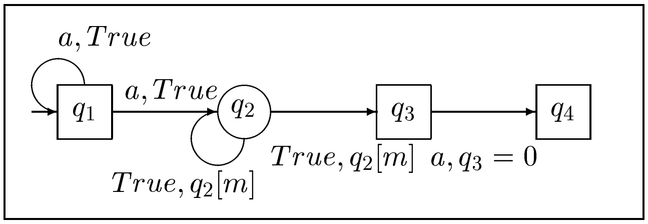

Figure 2. The *TCA* of Example 2.2.

Example 2.1. The language accepted by the TCA_{RU} of Fig.1 (taking $\mathcal{F} = \{s_4\}$) consists of words of the form $\langle \alpha_1, \alpha_2, \alpha_3 \rangle$ with

1. $\alpha_1(i) \subseteq \{a, b\}$ (for $i \geq 0$);
2. there are $j \geq 0$ and $k \geq j + 1$ such that

$$a \in \alpha_1(j),\ a \in \alpha_1(j+1),\ b \in \alpha_1(k) \text{ and } b \in \alpha_1(k+1);$$
$$\alpha_2(j+1) = \alpha_2(j) + 1 \text{ and } \alpha_2(k+1) = \alpha_2(k) + 1.$$

Example 2.2. Let us consider the automaton M of Fig.2 (taking $\mathcal{F} = \{q_4\}$). If M is reactive and urgent (i.e. M is a TCA_{RU}), then the language $L(M)$ accepted by M is

$$\{\langle a^\omega, \alpha_2, \alpha_3 \rangle : \ \alpha_2(i+1) - \alpha_2(i) = km, \ \text{for some } i \geq 0, k > 0\}.$$

If M is not reactive but urgent (i.e. M is a TCA_U), then the language $L(M)$ is

$$\{\langle a^\omega, \alpha_2, \alpha_3 \rangle : \ \alpha_2(j) - \alpha_2(i) = km, \ \text{for some } 0 \geq i > j, k > 0\}.$$

If M is not urgent but reactive (i.e. M is a TCA_R), then the language $L(M)$ is

$$\{\langle a^\omega, \alpha_2, \alpha_3 \rangle : \ \alpha_2(i+1) - \alpha_2(i) > m, \ \text{for some } i \geq 0\}.$$

Finally, if M is neither urgent nor reactive (i.e. M is a *TCA*), then the language $L(M)$ is

$$\{\langle a^\omega, \alpha_2, \alpha_3 \rangle : \ \alpha_2(j) - \alpha_2(i) > m, \ \text{for some } 0 \geq i > j\}.$$

In the following sections we discuss *TCA*s from the point of view of expressiveness, closure with respect to boolean operations and succinctness.

3. Expressiveness of *TCA*s

In this section we compare the expressive power of *TCA*s with respect to the expressive power of three classes of automata: Timed Automata *TA*s, Periodic Timed Automata TA_ps (see [5]) and Timed Automata with silent moves TA_ϵs (see [3]). The section is organised as follows: we first recall the definition of the three above mentioned classes of automata and then we present the expressiveness results for *TCA*s.

3.1. Preliminaries

Timed Automata (*TA*s)

We briefly recall the definition of a Timed Automaton (*TA*)). Timed Automata are state-transition diagrams annotated with timing constraints over real-valued clocks. When an untimed automaton makes a state-transition, the choice of the next state depends on the input symbol read. In case of a timed automaton, the choice depends also on the time of the input symbol relatively to the times of the symbols previously read. This is done by associating a finite set of clocks with the automaton. A clock can be set to zero simultaneously with any transition. At any instant, the reading of the clock equals the time elapsed since the last time it was reset. A transition may be taken only if the current values of the clocks satisfy a timing constraint associated with the transition. For a set of clocks C, let $\Phi(C)$ be the set of clock constraints consisting of propositions obtained by freely applying boolean connectives to the atomic propositions $x < \tau$ and $x = \tau$, with $\tau \in Time$.

A *TA* is a tuple $A = \langle \Sigma, S, S_0, C, E, F \rangle$, where

1. Σ is a finite alphabet;
2. S is a finite set of states and S_0 a set of initial states;
3. C is a finite set of clocks;
4. $E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C)$ gives the set of transitions. An arc $\langle s, s', a, \lambda, \delta \rangle$ represents a transition from state s to s' on an input symbol a. The set $\lambda \subseteq C$ gives the clocks to be reset with this transition and δ is a clock constraint over C;
5. $F \subseteq S$ is the set of accepting states.

An example of *TA* with a clock x is given in Fig.3 where $x := 0$ means the reset of clock x.

Let us recall now how a timed sequence is recognised by a timed automaton (we take the following definitions from [3]). A *path* of an automaton $A = \langle \Sigma, S, S_0, C, E, F \rangle$ is an infinite sequence of consecutive transitions

$$P = s_0 \overset{a_1,\lambda_1,\delta_1}{\longrightarrow} s_1 \overset{a_2,\lambda_2,\delta_2}{\longrightarrow} s_2 \ldots$$

such that $s_0 \in S_0$ and $\langle s_i, s_{i+1}, a_{i+1}, \lambda_{i+1}, \delta_{i+1} \rangle \in E$, for $i \geq 0$.
A *run* R of A over the path P is a timed sequence of consecutive transitions

$$R = s_0 \overset{a_1,\lambda_1,\delta_1}{\longrightarrow}_{\tau_1} s_1 \overset{a_2,\lambda_2,\delta_2}{\longrightarrow}_{\tau_2} s_2 \ldots$$

where, informally, $\tau_i \in Time$ is the time at which the transition $s_{i-1} \overset{a_i,\lambda_i,\delta_i}{\longrightarrow} s_i$ is taken provided that the clock constraint δ_i is satisfied by the current clock valuation and the clocks in λ_i are reset at time τ_i.
More precisely, the value of clock x at time τ belongs to $Time$ and is written $x(\tau)$, and the clock valuation at time τ can be written as the tuple $C(\tau) = \langle x(\tau) \rangle_{x \in C}$. Clocks are set to 0 at time $\tau_0 = 0$ and the clock valuation for $\tau > 0$ is determined by the considered run according to the following equation:

$$x(\tau_i) = \begin{cases} 0 & \text{if } x \in \lambda_i, \\ x(\tau_{i-1}) + \tau_i - \tau_{i-1} & \text{otherwise,} \end{cases}$$

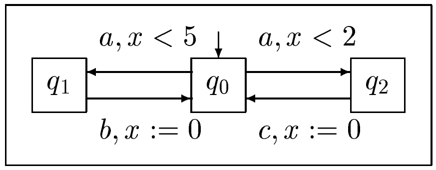

Figure 3. A Timed automaton.

and $x(\tau) = x(\tau_{i-1}) + \tau - \tau_{i-1}$ for $\tau_{i-1} \leq \tau < \tau_i$.
The clock constraint δ_i is satisfied at time τ_i if $C(\tau_{i-1}) + \tau_i - \tau_{i-1} \models \delta_i$ in the standard way.
A run R over a path P is *successful* if $Inf(P) \cap F \neq \emptyset$, where $Inf(P)$ is the set of states of A which occur infinitely often in P. A run $s_0 \xrightarrow[\tau_1]{a_1,\lambda_1,\delta_1} s_1 \xrightarrow[\tau_2]{a_2,\lambda_2,\delta_2} s_2 \ldots$ induces a timed sequence $\langle a_1, \tau_1 \rangle, \langle a_2, \tau_2 \rangle, \ldots$ and the language accepted by an automaton A, denoted by $L(A)$, is the set of timed sequences induced by a successful run of A.

Example 3.1. The *TA* of Fig. 3 with $F = S$ accepts the language
$\{\langle \alpha_1, \alpha_2 \rangle : \alpha_1 \in \{ab, ac\}^\omega$, if $\alpha_1(2i+1) = b$, then $\alpha_2(2i) - \alpha_2(2i-1) < 5$, if $\alpha_1(2i+1) = c$, then $\alpha_2(2i) - \alpha_2(2i-1) < 2$, for $i \geq 0\}$.

Timed Automata with periodic clock constraints (TA_ps)

The definition of automata with periodic clock constraints (see [5]) differs from the definition of *TA*s only in the type of conditions on clocks $\Phi(C)$.
Let $[\tau_1, \tau_2]$ denote the closed interval of $Time$ bounded by τ_1 and τ_2, with $\tau_1, \tau_2 \in Time$ and $\tau_1 \leq \tau_2$. For $\sigma \in Time$, let $T_\sigma[\tau_1, \tau_2]$ denote the set

$$\{k\sigma + \tau : k \in \mathbb{N}, \tau_1 \leq \tau \leq \tau_2\}.$$

The set $\Phi_p(C)$ of periodic clock constraints consists of propositions obtained by freely applying boolean connectives to the atomic propositions $x \in T_\sigma[\tau_1, \tau_2]$, x belonging to the set of clocks C. Intuitively, a constraint $x \in T_\sigma[\tau_1, \tau_2]$ is satisfied if the current value of the clock x belongs to the set $T_\sigma[\tau_1, \tau_2]$. A TA_p is a tuple $A = \langle \Sigma, S, S_0, C, E, F \rangle$, defined exactly as in the case of *TA* with $\Phi_p(C)$ taking the role of $\Phi(C)$. Also the concept of path, run and language accepted by a TA_p can be inherited from the definition of *TA*.

Example 3.2. The TA_p of Fig. 4 with $F = S$ accepts the language

$$\{\langle \alpha_1, \alpha_2 \rangle : \alpha_1 \in \{a\}^\omega, \alpha_2(i+1) - \alpha_2(i) \text{ is even, for } i \geq 0\}.$$

Timed Automata with silent transitions (TA_ϵs)

The definition of automata with silent transitions (see [3]) differs from the definition of *TA*s only since it permits transitions labelled over the alphabet Σ extended with the special symbol

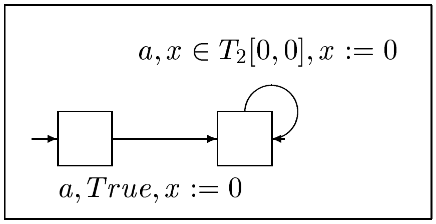

Figure 4. A TA_p .

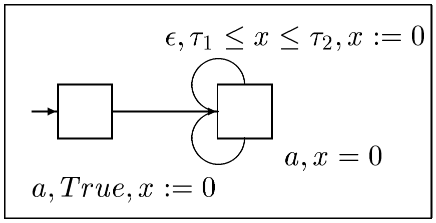

Figure 5. A TA_ϵ.

ϵ. A Timed Automaton with silent transitions (TA_ϵ) is a tuple $A = \langle \Sigma, S, S_0, C, E, F \rangle$, where the set of transitions is

$$E \subseteq S \times S \times \Sigma \cup \{\epsilon\} \times 2^C \times \Phi(C)$$

and the other items are defined as in the case of a TA.
The concept of path and (successful) run of a TA_ϵ is the same as in the case of TAs. A run $s_0 \xrightarrow{a_1,\lambda_1,\delta_1}_{\tau_1} s_1 \xrightarrow{a_2,\lambda_2,\delta_2}_{\tau_2} s_2 \ldots$ induces the timed sequence obtained from $\langle a_1, \tau_1 \rangle, \langle a_2, \tau_2 \rangle, \ldots$ by removing any occurrence of a silent transition, i.e. pairs belonging to $\{\epsilon\} \times Time$. The language accepted by the automaton A, denoted by $L(A)$, is the set of timed sequenced induced by a successful run of A.

Example 3.3. The TA_ϵ of Fig. 5 with $F = S$ accepts the language

$$\{\langle a^\omega, \alpha_2 \rangle : \text{ for each } i \text{ exists } k \text{ s.t. } \alpha_2(i+1) - \alpha_2(i) \in [\tau_1 k, \tau_2 k]\}.$$

3.2. Results

Given a class X of automata we denote with $\mathcal{L}(X)$ the set of languages accepted by the automata of the class X. It is known from the literature (see [3] and [5]) that the three classes $\mathcal{L}(TA)$, $\mathcal{L}(TA_p)$, $\mathcal{L}(TA_\epsilon)$ have different expressive power and that, in particular,

$$\mathcal{L}(TA) \subset \mathcal{L}(TA_p) \subset \mathcal{L}(TA_\epsilon).$$

We are interested in comparing $\mathcal{L}(TCA)$ with the classes mentioned, by investigating how expressive power is affected by the features of internal activity, urgency and reactivity. Since

$$
\begin{array}{ccccccccc}
 & & \mathcal{L}(STCA_{IRU}) & & \mathcal{L}(TCA_{RU}) & & & & \\
 & & \cap| & & \cup| & & & & \\
 & & \mathcal{L}(TA) & \subset & \mathcal{L}(TA_p) & & \subset & & \mathcal{L}(TA_\epsilon) \\
 & & \cap| & & \cap| & & & & \cap| \\
 & & \mathcal{L}(TCA_{IRU}) & \subset & \mathcal{L}(TCA_U) & & & & \mathcal{L}(TCA_R) \\
 & & || & & \cup| & & & & \cup| \\
\mathcal{L}(TCA^G_{IR}) & = & \mathcal{L}(TCA^G_{IU}) & & \mathcal{L}(STCA_U) & \subseteq & \mathcal{L}(STCA_{RU}) & \subset & \mathcal{L}(STCA_R) \\
 & & || & & & & & & \cup| \\
 & & \mathcal{L}(TCA^G_I) & & & & & & \mathcal{L}(STCA)
\end{array}
$$

Figure 6. The hierarchy of classes of languages accepted by TCA's.

TCAs accept timed sequences augmented with duration of transmission, in this section, to permit comparisons, we consider only environments of the form $\alpha = \langle \alpha_1, \alpha_2, \alpha_3 \rangle$ where $\alpha_1(i)$ is a singleton and $\alpha_3(i)$ is 0 for each $i \geq 0$.
In the following we shall show that the strongest limitation to expressiveness is due to the lack of internal activity. We prove that $\mathcal{L}(TCA_{IRU})$ is at least as powerful as $\mathcal{L}(TA)$ and that $\mathcal{L}(TA)$ is at least powerful as the sequential subclass of $\mathcal{L}(TCA_{IRU})$. Actually, it is not yet clear to us whether the sequential subclass of Timed Cooperating Automata (and its variants) is as expressive as the parallel class. Moreover, in absence of internal activity one does not gain from urgency and reactivity if transitions are controlled by the environment, as it follows from the fact that, if the transitions are guarded by the environment (denoted by an upper index G), we have $\mathcal{L}(TCA^G_I) = \mathcal{L}(TCA^G_{IU}) = \mathcal{L}(TCA^G_{IR}) = \mathcal{L}(TCA^G_{IRU}) = \mathcal{L}(TCA_{IRU})$.

If internal activity is permitted, Timed Cooperating Automata are at least as expressive as $\mathcal{L}(TA_p)$. In this case the request of urgency is a bound to expressiveness. In fact when we release this request, we attain at least the expressiveness of $\mathcal{L}(TA_\epsilon)$. This shows a strong connection between silent moves and lack of urgency.

The role of reactivity has been studied, at the moment, only in the sequential case and it remains to be studied in the parallel case. In fact, we prove that the reactive classes can simulate the non-reactive ones (i.e. $\mathcal{L}(STCA_U) \subseteq \mathcal{L}(STCA_{RU})$ and $\mathcal{L}(STCA) \subseteq \mathcal{L}(STCA_R)$) but we are not yet able to say whether these inclusions are proper. (We believe that the results given for the sequential subclasses can be extended to the general case.)

The hierarchy of the mentioned classes of timed languages is summarised in Figure 6.

Theorem 3.1. *The class of languages $\mathcal{L}(TA)$ is included in $\mathcal{L}(TCA_{IRU})$.*

Proof:
We give the translation of a TA $A = \langle \Sigma, S, S_0, C, E, F \rangle$ into a TCA_{IRU} M accepting the same language. For the sake of simplicity, we assume that the automaton A is such that, for each pair of states s_1 and s_2 in S, any transition from s_1 to s_2 resets the same set of clocks, namely for each pair of transitions $\langle s_1, s_2, a, \lambda, \delta \rangle$ and $\langle s_1, s_2, a', \lambda', \delta' \rangle$ in E, $\lambda = \lambda'$. This assumption is not

restrictive since an automaton not fulfilling this requirement can be transformed, by suitably duplicating states and transitions, into an automaton accepting the same language and fullfilling the requirement.
If $C = \{x_1, \ldots, x_n\}$ is the set of clocks, the automaton $M = \langle M_1, \ldots, M_{n+1}, \mathcal{I}, \mathcal{F}\rangle$ is composed by $n+1$ sequential components, i.e. a sequential component for each clock (components $M_1, \ldots, M_n$) and one simulating the state transition diagram of the automaton A (component M_{n+1}). The sequential component for the clock x_i consists of one state only and looping arcs which are taken when a transition resetting the clock is performed. More formally, $M_i = \langle\{c_i\}, c_i, \delta_i\rangle$ with $\delta_i = \{\langle c_i, True, s_1\{0\} \wedge s_2 = 0, c_i\rangle : \langle s_1, s_2, a, \lambda, \delta\rangle$ and $x_i \in \lambda\}$.
As regards the component simulating the state-transition diagram, we assume for simplicity that the set of initial states is a singleton (i.e. $S_0 = \{s_0\}$). The state-transition diagram of M_{n+1} is isomorphic to that of A. Each condition c is transformed into a condition $T(c)$ by replacing each atomic proposition $x_i = d$ (resp. $x_i < d$) with $c_i = d$ (resp. $\neg c_i[d]$). Hence, $M_{n+1} = \langle S, s_0, \delta_{n+1}\rangle$, where $\delta_{n+1} = \{\langle s_i, a, T(c), s_j\rangle : \langle s_i, s_j, a, \lambda, c\rangle \in E\}$. We also assume $\mathcal{F} = F$ and $\mathcal{I} = S \cup \{c_i : x_i \in C\}$. An example which illustrates the procedure is given in Fig.7.
It is easy to see that $L(A) = L(M)$. □

Theorem 3.2. *The class of languages $\mathcal{L}(STCA_{IRU})$ is included in $\mathcal{L}(TA)$.*

Proof:
We show that any $STCA_{IRU}$ $M = \langle\langle Q, q_0, \delta\rangle, \mathcal{I}, \mathcal{F}\rangle$ can be simulated by a *TA* A. In order to express conditions in A which are the equivalent of conditions of the form $q = \tau$ and $q[\tau]$ in M, A is endowed with a clock x_{in} which is reset each time a transition is performed. Conditions $q = \tau$ and $q[\tau]$ can be evaluated to true only if q is the currently enabled state (the automaton is sequential) and can be replaced in A by $x_{in} = \tau$ and $x_{in} > \tau$. To express a condition equivalent to $q\{\tau\}$ over a state $q \in Q$, the automaton A is endowed with a clock x_q for any $q \in Q$ (actually a clock would be required only for states referenced by this kind of condition). The clock x_q is reset by any transition having the state q as source and keeps track of the time elapsed from the last disabling of q. Since the condition $q\{\tau\}$ is equivalent to the fact that q has been enabled at least once and $x_q \leq \tau$, we have to keep track of the set of states visited at least once. To this purpose we use a set of states having the form $Q \times 2^Q$, where the intended meaning is that the second component of the state stores the set of already visited states. For a (current) state q and a set of visited states W, the function T translating a condition $\gamma \in \Gamma_Q$ of M into a condition of A is defined inductively as follows: $T(q, W, True) = True$; $T(q, W, q'[\tau]) = False$ if $q' \neq q$ and $x_{in} \geq \tau$ otherwise; $T(q, W, q' = \tau) = False$ if $q' \neq q$ and $x_{in} = \tau$ otherwise; $T(q, W, q'\{\tau\}) = True$ if $q' = q$ and otherwise $T(q, W, q'\{\tau\}) = x_{q'} \leq \tau$ if $q' \in W$ and $False$ otherwise; $T(q, W, \gamma_1 \wedge \gamma_2) = T(q, W, \gamma_1) \wedge T(q, W, \gamma_2)$ and $T(q, W, \neg\gamma) = \neg T(q, W, \gamma)$. Hence the *TA* A simulating M is $\langle \Sigma, Q \times 2^Q, \{\langle q_o, \emptyset\rangle\}, C, E, \mathcal{F}\rangle$, where

1. $C = \{x_{in}\} \cup \{x_q : q \in Q\}$;
2. $E = \{\langle\langle q_1, W\rangle, \langle q_2, W \cup \{q_1\}\rangle, a, \{x_{in}, x_{q_1}\}, T(q_1, W, \delta)\rangle : \langle q_1, a, \gamma, q_2\rangle \in \delta\}, W \in 2^Q\} \cup$
   $\{\langle\langle q, W\rangle, \langle q, W\rangle, a, \emptyset, True\rangle : a \notin \Sigma_q, W \in 2^Q\} \cup$

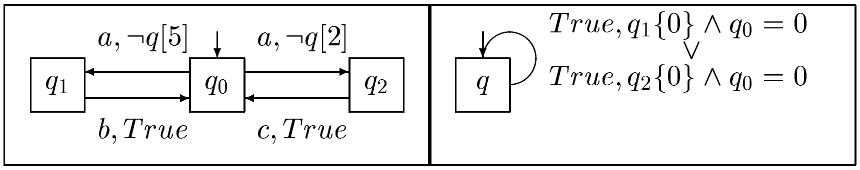

Figure 7. The translation of the automaton of Fig.3 into a TCA_{IRU}.

$\{\langle\langle q, W\rangle, \langle q, W\rangle, a, \emptyset, \neg\gamma_{q,a}\rangle : a \in \Sigma_q, W \in 2^Q\}$
where $\Sigma_q = \{a : \langle q, a, \gamma, q'\rangle \in \delta\}$ and $\gamma_{q,a} = \bigvee\{\gamma : \langle q, a, \gamma, q'\rangle \in \delta\}$.

In the definition of E, the two last subsets of self-loops simulate the idling of M in the state q when no transition exiting q can be taken.
It is easy to see that $L(A) = L(M)$. $\square$

Let us denote by $\mathcal{L}(TCA^G)$ the subclass of $\mathcal{L}(TCA)$ where transitions are guarded by the environment, i.e., more formally, where transitions have environment conditions in the set $\Pi_\Sigma - \{True\}$.

Proposition 3.1. *The classes $\mathcal{L}(TCA_I^G)$, $\mathcal{L}(TCA_{IR}^G)$, $\mathcal{L}(TCA_{IU}^G)$, $\mathcal{L}(TCA_{IRU}^G)$ and $\mathcal{L}(TCA_{IRU})$ coincide.*

We show now that permitting internal activity increases the expressive power.

Theorem 3.3. *The class of languages $\mathcal{L}(TA_p)$ is included in $\mathcal{L}(TCA_U)$ and $\mathcal{L}(TCA_{RU})$.*

Proof:
We sketch the translation of a TA_p $A = \langle\Sigma, S, S_0, C, E, F\rangle$ into a TCA_U M accepting the same language. The translation is similar to the one given in Th.3.1 and it differs only in the simulation of clocks. As in Th.3.1, we assume that the automaton A is such that for each pair of transitions $\langle s_1, s_2, a, \lambda, \delta\rangle$ and $\langle s_1, s_2, a', \lambda', \delta'\rangle$ in E, $\lambda = \lambda'$.
Let $CC = \{x_1 \in T_{\sigma_1}[\tau_1^1, \tau_1^2], \ldots, x_n \in T_{\sigma_n}[\tau_n^1, \tau_n^2]\}$, with $x_i \in C$, be the set of clock constraints occurring in the labels of transitions of A. The TCA_U M has at most $2n+1$ sequential components, i.e. at most a couple of components for each clock constraint in the set CC, and a component simulating the state transition diagram of the automaton A. Let us see how a clock constraint $x_i \in T_{\sigma_i}[\tau_i^1, \tau_i^2] \in CC$ is simulated. If $\tau_i^2 - \tau_i^1 \geq \sigma_i$, then each condition $x_i \in T_{\sigma_i}[\tau_i^1, \tau_i^2]$ is equivalent to $x_i \geq \tau_i^1$, and the clock for evaluating such a constraint is the sequential component given in the proof of Th.3.1. Hence, let us consider the case in which $\tau_i^2 - \tau_i^1 < \sigma_i$ and let $d = 0$ if $\tau_i^2 \leq \sigma_i$, $d = \tau_i^2 - \sigma_i$ otherwise. Note that the set $T_{\sigma_i}[\tau_i^1, \tau_i^2]$ is equal to $d + T_{\sigma_i}[\tau_i^1 - d, \tau_i^2 - d]$. The two components required for the simulation of a clock x_i suitable for checking the constraint $x_i \in T_{\sigma_i}[\tau_i^1, \tau_i^2]$ are represented in Figure 8. The first component simulates a clock which, after waiting for an amount of time d, goes on with a periodic tick σ_i (loop beetween states c_i^1 and c_i^5). The loop beetween states c_i^1 and c_i^5 is non-deterministically suspended either to permit the

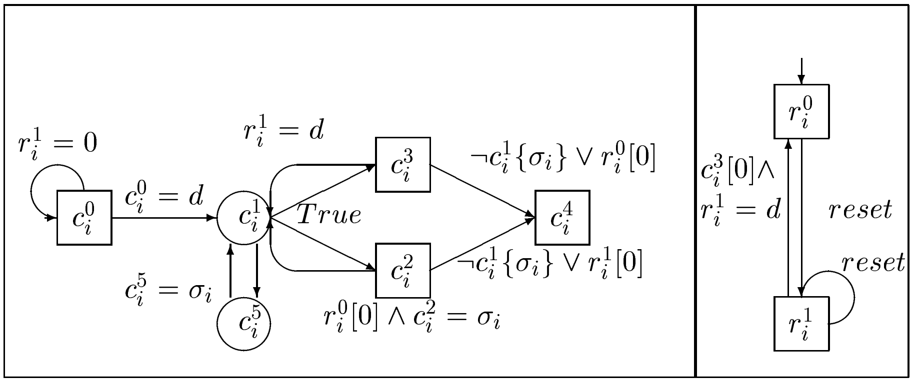

Figure 8. The clock for evaluating the condition $x_i \in T_{\sigma_i}[\tau_i^1, \tau_i^2]$.

evaluation of the clock $x_i \in T_{\sigma_i}[\tau_i^1 - d, \tau_i^2 - d]$ constraint (state c_i^2) or to respond to a reset (state c_i^3). If the non-deterministic guess is wrong (for instance the loop is exited too early), the state of error c_i^4 is reached and the whole computation is aborted. The second component keeps track of reset requests issued for clock x_i (no request in state r_i^0, one or more requests in state r_i^1). (In the Figure, $reset$ stands for the condition $\bigvee_{k,j} s_k\{0\} \wedge s_j[0]$ such that $\langle s_k, s_j, a, \lambda, \delta\rangle \in E, x_i \in \lambda$.) The condition $x_i \in T_{\sigma_i}[\tau_i^1, \tau_i^2]$ is therefore equivalent to the condition

$$\begin{aligned}\phi(x_i, \sigma_i, \tau_i^1, \tau_i^2) = \quad & (c_i^2[\tau_i^1 - d] \wedge (c_i^2 = \tau_i^2 - d \vee \neg c_i^2[\tau_i^2 - d])) \vee \\ & (r_i^0[0] \wedge c_i^3[\tau_i^1 - d] \wedge (c_i^3 = \tau_i^2 - d \vee \neg c_i^3[\tau_i^2 - d])).\end{aligned}$$

The state transition diagram of A can be simulated by a sequential component defined as in Th.3.1 with the only difference that each condition δ occurring in a transition of A which reset the set of clocks λ is substituted by a condition $\bigwedge_{i=1}^{n} \neg c_i^4[0] \wedge \bigwedge_{x_i \in \lambda} c_i^3[0] \wedge \overline{\delta}$, with $\overline{\delta}$ the condition obtained from δ by replacing each constraint having the form $x_i \in T_{\sigma_i}[\tau_i^1, \tau_i^2]$ by $\phi(x_i, \sigma_i, \tau_i^1, \tau_i^2)$. Moreover, in order to stop the computation whenever a state c_i^4 is reached in a clock component, to the component simulating the state transition diagram of A, a state q is added, and for each state q' a transition is added labelled by the constraint $\bigvee_{i=1}^{n} c_i^4[0]$ taking from q' to q. Now, it can be easily proved that $L(A) = L(M)$.
The proof that a TA_p can be simulated by a TCA_{RU} is quite similar and then is omitted. □

Since the simulation of a TCA_{IRU} by a TCA_U is straightforward, by Th.3.3 and by the strict inclusion of the class of languages $\mathcal{L}(TA)$ in the class $\mathcal{L}(TA_p)$ (see [5]), we have the following corollary.

Corollary 3.1. *The class of languages $\mathcal{L}(TCA_{IRU})$ is strictly included in $\mathcal{L}(TCA_U)$.*

We show now that TCA_Rs are at least as powerful as TA_ϵs.

Theorem 3.4. *The class of languages $\mathcal{L}(TA_\epsilon)$ is included in $\mathcal{L}(TCA_R)$.*

Proof:
Let $A = \langle \Sigma, S, S_0, C, E, F\rangle$ be a TA_ϵ. Without loss of generality, we assume that A is such that for each pair of states s_1 and s_2 in S any transition from s_1 to s_2 resets the same set of clocks, namely for each pair of transitions $\langle s_1, s_2, a, \lambda, \delta\rangle$ and $\langle s_1, s_2, a', \lambda', \delta'\rangle$ in E, $\lambda = \lambda'$. Moreover, since we are concerned only with Non-Zeno languages, it is not restrictive to assume that all of the states in F are sources of at least one guarded (non-ϵ) transition which is secured by assuming the property of progress. We first transform A into an equivalent TA_ϵ A' which has only one initial state and whose set of states can be partitioned into a set of states which are sources of ϵ-transitions and another set of states which are sources of guarded transitions. The automaton A' is the tuple $\langle \Sigma, S \times \{\epsilon, \Sigma\}, \{s_0\}, C, E', F \times \{\Sigma\}\rangle$, where E' is defined as follows:
$\{\langle\langle s_1, \Sigma\rangle, \langle s_2, \Sigma\rangle, a, \lambda, \delta\rangle, \langle\langle s_1, \Sigma\rangle, \langle s_2, \epsilon\rangle, a, \lambda, \delta\rangle : \langle s_1, s_2, a, \lambda, \delta\rangle \in E, a \in \Sigma\} \cup$
$\{\langle\langle s_1, \epsilon\rangle, \langle s_2, \epsilon\rangle, \epsilon, \lambda, \delta\rangle, \langle\langle s_1, \epsilon\rangle, \langle s_2, \Sigma\rangle, \epsilon, \lambda, \delta\rangle : \langle s_1, s_2, \epsilon, \lambda, \delta\rangle \in E\} \cup$
$\{\langle s_0, s, \epsilon, \emptyset, x = 0\rangle : s \in S_0 \times \{\epsilon, \Sigma\},$ for some $x \in C\}$.
Now, the automaton A' can be translated into an equivalent TCA_R M, by following exactly the same procedure given in the proof of Th. 3.1 but fixing the set of input states to be equal to $(S \times \{\Sigma\}) \cup \{s_0\}$ and replacing ϵ by $True$. $\square$

In the remaining part of this section we compare the variants of our model with respect to the features of urgency and reactivity, limiting ourselves to the sequential case.
In the following theorem we show that by releasing the requirement of urgency we obtain an increase of expressiveness.

Theorem 3.5. *The class of languages $\mathcal{L}(STCA_{RU})$ is strictly included in $\mathcal{L}(STCA_R)$.*

Proof:
Let $M = \langle\langle Q, q_0, \delta\rangle, \mathcal{I}, \mathcal{F}\rangle$ be a $STCA_{RU}$. We assume, without loss of generality that transitions from states in $I - \{q_0\}$ are guarded (since the environmente gives each time a symbol, the condition true can be replaced by $\bigvee_{a\in\Sigma} a$). For simplicity, we assume also that there is no transition $\langle q_1, \pi, \gamma, q_0\rangle \in \delta$ and that any internal condition $\gamma \in \Gamma_Q$ is the conjunction of positive/negative occurrences of condition of the form $q = \tau$, $q[\tau]$ and $q\{\tau\}$. The set of formulas having this form is denoted by Γ_Q^N. We introduce a function $TT : Q \times \Gamma_Q^N \to 2^{Time}$ which, for a state and an internal condition labelling a transition diparting from the state, gives the set of times in which the condition satisfied.
The urgency of the transitions depending from states in $I - \{q_0\}$ is guaranteed by the fact that these transitions are guarded. The urgency of the transitions depending from non-input states of the given automaton M can be simulated in a TCA_R by requiring that the source state q of the transition is left at time $min(TT(q, \gamma))$, if such minimum exists. Transitions of M such that $min(TT(q, \gamma))$ does not exist, will not be considered as they are never performed in a urgent automaton.
The function TT is defined as follows:
$TT(q, q'[\tau]) = [\tau, \infty)$ if $q = q'$ and $\emptyset$, otherwise;
$TT(q, \neg q'[\tau]) = [0, \tau)$ if $q = q'$ and $Time$, otherwise;

$TT(q, q' = \tau) = [\tau, \tau]$ if $q = q'$ and $\emptyset$, otherwise;
$TT(q, \neg q' = \tau) = Time - \{\tau\}$ if $q = q'$ and $Time$, otherwise;
$TT(q, q'\{\tau\}) = TT(q, True) = Time$;
$TT(q, \neg q'\{\tau\}) = \emptyset$ if $q = q'$ and $Time$ otherwise;
$TT(q, False) = \emptyset$; $TT(q, \gamma_1 \wedge \gamma_2) = TT(q, \gamma_1) \cap TT(q, \gamma_2)$.
The TCA_R M' simulating the automaton M is $\langle\langle Q, q_0, \delta'\rangle, \mathcal{I}, \mathcal{F}\rangle$ where $\delta' = \{\langle q_1, \pi, \gamma \wedge q_1 = min(TT(q_1, \gamma)), q_2\rangle : \langle q_1, \pi, \gamma, q_2\rangle \in \delta, q_1 \in Q - \mathcal{I}$ and $min(TT(q_1, \gamma))$ exists$\} \cup \{\langle q_1, \pi, \gamma, q_2\rangle : \langle q_1, \pi, \gamma, q_2\rangle \in \delta, q_1 \in \mathcal{I}, \pi \neq true\} \cup \{\langle q_0, true, \gamma \wedge q_0 = min(TT(q_0, \gamma)), q_1\rangle : \langle q_0, true, \gamma, q_1\rangle \in \delta$ and $min(TT(q_0, \gamma))$ exists$\}$.
It is easy to see that $L(M) = L(M')$.

As for the strictness of the inclusion, let us consider the language of Example 3.3 which is accepted by a TA_ϵ. It can be easily proved that such a language, which can be accepted by a $STCA_R$, cannot be accepted by any TCA_{RU}. □

We show now that urgent reactive Timed Cooperating Automata can simulate the urgent non reactive ones.

Theorem 3.6. *The class of languages $\mathcal{L}(STCA_U)$ is included in $\mathcal{L}(STCA_{RU})$.*

Proof:
Let $M = \langle\langle Q, q_0, \delta\rangle, \mathcal{I}, \mathcal{F}\rangle$ be a $STCA_U$. Without loss of generality we assume that if $\langle q_1, \pi, \gamma, q_2\rangle \in \delta$ and $q_1 \in Q - \mathcal{I}$, then $\pi = True$. We assume also, without loss of generality, that any formula labelling transitions (as in Th. 3.5) belongs to Γ_Q^N.
A TCA_U with internal activity may perform non reactive steps, namely steps that are completed after the next prompt from the environment. In order to simulate such a non reactive step on a TCA_{RU}, it should be possible to suspend the step at any stage of its progress and to resume it after the synchronization with the environment. This is obtained by duplicating non input states anticipating sequences of transitions for any possible suspension. The TCA_{RU} simulating M is $M' = \langle\langle Q', q_0, \delta'\rangle, \mathcal{I}', \mathcal{F}\rangle$, where:
$Q' = \mathcal{I} \cup (Q - \mathcal{I}) \times \{i, n\}$;
$\mathcal{I}' = \mathcal{I} \cup (Q - \mathcal{I}) \times \{i\}$);
$\delta' = \{\langle q_1, \pi, f(\gamma), q_2\rangle : \langle q_1, \pi, \gamma, q_2\rangle \in \delta, q_1, q_2 \in \mathcal{I}\} \cup$
$\{\langle q_1, \pi, f(\gamma), \langle q_2, x\rangle\rangle : \langle q_1, \pi, \gamma, q_2\rangle \in \delta, q_1 \in \mathcal{I}, q_2 \in Q - \mathcal{I}, x \in \{i, n\}\} \cup$
$\{\langle\langle q_1, x\rangle, \pi, f(\gamma) \wedge \langle q_1, x\rangle = min(TT(q_1, \gamma)), \langle q_2, y\rangle\rangle : \langle q_1, \pi, \gamma, q_2\rangle \in \delta, q_1, q_2 \in Q - \mathcal{I}, x, y \in \{i, n\}$, if $min(TT(q_1, \gamma))$ exists$\} \cup$
$\{\langle\langle q_1, x\rangle, \pi, f(\gamma) \wedge \langle q_1, x\rangle = min(TT(q_1, \gamma)), q_2\rangle : \langle q_1, \pi, \gamma, q_2\rangle \in \delta, q_1 \in Q - \mathcal{I}, q_2 \in \mathcal{I}, x \in \{i, n\}$, if $min(TT(q_1, \gamma))$ exists$\}$, with
$TT : Q \times \Gamma_Q^N \to 2^{Time}$ defined as in the proof of Th.3.5, and $f : \Gamma_Q \to \Gamma_{Q'}$ defined as follows
$f(True) = True$;
$f(q[\tau]) = q[\tau]$ if $q \in \mathcal{I}$ and $\langle q, i\rangle[\tau] \vee \langle q, n\rangle[\tau]$, otherwise;
$f(q\{\tau\}) = q\{\tau\}$ if $q \in \mathcal{I}$ and $\langle q, i\rangle\{\tau\} \vee \langle q, n\rangle\{\tau\}$, otherwise;
$f(q = \tau) = q = \tau$ if $q \in \mathcal{I}$ and $\langle q, i\rangle = \tau \vee \langle q, n\rangle = \tau$, otherwise;

f commutes with respect to $\neg$, $\wedge$ and $\vee$.
It is easy to see that $L(M) = L(M')$. $\square$

Proposition 3.2. *The class of languages $\mathcal{L}(STCA)$ is included in $\mathcal{L}(STCA_R)$.*

Proof:
The technique for simulating non reactive steps of a $STCA$ by a TCA_R is the one used for the proof of Th. 3.6. $\square$

As a conclusion, we observe that, while the simulation of non reactivity in a context of reactivity is possible, it is not clear to us whether and how in a non reactive context reactivity can be simulated.

4. Closures under boolean operations and decidability properties

All the classes of Timed Cooperating Automata we have considered are closed under union and intersection.

Let $M = \langle M_1, \dots M_m, \mathcal{I}, \mathcal{F} \rangle$ and $M' = \langle M'_1, \dots, M'_n, \mathcal{I}', \mathcal{F}' \rangle$ be two Timed Cooperating Automata which share the alphabet Σ and such that their sets of states are disjoint.

An automaton accepting the language $L(M) \cap L(M')$ is

$$M \cap M' = \langle M_1, \dots M_m, M'_1, \dots, M'_n, M_{check}, \overline{\mathcal{I}}, \overline{\mathcal{F}} \rangle$$

where M_{check} is the sequential component in Fig.9;
$\overline{\mathcal{I}} = \mathcal{I} \cup \mathcal{I}' \cup \{q_0, q_1, q_2, q_3\}$; $\overline{\mathcal{F}} = \{q_1, q_3\}$.
An automaton accepting the language $L(M) \cup L(M')$ is

$$M \cup M' = \langle M_1, \dots M_m, M'_1, \dots, M'_n, \mathcal{I} \cup \mathcal{I}', \mathcal{F} \cup \mathcal{F}' \rangle.$$

Notice that both the operations of union and intersection of languages L_1 and L_2 are implemented by composing in parallel the automata accepting L_1 and L_2 in a straightforward way. This permits to preserve the structure of the component automata and to have, in general, a more compact representation than that obtained by the cartesian product of automata which is necessary in the case of TAs.

Proposition 4.1. *Let M and M' be two Timed Cooperating Automata, then $L(M \cup M') = L(M) \cup L(M')$ and $L(M \cap M') = L(M) \cap L(M')$.*

We have no results on complementation.

We have a result on decidability of emptiness, which follows from the analogous result for $\mathcal{L}(TA)$ and from the containment of $\mathcal{L}(STCA_{IRU})$ in $\mathcal{L}(TA)$.

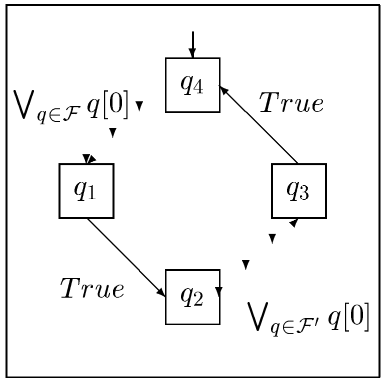

Figure 9. The component M_{check}.

Proposition 4.2. *The emptiness problem for $\mathcal{L}(STCA_{IRU})$ is decidable.*

We note that if the class $\mathcal{L}(TCA_{IRU})$ could be proved to coincide with $\mathcal{L}(TA)$, the classes $\mathcal{L}(TCA_{RU})$ and $\mathcal{L}(TCA_U)$ to coincide with $\mathcal{L}(TA_p)$ and the class $\mathcal{L}(TCA_R)$ to coincide with $\mathcal{L}(TA_\epsilon)$, then also the following results would hold:

1. the classes $\mathcal{L}(TCA_{IRU})$, $\mathcal{L}(TCA_{RU})$, $\mathcal{L}(TCA_U)$, and $\mathcal{L}(TCA_R)$ are not closed under complementation;
2. the emptiness problem for the classes $\mathcal{L}(TCA_{IRU})$, $\mathcal{L}(TCA_{RU})$, $\mathcal{L}(TCA_U)$, and $\mathcal{L}(TCA_R)$ is decidable (as a consequence of the decidability for the class $\mathcal{L}(TA_\epsilon)$, see [3]).

5. Succinctness

As suggested in [6], a criterion for comparing classes of automata (besides the obvious criterion of expressiveness) is *succinctness*, namely the inherent size of an automaton required to accept a given language. Given two classes of automata $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{B}$ is *more succinct* then $\mathcal{A}$ if

1. for each automaton $A \in \mathcal{A}$ accepting the language $L(A)$ there exists an automaton $B \in \mathcal{B}$ accepting $L(A)$ and such that size of B is polynomial in the size of A;
2. there is a family of languages L_n, for $n > 0$, such that each L_n is accepted by the automaton $B \in \mathcal{B}$ of a size polynomial in n, but the smallest $A \in \mathcal{A}$ accepting L_n is at least of size exponential in n.

The notion of size of a *TCA* and of a *TA* is standard and it is defined as follows.
The size of a *TCA* $M = \langle M_1, \ldots, M_n, \mathcal{I}, \mathcal{F}\rangle$, with $M_i = \langle Q_i, q_i, \delta_i\rangle$, is obtained by counting states, transitions and length of transitions labels, i.e. it is $\Sigma_i|Q_i| + \Sigma_i|\delta_i|$, where $|\delta_i|$ is the number of conditions $True$, $q[\tau]$, $q = \tau$, $q\{\tau\}$ occurring in δ_i.
The size of a *TA* $A = \langle \Sigma, S, S_0, C, E, F\rangle$ is $|S| + |E|$, where $|E|$ is the number of conditions $True$, $x < \tau$, $x = \tau$ occurring in E.

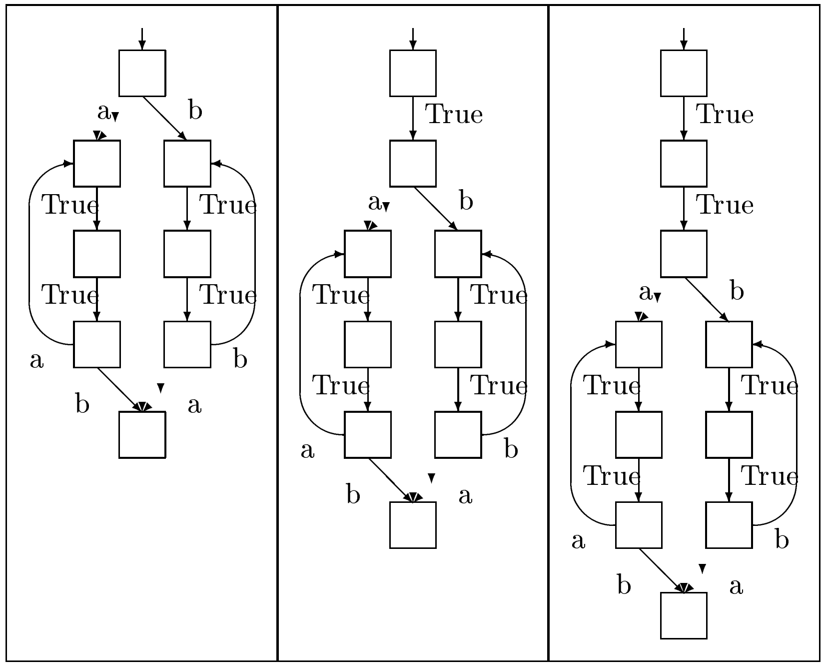

Figure 10. A TCA accepting L_3.

In [6], exponential discrepancies are proved in the size of finite automata, when augmented with mechanisms of concurrency, over sequential models. In particular, CAs are shown to be more succinct than Büchi automata. We can prove that an analogous result holds also in the timed case, namely that TCA_{IRU}s are more succinct than TAs.

Theorem 5.1. *TCA_{IRU}s are more succinct than TAs.*

Proof:
(Sketch) The first requirement of succinctness is satisfied since, at is it is easy to see, the TCA_{IRU} which simulates a given TA defined in the proof of Th. 3.1 has polynomial size in the size of the given TA.
As far as the second requirement is concerned, let us consider the family of languages $L_1, \ldots, L_n, \ldots$ over the alphabet $\Sigma = \{a, b\}$, defined as follows:

$$L_n = \{\langle \alpha_1, \alpha_2, \alpha_3 \rangle : \alpha_1(i) = \alpha_1(i+n) \text{ for } i \geq 0\}.$$

The idea of a TCA accepting L_n may be easily gathered from Fig.10, where it is presented the case for $n = 3$. Notice that number of components is n, and that the size of each component can be polynomially bounded by n. Viceversa, it is easy to see that a TA accepting the same language must use an exponential number of states (or clocks) to keep track of the 2^n possible sequences of words over $\{a, b\}$ of length n. $\square$

Our study of succinctness is at an early stage. We believe that it is interesting to investigate whether the features of internal activity, reactivity and urgency, besides affecting expressiveness, do affect also succincteness. Morever, a comparison of TA_ps, TA_ϵs and the various classes of Timed Cooperating Automata from the point of view of succinctness should be performed.

References

[1] Alur, R. and Dill, D.: "A Theory of Timed Automata", *Theoretical Computer Science*, **126**, 1994, 183–235.

[2] Alur, R., Fix, L. and Henzinger, A.: "Event-Clock Automata: A Determinizable Class of Timed Automata", *Theoretical Computer Science*, **211**, 1999, 253–273.

[3] Berard, B., Petit, A., Diekert, V. and Gastin, P.: "Characterization of the Expressive Power of Silent Transitions in Timed Automata", *Fundamenta Informaticae*, **36**, 1998, 145–182.

[4] Berry, G. and Gonthier, G.: "The Esterel Synchronous Programming Language: Design, Semantics, Implementation", *Science of Computer Programming*, **19**, 1992, 87–152.

[5] Choffrut, C. and Goldwurm, M.: "Timed Automata with Periodic Clock Constraints", Rapport L.I.A.F.A. n. 99/28, Université Paris VII, 1999.

[6] Drusinsky, D. and Harel, D.: "On the Power of Bounded Concurrency I: Finite Automata", *Journal of ACM*, **41**, 1994, 517–539.

[7] D'Souza, D. and Thiagarajan, P.S.: "Product Interval Automata: a Subclass of Timed Automata", *Proc. FSTTCS'99*, LNCS 1738, Springer, Berlin, 1999, 60–71.

[8] Harel, D.: "Statecharts: A Visual Formalism for Complex Systems", *Science of Computer Programming*, **8**, 1987, 231–274.

[9] Lanotte, R., Maggiolo-Schettini, A. and Peron, A.: "Timed Cooperating Automata", *Proc. of the CS&P'99 Workshop* (H.-D. Burkhard, L. Czaja, H.S. Nguyen, P. Starke Eds.), Warsaw University Press, Warsaw, 1999, 96–106.

[10] Maggiolo-Schettini, A. and Peron, A.: "Retiming Techniques for Statecharts", *Proc. FTRTFT '96*, LNCS 1135, Springer, Berlin, 1996, 55–71.

[11] Mikk, E., Laknech, Y. and Siegel, M.: "Hierarchical Automata as Models for Statecharts", *Proc. ASIAN '97*, LNCS 1345, Springer, Berlin, 1997, 181–196.

[12] Peron, A. and Maggiolo-Schettini, A.: "Transitions as Interrupts: A New Semantics for Timed Statecharts", *Proc. TACS '94*, LNCS 789, Springer, Berlin, 1994, 806–821.

[13] Thomas, W.: "Automata on Infinite Objects", *Handbook of Theoretical Computer Science* (J. van Leeuwen Ed.), Elsevier Science Publishers, Amsterdam, 1990, 134–191.