HAL open science

Robustness in Scenarios. ANR IMPRO Deliverable L 4.3

Loïc Hélouët

To cite this version:

Loïc Hélouët. Robustness in Scenarios. ANR IMPRO Deliverable L 4.3. [Technical Report] ANR IMPRO. 2014, pp.23. hal-01094363

HAL Id: hal-01094363

https://inria.hal.science/hal-01094363

Submitted on 12 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Robustness in Scenarios.
ANR IMPRO
Deliverable L 4.3

Loïc Hélouët

INRIA Rennes, France
loic.helouet@inria.fr

Abstract. This document considers robustness problems for timed scenarios. Robustness questions come from discrepancies between idealized representation of time in models and the actual measure of time and implementation of clocks in real architectures. A robustness question can usually we described as: given a set of requirements, how does architectural assumptions and imperfect time measurement affect the semantics of the specified behaviors? We propose different interpretations of time, that consider that measured time can be imprecise, that clocks are discrete, or that some time may elapse between the moment when the decision to perform an action is taken (for instance when a timer expires) and its actual execution. We then consider properties of timed scenarios such as consistency (does a specification describe at least one execution), path consistency (is it the case that for every timed run of a scenario description, there exists a consistent timed execution ?), and semantics preservation (is the behavior of the model the same with a timed semantics as without time constraints ?), and the robustness of such properties with respect to different timed semantics.

1 Introduction

Scenarios are a popular formalisms to design typical behaviors of distributed systems. Recently, they have been enhanced with time annotations, to allow specification of timing constraints within requirements [15]. Such timing constraints can be seen as prescriptions of behaviors that an implementation of a system executing a scenario specification should enforce. However, adding time to scenarios may completely change the behavior of the modeled system. Indeed, a scenario that is executable in an untimed setting might not be executable in a timed setting, due to inconsistent constraints on occurrence dates of events.

It is well known that a real system can not implement timing issues as precisely as required within a timed model. For instance, clocks measure time with some imprecision, and launching an action that is triggered by time (for instance expiration of a timer) is not immediate, and may not be enforceable without some delay. Such situation can be painful: formal models with idealized representation of time are frequently used for verification purpose, but a successful

verification of a timed system does not means that any implementation of this system will satisfy the verified property, nor that there exists an implementation with precise enough measurement of time and sufficient reactivity that preserves the verified property. This *robustness* problem was first formalized by Puri [17] within the context of timed automata. Later works showed that robustness of ω-regular properties [11] can also be verified.

This report addresses robustness issues for time constrained scenarios (TC-MSCs). We first provide several notions of "realistic time". More precisely, we consider imperfect time measurement and delayed triggering. The semantics of such timed systems is given by enlarging or shrinking semantics, or assuming that durations are measured up to some imprecision. We also consider discretized semantics, i.e semantics in which events can only occur at integer dates. This is particularly useful when considering that the modeled system in implemented on an architecture with discrete clocks. We then address several formal properties of TC-MSCs. The first property is consistency: for a TC-MSC description G, is it the case that G has at least one consistent run ? The second property is path consistency : is it the case that every run of a TC-MSC has a consistent execution under realistic time semantics ? We then consider robustness of (path) consistency properties: is it the case that a (path) consistent TC-MSC is still consistent under some "'realistic"' semantics. The second robustness issue considered is untimed semantics preservation: is it the case that the untimed language of a TC-MSC graph is the same under idealized time semantics and under realistic semantics ? The last issue we consider is whether there exists some clock precision guaranteeing robustness of the above properties.

Unsurprisingly, most of robustness questions are undecidable for TC-MSCs. This comes mainly from undecidability of consistency for TC-MSCs in the general case, but also from undecidability in MSC graphs (language equivalence or inclusion , regularity,... are undecidable). Fortunately, subclasses allow to address robustness questions for a large subclass of the model.

This report is organized as follows: Section 2 introduces the formalisms that will be used throughout the document, Section 3 introduces different "realistic" semantics, that consider imperfect of discretized time. Section 4 considers robustness issues (preservation of untimed language and consistency properties) under different interpretation of time when the nature and a bound on imprecision of implementation is known. Section 5 considers robustness under arbitrary precision, and section 6 concludes this work.

2 Preliminaries

This section describes the main models that will be used throughout the report, namely timed automata, and time constrained Message Sequence Charts. The reason to consider timed automata is twofold: first, a lot of litterature has been devoted to robustness for this model. second, we will show that most of decidable robustness issues for TC-MSCs can be brought back to a timed automaton setting.

2.1 Timed automata

Imperfection in time measurement has been extensively addressed for timed automata [8, 7]. The problem of timed robustness has been first addressed by [17] and then in many reference such as [12, 11, 13, 10, 9, 14, 18, 20, 21].

Let $C = \{x_1, ...x_n\}$ be a finite set of clocks. We denote by Φ_C the set of conjunctions of formulas of the form $k \approx x_i \approx' l$ where $k, l \in \mathbb{Q}^+$ and $\approx, \approx' \in \{<, \leq\}$. A valuation for a set of clocks is a function v that assigns a value from $\mathbb{R}$ to every clock. We also denote by $v + d$ the valuation that assigns value $v(c) + d$ to every clock c, and by $v_{|R}$ the valuation that assign 0 to every clock in R and $v(c)$ to every other clock c.

A timed automaton is a tuple $\mathcal{A} = (L, C, l_0, E, F)$, where L is a set of locations, with $l_0 \in L$ the initial location and $F \subseteq L$ is a set of accepting locations, C is a set of clocks, and $E \subset L \times \Phi_C \times \Sigma \times 2^C \times L$ is a set of edges. An edge $e = (l, g, \sigma, R, l')$ shall be interpreted as follows : the automaton moves from location l to a location l' while execution action $\sigma \in \Sigma$ as soon as the valuation of its clocks satisfies guard g. During this move, all clocks from R are reset.

A timed word over an alphabet Σ is a sequence $w = (\sigma_0, d_0)(\sigma_1, d_1), ...$ where each σ_i is a letter from Σ, each $d_i \in \mathbb{R}$ is a date representing the occurrence date of an event of type σ_i, and for every $i < j$, $d_i \leq d_j$. the untiming of a timed word $w = (\sigma_0, d_0)(\sigma_1, d_1) \dots (\sigma_n, d_n)$ is the word $Unt(w) = \sigma_1.\sigma_2 \dots \sigma_n$.

A *run* of an automaton $\mathcal{A}$ is a sequence $(l_0, v_0) \xrightarrow{a_0} (l_1, v_1) \xrightarrow{a_1} (l_2, v_2)...(l_n, v_n)$, where each a_i is either an action $\sigma \in \Sigma$, or a value δ_i from $\mathbb{R}$. We distinguish timed moves, i.e., moves of the form $(l_i, v_i) \xrightarrow{\delta_i} (l_i, v_{i+1})$ where $v_{i+1} = v_i + \delta_i$, and discrete moves, i.e.moves of the form $(l_i, v_i) \xrightarrow{\sigma} (l_{i+1}, v_{i+1})$, such that there exists an edge $(l_i, g, \sigma, R, l_{i+1})$ of $\mathcal{A}$ such that $v_i \models g$, $v_{i+1} = v_{i_{|R}}$. A run is accepting if $l_n \in F$.

A timed word $w = (\sigma_0, d_0)(\sigma_1, d_1), ...$ is *recognized* by $\mathcal{A}$ if there exists an accepting run of $\mathcal{A}$ such that the sequence of letters read in the run and in w coincide (there exists a mapping f from indices of w to discrete moves of the run such that for every $i < j$ $f(i) < f(j)$), and every d_i is equal to the sum of elapsed durations since the beginning of the run, i.e. $d_i = \sum_{j<f(i)} \delta_j$.

The semantics $\mathcal{L}(\mathcal{A})$ of a timed automaton is the set of timed words recognized by $\mathcal{A}$. It is well known that timed automata enjoy some nice properties: reachability of some location, emptiness of $\mathcal{L}(\mathcal{A})$, and omega-regular properties are also decidable. However, inclusion of timed languages is not decidable in general [8]. A timed automaton recognizes timed words which dates need not be integer. The *integer semantics* of a timed automaton is $\mathcal{L}^{int}(\mathcal{A}) = \mathcal{L}(\mathcal{A}) \cap (\Sigma \times \mathbb{N})$. The *untimed semantics* of $\mathcal{A}$ is the set of words $Unt(\mathcal{L}(\mathcal{A})) = \{Unt(w) \mid w \in \mathcal{L}(\mathcal{A})\}$.

Proposition 1 (Integer semantics of timed automata). *Let $\mathcal{A} = (L, C, l_0, E, F)$ be a timed automaton. Then one can build an automaton $\mathcal{A}^{int}$ over an alphabet $\Sigma \cup \{\tau\}$, where τ is an unobservable letter, such that $\mathcal{L}(\mathcal{A}^{int}) = \mathcal{L}^{int}(\mathcal{A})$.*

Proof : We can design an automaton $A^{int} = (L, C', l_0, E', F)$ where

- $C' = C \uplus \{c_t\}$
- $E = \{(l, \{c_t = 1\}, \tau, \{c_t\}, l) \mid l \in L\} \cup \{l, g \wedge c_t = 1, \sigma, R \cup \{c_t\}, l') \mid (l, g, \sigma, R, l') \in E\} \cup \{l, g \wedge c_t = 0, \sigma, R \cup \{c_t\}, l') \mid (l, g, \sigma, R, l') \in E\}$

Then, one can easily prove, by induction on the length of runs that any run of $\mathcal{A}$ that recognizes a timed word with integer dates is also recognized by $\mathcal{A}^{int}$, and conversely that for any accepting run of $\mathcal{A}^{int}$, there exists an accepting run of $\mathcal{A}$ with the same sequence of observable letters and integer dates. □

2.2 Time-Constrained MSCs, HMSCs

Let $\mathbb{R}_{\geq 0}$ denote the set of non-negative reals, $\mathbb{N}$ the set of integers and $\mathcal{I}$ the collection of open and closed intervals with end points in $\mathbb{Q}$ as well as intervals of the form $[c, \infty), (c, \infty)$, where $c \in \mathbb{Q}$. Throughout this report, we fix a finite set $\mathcal{P}$ of processes and let p, q range over $\mathcal{P}$. Let $\Sigma = \{p!q, p?q \mid p, q \in \mathcal{P},\ p \neq q\}$ be the *communication alphabet*. The letter $p!q$ represents p sending a message to q, while $p?q$ signifies p receiving a message sent by q. We define the map $loc : \Sigma \to \mathcal{P}$ via $loc(p!q) = p = loc(p?q)$, and call $loc(a)$ the *location* of a. We define Message Sequence Charts (MSCs) and time-constrained MSCs (TC-MSCs) as usual.

Definition 1. *An* MSC *is a tuple* $(E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda)$. *The set of events is* E *and* $\lambda : E \to \Sigma$ *labels events with letters. For each* p, $<_p$ *is a total order over* $E_p = \{e \in E \mid loc(\lambda(e)) = p\}$. *The message function* $\mu \subseteq E_S \times E_R$ *is a bijection, such that* $f = \mu(e)$ *implies* $\lambda(e) = p!q$, $\lambda(f) = q?p$ *for some* $p, q \in \mathcal{P}$, *with* $E_S = \{e \in E \mid \exists p, q \in \mathcal{P}, \lambda(e) = p!q\}$ *and* $E_R = \{f \in E \mid \exists p, q \in \mathcal{P}, \lambda(f) = q?p\}$. *We require that the transitive closure* $\leq$ *of* $\lessdot = \bigcup_{p\in\mathcal{P}} <_p \cup \mu$ *is a partial order.*

The relation $\leq$ reflects causal ordering of events. We will write $e < f$ when $e \leq f$ and $e \neq f$. Notice that E_p has a unique $<_p$-maximal event (respectively, minimal event), which we refer to as the last (respectively, first) event of E on p. It is frequently assumed in the literature that messages exchanges in a MSC respect a FIFO ordering (messages between pairs of processes p, q are received in the same order as they are sent), but this is not needed in this report.

Definition 2. *A* TC-MSC *is a tuple* $(E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, \delta)$ *where* $(E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda)$ *is an MSC and* $\delta : E \times E \to \mathcal{I}$ *is a function associating an interval* $\delta(e, e') \in \mathcal{I}$ *to each* $e \lessdot e'$.

For each pair of events $e \lessdot e'$, the interval $\delta(e, e')$ constrains the range in which the difference between the occurrence time of e' and the occurrence time of e can lie. For clarity, we shall refer to occurrence times as *dates*. A TC-MSC T defines a collection of MSCs with dates such that the relative differences of dates fulfill the constraints asserted in T.

Definition 3. *Let* $T = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, \delta)$ *be a TC-MSC. A* dated MSC generated by T *is a tuple* $(E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, d)$ *where* $d : E \to \mathbb{R}^+$ *is such that for each* $e \lessdot e'$, $d(e') - d(e)$ *is in the interval* $\delta(e, e')$.

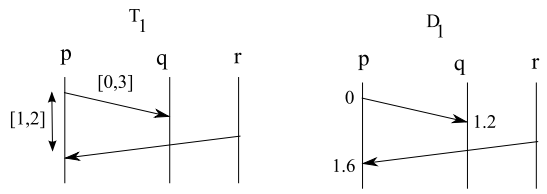

Fig. 1. A TC-MSC T_1 and a dated MSC $D_1 \in \mathcal{D}(T_1)$.

In the rest of the report, we will denote by $\mathcal{T}$ the set of all TC-MSCs, and by $\mathcal{D}$ the set of all dated MSCs. We denote by $\mathcal{D}(T)$ the set of dated MSCs generated by T. Remark that for a given TC-MSC T, we may have $\mathcal{D}(T) = \emptyset$ if the set of constraints attached to pairs of events in T are not satisfiable. We will say that a TC-MSC T is consistent if $\mathcal{D}(T) \neq \emptyset$, and inconsistent otherwise.

Definition 4. *Let T be a TC-MSC with n events. The* language $\mathcal{L}(T)$ *of a TC-MSC T is the set of timed words of the form $w = (a_1, d_1).\dots.(a_n, d_n)$ such that there exists a dated MSC $D = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, d) \in \mathcal{D}(T)$, $a_1 \dots a_n = \lambda(e_1 \dots e_n)$ for a linear extension $e_1 \dots e_n$ of $(E, \leq)$, and for every $i \in 1..n$, $d_i = d(e_i)$. We denote by $\mathcal{L}(T)$ the set of timed words defined by T.*

To capture infinite collections of TC-MSCs, we define TC-MSC graphs as in [5, 15], which are finite graphs whose nodes are labeled by TC-MSCs. Each path ρ of a TC-MSC graph G induces a TC-MSC by concatenating TC-MSCs labeling nodes of ρ. Transitions of G are labeled by interval constraints, one for each process, that act as constraints on the timing between the last and first event of each process in consecutive nodes of ρ.

Definition 5. *A* TC-MSC graph *is a structure $G = (N, \mathcal{T}, \Lambda, n_{in}, N_{fi}, \longrightarrow, \Delta)$ where N is a finite non-empty set of nodes, $\mathcal{T}$ a finite set of TC-MSCs, $\Lambda : N \to \mathcal{T}$ labels each node with a TC-MSC, n_{in} is the initial node, N_{fi} the set of final nodes, $\longrightarrow \subseteq N \times N$ is the transition relation, and Δ is a labeling function which associates an interval $\Delta_p(n{\to}n') \in \mathcal{I}$ to each transition $n{\to}n'$ and each process p, such that $\Delta_p(n{\to}n') = [0, \infty)$ if $\Lambda(n)$ or $\Lambda(n')$ has no event on process p.*

A *path* ρ of the TC-MSC graph G is a sequence $n_0 n_1 \dots n_\ell$ such that $n_0 = n_{in}$ and $n_i{\to}n_{i+1}$ for $i = 0, \dots, \ell - 1$. The path ρ is said to be *final* if $n_\ell \in N_{fi}$. For each $n{\to}n'$, the *concatenation* of TC-MSCs $\Lambda(n)$, $\Lambda(n')$ is defined with respect to $\Delta(n{\to}n')$, and is denoted $\Lambda(n) \circ \Lambda(n')$. Roughly speaking, this consists of putting $\Lambda(n')$ after $\Lambda(n)$ and for every process p, attaching to the pair (e_p, f_p) the constraint $\Delta_p(n{\to}n')$, for e_p the last event of $\Lambda(n)$ on process p and f_p the first event of $\Lambda(n')$ on p. Formally, let $\Lambda(n) = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, \delta)$, $\Lambda(n') = (E', (<'_p)_{p\in\mathcal{P}}, \mu', \lambda', \delta')$. Then $\Lambda(n) \circ \Lambda(n') = (E'', (<''_p)_{p\in\mathcal{P}}, \mu'', \lambda'', \delta'')$ where E'' is the disjoint union of E and E', $<''_p$ is the transitive closure of the union of $<_p$, $<'_p$ and $E_p \times E'_p$, and λ'' is given by: $\lambda''(e) = \lambda(e)$ for $e \in E$, $\lambda''(e) = \lambda'(e)$ for $e \in E'$. We also set $\mu''(e) = \mu(e)$ when $\mu(e)$ is defined, and $\mu''(e) = \mu'(e)$ when $\mu'(e)$ is

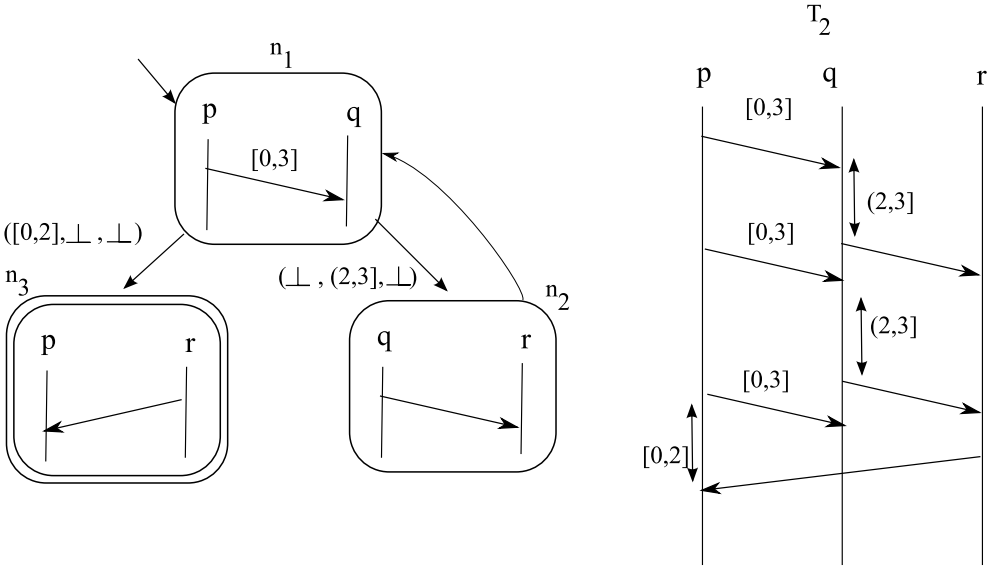

Fig. 2. A TC-MSC graph G_1, and a TC-MSC T_2 in $\mathcal{T}(G_1)$.

defined. At last, δ'' is given by: $\delta''(e,f) = \delta(e,f)$ for $e \lessdot f$, $\delta''(e,f) = \delta'(e,f)$ for $e \lessdot' f$. For each p, if both E_p and E'_p are nonempty, we set $\delta''(e_p, f_p) = \Delta_p(n \to n')$ for e_p the last event of E_p and f_p the first event of E'_p.

We emphasize that by definition, $\Delta_p(n \to n') = [0, \infty)$ if E_p or E'_p is empty. It follows that for $n \to n' \to n''$, $(\Lambda(n) \circ \Lambda(n')) \circ \Lambda(n'')$ is the same as $\Lambda(n) \circ (\Lambda(n') \circ \Lambda(n''))$. Thus, we unambiguously define the TC-MSC T^ρ *induced* by a path $\rho = n_0 \ldots n_\ell$ of G to be $\Lambda(n_0) \circ \ldots \circ \Lambda(n_\ell)$.

The notions of dated MSCs and timed languages generated by a TC-MSC extends naturally to paths. A path ρ of G is called *consistent* if $\mathcal{D}(T^\rho) \neq \emptyset$ (or equivalently, $\mathcal{L}(T^\rho) \neq \emptyset$). We denote by $\mathcal{T}(G)$ the set of all TC-MSCs generated by G, i.e., $\mathcal{T}(G) = \{T^\rho \mid \rho \text{ is a final path of } G\}$. We also denote by $\mathcal{D}(G)$ the union of all dated MSCs generated by G, that is $\mathcal{D}(G) = \{\mathcal{D}(T^\rho) \mid T^\rho \in \mathcal{T}(G)\}$, and by $\mathcal{L}(G) = \mathcal{L}(\mathcal{T}(G))$ the timed language of G.

An example of a TC-MSC graph G_1 is in Figure 2. The TC-MSC T_2 is induced by path $\rho = n_1 \cdot n_2 \cdot n_1 \cdot n_2 \cdot n_1 \cdot n_3$ of G_1, i.e., $T_2 = T^\rho$. As n_3 is final, all dated MSCs in $\mathcal{D}(T_2)$ belong to $\mathcal{D}(G_1)$.

Definition 6. *A TC-MSC graph is* regular *(locally synchronized) iff, for every cycle ρ of G, for every pair of processes p, q that are active in ρ, the TC-MSC T^ρ contains a causal dependency from p to q and another causal dependency from q to p.*

The class of locally synchronized HMSCs (TC-MSC graphs without tile information) was defined in [16, 6], and lifted in [1] to a timed setting. Checking if a

TCMSG is locally synchronized is a co-NP-complete problem. More interestingly, timed languages of locally synchronized TC-MSC graphs can be recognized by event clock timed automata. An event clock automaton is an automaton that has one clock c_a per type of action a in the alphabet of the automaton, and that resets c_a every time action a is played. An interesting property of these event clock automata is that they are closed under complementation, which is not necessarily the case for timed automata in general.

Theorem 1 ([1]). *Let G be a regular TC-MSC graph. Then one can compute a event-clock timed automaton $\mathcal{A}_G$ such that $\mathcal{L}(\mathcal{A}_G) = \mathcal{L}(G)$.*

3 Different interpretation/implementations of time

The pioneering works of [17] and [14] have highlighted one weakness of timed models: the representation of time is idealized. To be more precise, time measurements assumed by most of models are supposed arbitrarily precise. This means that when a guard stipulates that event b must occur at most 10 time units after event a, the implementation of the model system has the capacity to decide whether exactly 10 time units have elapsed according to some canonical time. Supposing that some perfect clock c_a is capable of measuring perfectly the time elapsed since a, one still has to consider that some time can elapse between the date at which decision to execute b is taken (when c_a approaches value 10) and the actual execution of b. This delay is not necessarily constant. So deciding to execute b as soon as $c_a = 10$ means that b could be executed later, violating the specified constraint between a and b.

Another issue is the discrete nature of time measurement. Timed automata specify guards and allow clocks to take real values. However, it might be natural to consider that clocks take integer values to model the discrete nature of clocks in computers.

Imprecision in realistic time is frequently captured by the notion of enlargement and shrinking (guards and constraints are specified up to some small variation ϵ. Puri [17] has proved that for some models, delays can accumulate, and that even the smallest variation ϵ may change the semantics of the model, i.e. its reachable locations, or its timed and untimed languages. This can be harmful, as properties that hold under an idealized semantics might be wrong under realistic time semantics, i.e. for any implementation of the model, regardless of clocks precision.

A large literature has been devoted to robustness for timed automata [17, 12, 11, 13, 10, 9, 14, 18, 20, 21], and an extension of the problem for Petri nets has been proposed [4]. Regardless of the considered model, robustness issues mainly consists in comparing properties and semantics of a model under the idealized time representation and under some realistic representation of time. We propose hereafter several "realistic time" semantics for TC-MSC graphs. We will denote by $[\![G]\!]^\alpha$ the semantics of TC-MSC graph G under semantics α, and define it as the set of dated MSCs generated by G when time follows principles of semantics α (approximate measurement, discrete time,....).

Idealized time As mentioned before, robustness issues are often brought back to comparison of a system with realistic semantics and of the same system with idealized time semantics. For TC-MSC graphs this idealized semantics of time is the standard semantics. We have

$$[\![G]\!]^{id} = \{M \in \mathcal{D}(T^\rho) \mid \rho \text{ is a final path of } G\}$$

$$\mathcal{L}^{id}(G) = \bigcup_{M \in [\![G]\!]^{id}} \mathcal{L}(M)$$

The idealized time semantics of a time constrained MSC graph G considers that all time constraints that are described within a consistent MSC of G can be implemented. For instance, if a pair of events e, f is equipped with a constraint $\delta(e, f) = [10, 10]$, the idealized semantics considers that f must be executed exactly 10 time units after e. Even if the model seems sound, it is quite obvious that no hardware can implement such perfect timing. Forbidding use of singletons as time constraints does not solve the problem: the execution date of a message reception has two preceding events : the sending of a message, and the preceding event on the same process. The execution date of a reception must occur at a date satisfying a pair of constraints, and in practice can hence be reduced to a singleton, even when constraints of the form $\delta(e, f) = [x, x]$ are forbidden. Consider for example the TC-MSC of figure 3. Constraints on dates of events are all of the form $[a, b]$ with $a < b$. However, in this scenarion, the last reception on process q can only occur exactly 4 time units after process p sends the firts message to process r.

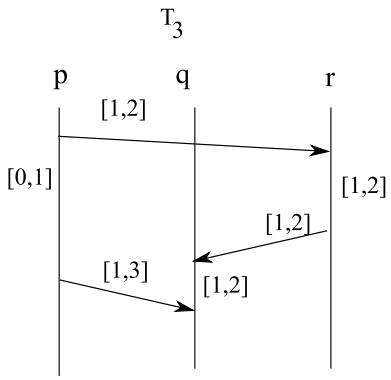

Fig. 3. A TC-MSC T_3.

discrete time semantics The discrete time semantics assumes that all events occur at integral clock values. This is still an idealized semantics of time, but

it is justified by the fact that computers measure time with clocks that tick at regular intervals, and that time is hence measured as a number of ticks.

$$[\![G]\!]^{int} = \{M = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, d) \in \mathcal{D}(T^\rho) \mid \rho \text{ is a final path of } G \wedge \forall e \in E, d(e) \in \mathbb{N}\}$$

$$\mathcal{L}^{int}(G) = \bigcup_{M\in[\![G]\!]^{int}} \mathcal{L}(M)$$

One can immediately notice that $[\![G]\!]^{int} \subseteq [\![G]\!]^{id}$.

Imperfect measurement As already pointed out, measurement of time in computers may be imperfect. Imprecision in measurement can be due to clock imprecision, but also to the way a system is observed. Suppposing that a system is equipped by an observer that collects occurrences of events asynchronously. Observation may introduce some delay. This means that a an event can occur at precisely 10 time units, but that it might be reported as occurring a fraction of seconds earlier or later. In some contexts, one may consider that the date of an event fulfills the time constraints expressed by the specification if there is a different timing of the observed behavior that is ϵ close to a timed behavior that meets all constraints of the specification.

$$[\![G]\!]^{\approx\epsilon} = \left\{ \begin{array}{l} M = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, d) \in \mathcal{D} \mid \\ \exists M' = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, d') \in [\![G]\!]^{id} \\ \wedge \forall e \in E, d(e) \in [d'(e) - \epsilon, d'(e) + \epsilon] \end{array} \right\}$$

$$\mathcal{L}^{\approx\epsilon}(G) = \bigcup_{M\in[\![G]\!]^{\approx\gamma}} \mathcal{L}(M)$$

One can notice that approximation in time measurement does not change the untimed semantics of TC-MSC graphs.

Proposition 2. *For every TC-MSC graph G, and for every $\epsilon \in \mathbb{R}$, we have the following properties.*

- $[\![G]\!]^{id} \neq \emptyset \Rightarrow [\![G]\!]^{\approx\epsilon}$
- $\forall\rho$, *path of* G, $\mathcal{D}(T^\rho) \neq \emptyset \Rightarrow [\![T^\rho]\!]^{\approx\epsilon} \neq \emptyset$
- $Unt(\mathcal{L}^{\approx\epsilon}(G)) = Unt(\mathcal{L}(G))$

Delayed/early triggering semantics As already mentioned in the introduction, enforcing an event b to occur 10 time units after event b can be enforced by a constraint $\delta(a, b) = [10, 10]$; In practice, such a constraint can be implemented using a timer, that is set right after the execution of a, and expires 10 time units later, enforcing execution of b. However, deciding to execute an instruction at a precise date does not mean that the instruction executes immediately, and a small delay might be needed before starting event b. That is, if b is supposed to start 10 time units after a, it may indeed start only $10 + \psi$ time units after a, for some small delay $\psi \in \mathbb{R}$. This can be captured by a *guard enlargement*.

Definition 7. *Let $I = [a, b]$ be an interval, and $\psi, \epsilon \in \mathbb{R}$ be small constants. The* right enlargement *of I by ψ is the interval $I_{+\psi} = [a, b + \psi]$ if $b \neq \infty$ and $I_\psi = [a, \infty)$ otherwise. The enlargement of I by ϵ is the interval $I_{\pm\epsilon} = [\max(0, a - \epsilon), b + \epsilon]$ if $b \neq \infty$ and $I_\psi = [\max(0, a - \epsilon), \infty)$ otherwise. For a set of constraint δ over a set of event dates, the right enlargement of δ by ψ is the constraint $\delta_{+\psi}$ such that $\delta_{+\psi}(e, f) = \delta(e, f)_{+\psi}$ if $\delta(e, f)$ is defined, and is undefined otherwise. The enlargement of δ by ϵ is the constraint $\delta_{\pm\epsilon}$, where $\delta_{\pm\epsilon}(e, f) = \delta(e, f)_{\pm\epsilon}$ if $\delta(e, f)$ is defined, and is undefined otherwise. Last, the right enlargement of a TCMSC $T = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, \delta)$ by ψ is the TCMSC $T_{+\psi} = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, \delta_{+\psi})$ and the enlargement of T by ϵ is the TCMSC $T_{\pm\epsilon} = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, \delta_{\pm\epsilon})$.*

The semantics of a TCMSG under delayed triggering is defined as follows:

$$[\![G]\!]^{+\psi} = \{M \in \mathcal{D} \mid \exists\rho \text{ final path of } G, M \in \mathcal{D}(T^\rho_{+\psi})\}$$

$$\mathcal{L}^{+\psi}(G) = \bigcup_{M\in[\![G]\!]^{+\psi}} \mathcal{L}(M)$$

Similarly, one can consider the semantics of a TCMSG under an eager triggering semantics, that fires events earlier that specified by guards. We define the right shrinking of an interval $[a, b]$ (resp. $[a, \infty)$) by some value ψ as the interval $[a, b - \psi]$ (resp. $[a,_i nfty)$, and define $T_{-\psi}$ accordingly Then the early semantics of G is defined as follows:

$$[\![G]\!]^{-\psi} = \{M \in \mathcal{D} \mid \exists\rho \text{ final path of } G, M \in \mathcal{D}(T^\rho_{-\psi})\}$$

$$\mathcal{L}^{-\psi}(G) = \bigcup_{M\in[\![G]\!]^{-\psi}} \mathcal{L}(M)$$

Enlarged semantics The effect of imperfect measurement of time also has an effect on guards. Beyond triggering effect, one may also consider that a guard, measured using a clock, is satisfied too early, or conversely too late. Hence, satisfaction of a constraint ensuring that a delay between two events should stay within an interval $[a, b]$ might only be implemented up to an imprecision of some ϵ on the lower and upper bound. This justifies the *enlarged semantics* defiend as follows:

$$[\![G]\!]^{\pm\epsilon} = \{M \in \mathcal{D} \mid \exists\rho \text{ final path of } G, M \in \mathcal{D}(T^\rho_{\pm\epsilon})\}$$

$$\mathcal{L}^{\pm\epsilon}(G) = \bigcup_{M\in[\![G]\!]^{\pm\epsilon}} \mathcal{L}(M)$$

Remarks One can immediately notice that $[\![G]\!]^{\approx\delta}$ and $[\![G]\!]^{\pm\delta}$ are different notions. Indeed, if $[\![G]\!]^{id} = \emptyset$, then $[\![G]\!]^{\approx\epsilon} = \emptyset$ for any value of ϵ. On the other hand, one may have $[\![G]\!]^{id} = \emptyset$ and $[\![G]\!]^{+\psi} \neq \emptyset$, or $[\![G]\!]^{\pm\epsilon} \neq \emptyset$ for some values of ψ and ϵ.

One can also remark that $[\![G]\!]^{id} \subset [\![G]\!]^{\approx\epsilon}$. Furthermore, for any value of δ, we have:

$$[\![G]\!]^{int} \subseteq [\![G]\!]^{id} \subseteq [\![G]\!]^{+\epsilon} \subseteq [\![G]\!]^{\pm\epsilon}$$

Note also that $[\![G]\!]^{\pm\epsilon}$ and $[\![G]\!]^{\approx\epsilon}$ are usually uncomparable.

4 Robustness issues for fixed variation of time

Even when a model is designed with an idealized representation of time, and hence does not correspond exactly to the sytem that can be ectually implemented by an architecture, one can still infer properties of a possible implementation of the model. In many cases, several assumptions on the architecture and its precision are known. For instance, clocks are usually imprecise, but during the lifetime of the system, this imprecision will never exceed a threshold δ. Similarly, one can usually assume that clocks are discrete, and hence firing dates of an event usually lay around some integer value of time, ... A natural question is then to consider if there is an important difference between the behavior of the model under idealized time semantics, and the implementable behaviors, and if properties of the model are preserved within an architectural context. This is what we will call robustness question in the rest of the report.

In this section, we review several robustness issues and their decidability for TC-MSC graphs when the nature of clocks and a bound on clock imprecision is known.

4.1 Robust consistency

A first question to ask when a timed specification is designed is whether this specification describes at least one behavior. If this is not the case, then one can consider that this description does not make sense. For timed automata, the emptiness problem consists in checking whether $\mathcal{L}(\mathcal{A}) = \emptyset$. This can occur if guards prevent from reaching an accepting state. For TC-MSC graphs, if a TCMSC T contains an unsatisfiable constraint, then $\mathcal{D}(T) = \emptyset$. Similarly, when a TCMSG G contains only paths ρ such that $\mathcal{D}(T^\rho) = \emptyset$, we have $\mathcal{D}(G) = \emptyset$ $\mathcal{L}(G) = \emptyset$. We will say that a TC-MSC graph is consistent iff $\mathcal{D}(G) \neq \emptyset$.

The *consistency* problem for TC-MSC graphs is defined as follows: given a TC-MSC graph G, determine whether $[\![G]\!]^{id} = \emptyset$, that is, whether it has at least one *consistent and final* path. In [15], it is shown that this problem is undecidable in general.

Theorem 2 ([15]). *The consistency problem for TC-MSC graphs is undecidable.*

The proof of this theorem uses an encoding of a counter machine using constraints in TC-MSCs. Counters are encoded as difference between occurrence dates of last events on processes minus 1, and zero test obtained by enforcing that two events occur at dates that differ by exactly one time unit.

Consistency has been addressed for regular TC-MSCs [5]. Indeed, for a regular TC-MSC graph, one can build a event clock timed automaton A_G which language is the set of timed words in $\mathcal{L}([\![G]\!]^{id})$, and then check if this automaton recognizes at least one timed word.

Theorem 3 ([5]). *The consistency problem is decidable for regular TC-MSC graphs.*

However, the class of regular TC-MSCs is quite restrictive. Further work has shown [2, 3] that checking emptiness for TC-MSC graphs is decidable under a weaker condition on time constraints which does not impose regularity. This restriction is called drift-boundedness. In [2, 3], the authors show that one can test whether a given TC-MSC graph satisfies this condition.

Definition 8. *Let us fix a TC-MSC graph G. Let $\rho = n_0 \dots n_\ell$ be a consistent path of G and $M = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, d)$ be a dated MSC generated by T^ρ. For an integer K, we say that M is a K-*drift-bounded *dated MSC of ρ iff for each $i = 0, \dots, \ell$, for any two events e, e' in $\Lambda(n_i)$, it is the case that $|d(e)-d(e')| \leq K$. We say that ρ is K-*drift-bounded *iff there* exists *a K-drift-bounded dated MSC in $\mathcal{D}(T^\rho)$. We emphasize that $\mathcal{D}(T^\rho)$ may also contain dated MSCs which* are not *K-drift-bounded. We say that G is K-drift-bounded iff every* consistent *(but not necessarily final) path of G is K-drift-bounded.*

In other words, for each *consistent* path ρ, we can find a dated MSC in $\mathcal{D}(T^\rho)$ such that the difference between the dates of any two events from the same instance of a node is at most K. Notice that we can have $[\![G]\!]^{id} = \emptyset$ even though G is K-drift-bounded. In fact, G is vacuously K-drift-bounded for any K if it has no consistent path.

As an example, consider the TC-MSC graph G_1 from Figure 2. G_1 is 3-drift-bounded since in every timed execution, we can be sure that all events in node n_1 or n_2 can be completed within a delay of 3 time units. But if we change the constraints on the loop on n_1 from $([0,1]_r, [0,1]_s)$ to $([4,5]_r, [1,2]_s)$ then for any integer K, G_1 is not K-drift-bounded. Note that G_1 is not locally synchronized. In fact, we can simulate a producer-consumer protocol in which a process continuously sends messages without waiting for an acknowledgement, and obtain non-regular behaviors. Thus, this example cannot be handled by the decidability result in [1].

Drift-boundedness is a practical ans sensible notion. Interpreting a node of a TC-MSC graph as a phase or a transaction of a distributed protocol, we expect any scenario labeling the node to be executable in a bounded time, say K. A protocol specified as a TC-MSC graph that is not K-drift-bounded should thus be considered as ill-formed. Indeed, while a TC-MSC graph specification is usually incomplete (as it abstracts away some events and constraints used in the

actual implementation), if it is not K-drift-bounded, then every implementation of this specification will not be K-drift-bounded either.

Let us first recall the results of [2, 3]. The first result shows that K-drift boundedness is a decidable property.

Theorem 4 ([2]). *Let $K \in \mathbb{N}$ and G be a TC-MSC graph. Then checking whether G is K-drift-bounded is decidable in PSPACE.*

A nice property of drift boundedness is that it allows to check for emptiness.

Theorem 5 ([2]). *Let G be a TCMSG, and let $K \in \mathbb{N}$ be an integer. Then if G is $K-$drift bounded, one can check if $[\![G]\!]^{id} = \emptyset$.*

Now that several decidability issues have been considered, we can address the question of robustness of the consistency property. Given a TCMSG G such that $[\![G]\!]^{id} \neq \emptyset$ and for a choosen $\alpha \in \{int, \approx \epsilon, -\epsilon, \pm\epsilon\}$, is it still the case that $[\![G]\!]^{\alpha} \neq \emptyset$? For a given α, we will call this problem robust consistency of G w.r.t semantics α. For enlarged or approximate semantics, the answer is straightforward:

Proposition 3. *If $[\![G]\!]^{id} \neq \emptyset$, then $[\![G]\!]^{\approx\epsilon} \neq \emptyset$, $[\![G]\!]^{\pm\epsilon} \neq \emptyset$ and $[\![G]\!]^{+\epsilon} \neq \emptyset$*

A direct consequence of this proposition is that robust consistency is guaranteed by definition for all TC-MSC graphs, and and semantics $\alpha \in \{\approx \epsilon, +\epsilon, \pm\epsilon\}$. However, a consequence of theorem 2 is that robust consistency is undecidable in general for TC-MSCs with semantics $\alpha \in \{int, -\psi\}$.

Theorem 6. *Robust consistency is undecidable in general for TC-MSCs with semantics $\alpha \in \{int, -\psi\}$.*

Proof One can easily design a TC-MSC graph G with initial node n_0 that encodes a counter machine as in theorem 2. We build a new TC MSC graph G' by adding a trivial transition from a n_0 to a new node n labeled by a TC MSC which is satisfiable with idealized semantics, but unsatisfiable with integer semantics or $-\psi$ semantics. Then, $[\![G']\!]^{int} \neq \emptyset$ iff the TC-MSC graph G encoding the Minsky machine is consistent, which is undecidable. □

Obviously, remaining within regular TC-MSCs allows for decidability of the emptiness question for any semantics, as the realistic semantics of the TC-MSC graph G can be described by an automaton $\mathcal{A}'$ that can be computed directly from $\mathcal{A}_G$. Indeed, given a regular TC-MSG G and its associated automaton $\mathcal{A}_G$ we have $\mathcal{L}([\![G]\!]^{int}) = \mathcal{L}(\mathcal{A}_G^{int})$ and $\mathcal{L}([\![G]\!]^{\pm\epsilon}) = \mathcal{L}(\mathcal{A}_G^{\pm\epsilon})$, where $\mathcal{A}^{\pm\epsilon}$ is the automaton obtained by enlarging guards of $\mathcal{A}$ by ϵ. Similarly, one can shrink, enlarge guards, and remain in the class of times automata, for which reachability of an accepting state is decidable.

Theorem 7. *The robust consistency problem is decidable for regular TC-MSC graphs, for any semantics $\alpha \in \{int, \approx \epsilon, -\epsilon, \pm\epsilon\}$.*

Again, it is desirable to extend decidability of robust consistency beyond regular TC-MSC graphs.

Theorem 8. *The robust consistency problem is decidable for $K-$drift bounded TCMSGs under integer semantics when constraints are given as integer intervals.*

Proof sketch: We reuse elements of the proof of [3] that builds an automaton which non-emptiness characterizes non-emptiness of $\mathcal{D}(G)$. To decide non-emptiness for K-drift bounded TC-MSC graphs, one can compute an automaton that recognizes consistent accepting paths of G. These path are all K-drift bounded, and states of the automaton contain a node of the TC-MSC, plus a symbolic representation of differences between dates of last events of each process (this can be represented as a set of inequations of the form $x_p - x_q \leq n$, where n is an integer where x_p denotes the occurrence date of the last event on process p). Appending a TC-MSC to a path ending in a state of this form is allowed only if there exists a way to date a TCMSC with dates starting from the set of constraints given by the inequations. This can be decided as a simple linear programming problem. Accepting states are states that end on an accepting node of G. In the integer semantics, the events dates and their differences are integers, so the encoding remains valid. Deciding whether a TC-MSC can be appended to a path can be solved using integer hull techniques (see [19] for details). □

A remaining open question is whether decidability still holds when G has rational constraints, and for early semantics $-\psi$. Both cases can force TC-MSC graphs to drift (the longer a path is, the larger the difference between occurrence dates of last events on processes is), and we conjecture that this leads to undecidability.

4.2 Path Consistency

Consistency is a sanity check that guarantees that the specified set of requirements contains at least one valid execution under some time interpretation fixed by a target architecture. As already pointed out, a path ρ of a TC-MSC graph G may not allow any valid execution under its time constraints. We call such paths *inconsistent* paths. The consistency problem consists in checking whether at least one path of G is consistent. Now, a more precise and interesting question is whether *one paths* of G is inconsistent. Designing a TC-MSC graph with inconsistent accepting paths means that some behaviors that were supposed realizable in an untimed setting are not realizable due to timing constraints.

The *path consistency* question for a TC-MSG G consists in deciding whether there exists an accepting path ρ of G such that $\mathcal{D}(T^\rho) = \emptyset$. We will say that G is *path consistent* if all its paths have a non-empty language.

We conjecture that path consistency is undecidable in general for TC-MSC graphs. However, we can show that for regular and K-drift bounded TC-MSC graphs, this property is decidable.

Theorem 9. *Path consistency of regular TC-MSGs is decidable.*

Proof: It has been shown that one can design a timed automaton $\mathcal{A}_G$ from any regular TC-MSC graph G. This automaton memorizes a node n of G reached while following a path ρ of G, plus a suffix of T^ρ not yet executed. The automaton $\mathcal{A}_G$ moves from one state to another by recognizing an action to be executed (minimal w.r.t ordering) in the suffix, or an action a that is executable within a TC-MSC labeling a transition $t = (n, T, n')$, appending $T \setminus \{a\}$ to the current suffix. To design a timed automaton, a clock is associated to pair of event and predecessors of this event in the TC-MSC graph. Untimed transitions are enriched by guards, that measure whether the constraint w.r.t. the predecessor events is satisfied, and clock resets. It has been proved in [1] that this construction is sound and finite.

We can reuse this construction and design an automaton $\mathcal{B}_G$ that performs invisible τ moves when executing actions from the suffix of states, and moves labeled by T as soon as an action is consumed from a new transition $t = (n, T, n')$. The timed automaton $\mathcal{B}_G$ accepts only executions ending on states with empty suffix, i.e. it recognizes sequences of transitions of G that admit at least one timed word, i.e. consistent paths. Then, one can compare the regular language of $\mathcal{B}_G$ to the sequence of transitions allowed by G. If they differ, then G is not path consistent. $\square$.

Theorem 10. *Path consistency of $K-$drift bounded TCMSGs is decidable.*

Proof sketch : As for the proof of theorem 8, we can reuse the the symbolic automaton of [2]. The symbolic automaton recognizes sequences of TCMSCs with non-empty language, that is consistent paths. If these sequences of TCMSCs are the same as in the original TC-MSC graph, then all paths of G have non-empty language. Conversely, if one can reach a state of the symbolic automaton from which a transition labeleed by a TC-MSC T exists but is not allowed due to inconsistency of constaints in T and in the inequations of the symbolic state, then there is at least one inconsistent path in G.$\square$

Path consistency is a property of TC-MSC graphs, which robustness can be also considered. The *path consistency robustness problem* is hence defined as follows: Given an interpretation α of time, a path consistent TC-MSC graph G, is G path consistent w.r.t α semantics, i.e. is it the case that all path of G have a dated execution under semantics α ?

We know that we have the following implications:

$(\llbracket T^\rho \rrbracket^{int} \neq \emptyset) \Rightarrow (\llbracket T^\rho \rrbracket^{id} \neq \emptyset)$

$(\llbracket T^\rho \rrbracket^{id} \neq \emptyset) \Leftrightarrow (\forall \epsilon \in \mathbb{R}, \llbracket T^\rho \rrbracket^{\approx\epsilon} \neq \emptyset)$

$(\llbracket T^\rho \rrbracket^{id} \neq \emptyset) \Rightarrow (\forall \epsilon \in \mathbb{R}, \llbracket T^\rho \rrbracket^{+\epsilon} \neq \emptyset)$

$\forall \epsilon \in \mathbb{R}, (\llbracket T^\rho \rrbracket^{+\epsilon} \neq \emptyset) \Rightarrow (\llbracket T^\rho \rrbracket^{\pm\epsilon} \neq \emptyset)$

$\forall \epsilon \in \mathbb{R}, (\llbracket T^\rho \rrbracket^{\approx\epsilon} \neq \emptyset) \Rightarrow (\llbracket T^\rho \rrbracket^{\pm\epsilon} \neq \emptyset)$

$\forall \epsilon \in \mathbb{R}, (\llbracket T^\rho \rrbracket^{-\epsilon} \neq \emptyset) \Rightarrow (\llbracket T^\rho \rrbracket^{id} \neq \emptyset)$

Hence, path consistency may change only under integer or early triggering semantics.

Theorem 11. *Let G be a path consistent TC-MSC graph. Then, G is path consistent under semantics $\alpha \in \{\approx \epsilon, +\psi, \pm\epsilon\}$.*

For Regular TC-MSC graphs, we can reuse the construction of $\mathcal{A}_G$ and $\mathcal{B}_G$ described in theorem 9, to obtain the following result.

Theorem 12. *Let G be a regular path consistent TC-MSC graph. Then, one can decide if G is path consistent under any semantics $\alpha \in \{int, \approx \epsilon, +\psi, \psi, \pm\epsilon\}$.*

Proof: For $\alpha \in \{\approx \epsilon, +\psi, \pm\epsilon\}$, path consistency of G guarantees path consistency of G under α semantics. Now, if $\alpha \in \{int, -\psi\}$, we can build the automaton $\mathcal{B}_G$ as in theorem 9, that recognizes sequences of TC-MSCs of G that have a timed execution. Then, one can adapt this timed automaton to build an automaton $\mathcal{B}_G^{int}$ that recognizes sequences of TC-MSCs which languages contain timed words with exclusively integer dates, following the construction in Proposition 1. Similarly, one can build an automaton $\mathcal{B}_G^{\alpha}$, that recognizes sequences of TC-MSCs recognized under shrinked semantics $-\psi$ by shrinking guards of $\mathcal{B}_G$. Then, G is path consistent under integer (resp. early) semantics iff $\mathcal{L}(\mathcal{B}_G) = \mathcal{L}(\mathcal{B}_G^{int})$ (resp. $\mathcal{L}(\mathcal{B}_G) = \mathcal{L}(\mathcal{B}_G^{-\psi})$). $\square$

We conjecture that robustness of path consistency is undecidable in general for $\alpha \in \{int, -\psi\}$, even for K-drift bounded TC-MSC graphs, and is decidable for K-drift bounded TC-MSC graphs with integer constraints under integer semantics.

4.3 Untimed semantics preservation

Now, path consistency only considers paths of TC-MSC graphs, and not the contents of the assembled scenarios. We would like now to address robustness issues that consider whether the language depicted by a specification changes under a different implementation of time. Of course, requiring the timed language of a specification to remain identical under change of time implementation is requiring too much, as changing the semantics relaxes of reinforces existing constraints on execution dates of events. A more reasonable approach is to require that the untimed languages, that is the logical dependencies among events remain unchanged.

Definition 9. *Let $T = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda, \delta)$ be a TC-MSC. The unconstrained version of T is denoted by T^∞, and is the MSC $T^\infty = (E, (<_p)_{p\in\mathcal{P}}, \mu, \lambda)$. For a TC-MSC graph$G = (N, \mathcal{T}, \Lambda, n_{in}, N_{fi}, \longrightarrow, \Delta)$, the unconstrained version of G is the TC-MSC graph $G^\infty = (N, \mathcal{T}, \Lambda^\infty, n_{in}, N_{fi}, \longrightarrow, \Delta^\infty)$, where*

- $\Lambda^\infty(n) = (\Lambda(n))^\infty$
- $\Delta^\infty(n, n')[p] = [0, \infty)$

We will say that a TC-MSC graph has a *robust untimed semantics* if and only if $Unt(\mathcal{L}([\![G]\!]^{id})) = Unt(\mathcal{L}([\![G^\infty]\!]^{id}))$

Theorem 13. *One can not decide if a TC-MSC graph G has a robust untimed semantics, even if G is K-drift bounded.*

Proof sketch: This can be brought back to HMSC language inclusion problem. Take H_1, H_2 two HMSCs with initial nodes n_1, n_2. It is well known that one can not decide whether $\mathcal{L}(()H_1) \subseteq \mathcal{L}(()H_2)$. Not, we can build a TC-MSC graph G, with initial node n_0, all transitions of H_1 and H_2, and additional transitions $t_1 = (n_0, T, n_1)$ and $t_2 = (n_0, T', n_2)$, where T is an inconsistent TC-MSC, and T' is the unconstrained version of T. Obviously, in the untimed semantics, G can recognize T, then behave as H_1, or T', and behave as H_2. In the timed semantics, G can only take transition labeled by T' and behave as in H_2. Then the untimed semantics of G is robust if and only $\mathcal{L}(H_1) \subseteq \mathcal{L}(H_2)$.□

The above encoding works even if G is drift-bounded. Again, one way to obtain decidability is to consider regular TC-MSC graphs.

Theorem 14. *One can decide if a regular TC-MSC graph G has a robust untimed semantics.*

Proof: One can design the timed automaton $\mathcal{A}_G$, and similarly, following the construction of [6], for an untimed TC-MSC graph, one can design another automaton $\mathcal{A}_G^U$ recognizing $Unt(\mathcal{L}(G))$. It then suffices to compare the untimed language of $\mathcal{A}_G$ with the language of $\mathcal{A}_G^U$.□

We can derive a robustness question with respect to semantics α from this question. We will say that a TCMSG has a robust untimed semantics w.r.t a chosen semantics α if and only if $Unt(\mathcal{L}([\![G]\!]^\alpha)) = Unt(\mathcal{L}([\![G]\!]^{id}))$.

A specification has a robust untimed semantics w.r.t. time intrepretation α if it exhibits the same behavior that the idealized semantics under time interpretation α. Obviously, changing the interpretation of time may affect constraints on valid dates, and hence timed behavior, already at the level of TC-MSCs. As already mentioned, considering integer dates or shrinking constraints allow less behaviors that in the idealized semantics. That is, for every G, we have $Unt([\![G]\!]^{int}) \subseteq Unt([\![G]\!]^{id})$ and $Unt([\![G]\!]^{-\psi}) \subseteq Unt([\![G]\!]^{id})$.

Similarly, enlargement can only result in a larger set of timed executions of a TC-MSC graph. For every G and every $\epsilon \in \mathbb{R}$, we have $[\![G]\!]^{id} \subseteq [\![G]\!]^{+\epsilon} \subseteq [\![G]\!]^{\pm\epsilon}$, and consequently $Unt([\![G]\!]^{id}) \subseteq Unt([\![G]\!]^{+\epsilon}) \subseteq Unt([\![G]\!]^{\pm\epsilon})$.

However, comparing behavior only with respect to respective dates of events does not bring a lot of information, as it is quite obvious that enlarging guards allows new behaviors. Now, it seems important to discover when time interpretation adds or remove untimed behaviors, that is modifies the logics of the specification.

Theorem 15. *Robustness of TC-MSC graphs semantics w.r.t a semantics α is undecidable, even for K-drift bounded TC-MSC graphs for any $\alpha \in \{int, +\psi, -\psi, \pm\epsilon\}$.*

Proof: We can reuse the encoding of HMSC language inclusion in theorem 13, but for each α, replace TC-MSC T that is unsatisfiable by a TC MSC T_α that is satisfiable under idealized time semantics, and unsatisfiable under semantics α.

The converse encoding with T_α unsatisfiable under idealized time semantics, and satisfiable under semantics α proves undecidability for semantics that enlarge time constraints. □

Theorem 16. *Robustness of regular TCMSG untimed languages w.r.t a semantics α is decidable.*

Proof : First, we can build build an automaton $\mathcal{A}_G$ that recognizes timed words in $[\![G]\!]^{id}$. To compare this language to $[\![G]\!]^{int}$ we can design a new automaton $\mathcal{A}^{int}$ that recognizes timed words with integer dates. One can then design region automata for $\mathcal{A}$ and $\mathcal{A}^{int}$ (they recognize respectively the untimed words in $[\![G]\!]^{id}$ and $[\![G]\!]^{int}$) and then compare their language (nb: this comparison is PSPACE in the size of the automata, which can be of exponential size w.r.t. the original TC-MSC graph).

Then, it has been shown that the enlarged semantics (by a small value ϵ) of a timed automaton $\mathcal{A}$ is also recognized by a timed automaton $[\![\mathcal{A}]\!]^{\pm\epsilon}$ obtained by enlarging guards in $\mathcal{A}$. Hence, to compare $[\![G]\!]^{id}$ with $[\![G]\!]^{\pm\epsilon}$ for a regular TCMSg, it suffices to build $\mathcal{A}_G$, enlarge it, and then compare the untimed languages of both automata.

A similar construction can be performed for delayed semantics $+\psi$ by right enlarging of guards in $\mathcal{A}_G$, and for early semantics $-\psi$ by shrinking guards in $\mathcal{A}_G$. □

5 Robustness issues for arbitrary precision

Former section considers various interpretation of timed implementation, and robustness of specifications with respect to a choice of architecture, and a known bound ϵ on imprecision of time measurement or implementation. Such parameter is not necessarily known, and in addition, improving performances and accuracy of a system may help reducing imprecision, and preserve some properties.

However, there are situations in which even the smaller imprecision may change the overall behavior and properties of a system. Consider for instance the TCMSG G_2 of Figure 4, and in particular the TC-MSC of node n_3. Let us call d_1 the occurrence date of the message sending from p to r and d_2 the occurrence date of the message from q to r in n_3. The TCMSC in node n_3 is realizable within a context where the difference between d_2 and d_1 is at most 2. Now, one can note that any path of this example is a sequence of nodes of the form $n_1.n_2^k.n_3$, and that after playing n_1, the date of the last executed event on q is greater by 15 time units than the date of the last event executed by p. After every occurrence of n_2 satisfying the attached constraints, the difference between dates of the last events on p and q can only be preserved or increased. Hence, no path of G_2 generates a correct dated MSC, and G_2 is inconsistent.

Now, if we enlarge all constraints by some small value ϵ, every occurrence of n_2 can be used to reduce the possible difference between the dates of the last events on p and q by $2.\epsilon$ in $\pm\epsilon$ semantics. Hence a path of the form $n_1.n_2^{\frac{13-2.\epsilon}{2.\epsilon}}.n_3$

admits a dated MSC. One can notice that this remark holds for any value of ϵ, but this needs not be always the case.

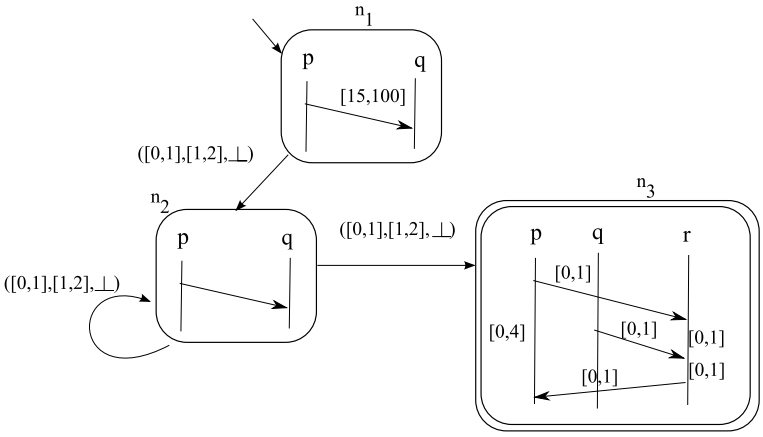

Fig. 4. A TC-MSC graph G_2 : Enlargement modifies the language and consistency

Now a question that arises is whether there exists a precision, i.e. a value of parameter ϵ that preserves or guarantees good properties of a system. Consider for example the TC-MSC graph G_3 of Figure 5. The constraint $([0, 0.9][1, 2])$ on the loop of node n_2 implies that the TC-MSC graph is not K-drift bounded, as process q tends to receive message at dates that are further from their sending date by process p at every iteration of n_2. However, an enlargement of the constraint by 0.1 avoids this delay accumulation. Conversely, the constraint $\delta(n_1, n_1)$ for the loop on node n_1 does not force G to drift, but for any small ψ the semantics $[\![G]\!]^{-\psi}$ drifts, as any passage through n_1 forces process q's last event to be later with respect to process p's last event by at least ψ time units. This example shows that there are situation where the smallest enlargement or shrink of constraints may change the overall property of a TC-MSC graph with respect to consistency, path consistency, and even drift.

From a practical point of view, this means that even very accurate architectures might be unable to preserve good properties of a specification. This raises the issue of robustness problems for architectures with arbitrary precision. We can define the following problems for arbitrary precision:

i) Robust consistency : Given G such that $[\![G]\!]^{id} \neq \emptyset$, and a semantics α , does there exists $\epsilon \in \mathbb{R}, \forall 0 < \epsilon' \leq \epsilon, [\![G]\!]^{\alpha} \neq \emptyset$ with enlargement/right enlargement/shrink of ϵ' ?
ii) Robust path consistency : Given G that is path consistent, and a semantics α, does there exists $\epsilon \in \mathbb{R}, \forall 0 < \epsilon' \leq \epsilon, [\![G]\!]^{\alpha}$ is path consistent with enlargement/right enlargement/shrink of ϵ' ?

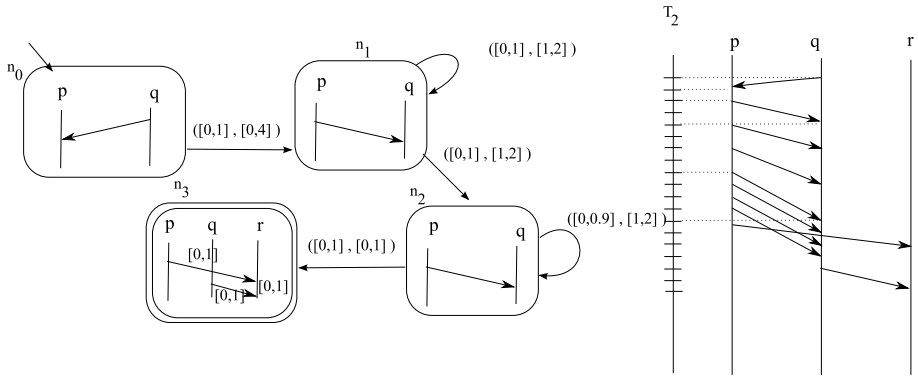

Fig. 5. A TC-MSC graph G_3 : modification of drift by enlargement

iii) Robust semantics : Does there exists $\epsilon, \in \mathbb{R}, \forall 0 < \epsilon' \leq \epsilon, Unt([\![G]\!]^{\alpha}) = Unt([\![G]\!]^{id})$ with enlargement/right enlargement/shrink of ϵ' ?

Of course , we consider the the semantics α ranges in $\{\approx \epsilon, \pm\epsilon, -\psi, +\psi\}$, because the questions do not make sense for integer semantics, that is not parameterized by any value.

Theorem 17. *Let G be a regular TC-MSC graph. Then one can decide the robust enlargement, robust path consistency, and robust semantics problems.*

Proof sketch: As G is regular, one can compute a timed automaton $\mathcal{A}_G$, and an (untimed) automaton $\mathcal{B}$ that recognizes the untimed language of $\mathcal{A}_G$. Then, one can apply the techniques of [11] to check whether there exists some value ϵ satisfying the robust consistency, robust path consistency, and robust semantics requirements. We can use the timed automaton recognizing paths of G to check path consistency, and we can compare $Unt(\mathcal{L}(()\mathcal{A}_G))$ with $Unt(\mathcal{L}(()\mathcal{A}_G^{\alpha}))$, where $\mathcal{A}_G^{\alpha}$ is the timed automaton built by shrinking or enlarging guards of $\mathcal{A}_G$. $\square$

We already know, from theorem 15 that for a given ϵ, one can not check properties such as consistency robustness, path consistency robustness, and robustness of a TC-MSC graph w.r.t any semantics of parameter ϵ. Furthermore, there exist TC-MSC graphs for which these undecidability results hold for any value of ϵ. One can design TC-MSC graphs that are consistent under idealized semantics, and inconsistent under $[\![]\!]^{-\psi}$, for any value of ψ. One can also design TC-MSC graphs that are trivially consistent under idealized time semantics, and which consistency depends on consistency of a part of TC-MSC graph encoding runs of a Minsky machine. Last, one can design TC-MSC graphs which language is not preserved by $[\![]\!]^{-\psi}, [\![]\!]^{+\epsilon}, [\![]\!]^{\pm\epsilon}$ for any value of ϵ. Though we have $[\![G]\!]^{-\psi} \subseteq [\![G]\!]^{-\psi'} \subseteq [\![G]\!]^{id}$ for every $\psi < \psi'$, and $[\![G]\!]^{id} \subseteq [\![G]\!]^{\pm epsilon} \subseteq [\![G]\!]^{\pm epsilon'}$ for $\epsilon < \epsilon'$, an algorithm starting from an arbitrary value of ψ or ϵ to reach a fixpoint

satisfying the checked property is not guaranteed to terminate, and at each step would have to perform and undecidable check.

Hence we have the following conjecture:

Conjecture : Robust consistency, robust path consistency and robust semantics w.r.t. semantics $\alpha \in \{int, \approx \epsilon, -\psi, \pm\epsilon, +\psi\}$ under arbitrary precision are undecidable in general for TC-MSC graphs and $K-$drift bounded TC-MSC graphs.

6 Conclusion

We summarize in the following tables the results of the paper: We consider the emptiness problem supposing that the considered idealized time TC-MSC graph is non-empty fro the emptiness problem,that is describes at least one execution and consistent for the consistency problem. First of all we consider decidability results for regular TC-MSC graphs.

	int	$\approx \epsilon$	$+\psi$	$-\psi$	$\pm\epsilon$
Robust consistency	decidable (Thm 7)	non-emptiness guaranteed (Prop 2)	non-emptiness guaranteed (Prop 3)	decidable (Thm 7)	non-emptiness guaranteed (Prop 3)
Path Consistency	decidable (Thm 12)	guaranteed (Prop 2)	guaranteed (Thm 11)	decidable (Thm 12)	guaranteed (Thm 11)
semantics preservation	decidable (Thm 16)	guaranteed (Prop 2)	decidable (Thm 16)	decidable (Thm 16)	decidable (Thm 16)
Problems with arbitrary precision	irrelevant	guaranteed (Prop 2)	decidable (Thm 17)	decidable (Thm 17)	decidable (Thm 17)

Fig. 6. Decidability of robustness problems for **regular** TC-MSC graphs w.r.t different interpretations of time.

Let us now address decidability for non-regular TC-MSC Graphs. As for the regular case, we suppose that the considered TC-MSC is consistent (resp path consistent) when checking robust (path) consistency:

As one can notice, leaving the regular class makes most of robustness issues undecidable, except for those that are guaranteed by definition.

As future work, we would like to complete the above pictures, and answer fully the questions of path consistency for integer and early semantics, and the conjecture on robustness problems with arbitrary precision outside the class of regular TC-MSC graphs.

	int	$\approx \epsilon$	$+\psi$	$-\psi$	$\pm\epsilon$
Robust consistency	undecidable (Thm 6)	non-emptiness guaranteed (Prop 2)	non-emptiness guaranteed (Prop 3)	undecidable (Thm 6)	non-emptiness guaranteed (Prop 3)
Path Consistency	???	guaranteed (Prop 2)	guaranteed (Thm 11)	???	guaranteed (Thm 11)
semantics preservation	undecidable (Thm 15)	guaranteed (Prop 2)	undecidable (Thm 15)	undecidable (Thm 15)	undecidable (Thm 15)
Problems with arbitrary precision	irrelevant	guaranteed (Prop 2)	conjectured undecidable	conjectured undecidable	conjectured undecidable

Fig. 7. Decidability of robustness problems for TC-MSC graphs w.r.t different interpretations of time.

	int	$\approx \epsilon$	$+\psi$	$-\psi$	$\pm\epsilon$
Robust consistency	decidable (with integer intervals) (Thm 8)	non-emptiness guaranteed (Prop 2)	non-emptiness guaranteed (Prop 3)	???	non-emptiness guaranteed (Prop 3)
Path Consistency	???	guaranteed (Prop 2)	guaranteed (Thm 11)	???	guaranteed (Thm 11)
semantics preservation	undecidable (Thm 15)	guaranteed (Prop 2)	undecidable (Thm 15)	undecidable (Thm 15)	undecidable (Thm 15)
Problems with arbitrary precision	irrelevant	guaranteed (Prop 2)	conjectured undecidable	conjectured undecidable	conjectured undecidable

Fig. 8. Decidability of robustness problems for K-drift bounded TC-MSC graphs w.r.t different interpretations of time.

References

1. S. Akshay, Paul Gastin, Madhavan Mukund, and K. Narayan Kumar. Model checking time-constrained scenario-based specifications. In *Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010*, volume 8 of *LIPIcs*, pages 204–215. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.
2. S. Akshay, Blaise Genest, Loïc Hélouët, and Shaofa Yang. Regular set of representatives for time-constrained msc graphs. *Inf. Process. Lett.*, 112(14-15):592–598, 2012.
3. S. Akshay, Blaise Genest, Loïc Hélouët, and Shaofa Yang. Symbolically bounding the drift in time-constrained msc graphs. In *Theoretical Aspects of Computing - ICTAC 2012*, volume 7521 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2012.
4. S. Akshay, Loïc Hélouët, Claude Jard, and Pierre-Alain Reynier. Robustness of time petri nets under guard enlargement. In *Proc. of RP 2012*, volume 7550 of *LNCS*, pages 92–106. Springer, 2012.
5. S. Akshay, Madhavan Mukund, and K. Narayan Kumar. Checking coverage for infinite collections of timed scenarios. In *CONCUR 2007 - Concurrency Theory*, volume 4703 of *Lecture Notes in Computer Science*, pages 181–196. Springer, 2007.
6. R. Alur and M. Yannakakis. Model checking of Message sequence Charts. In *Proc. of CONCUR'99*, number 1664 in LNCS, pages 114–129. Springer, 1999.

7. Rajeev Alur. Timed automata. In *Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 8–22. Springer, 1999.
8. Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
9. Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust model-checking of linear-time properties in timed automata. In *Proc. of LATIN'06*, volume 3887 of *LNCS*, pages 238–249, 2006.
10. Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust analysis of timed automata *via* channel machines. In *Proc. of FoSSaCS'08*, volume 4962 of *LNCS*, pages 157–171. Springer, 2008.
11. Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust model-checking of timed automata via pumping in channel machines. In Uli Fahrenberg and Stavros Tripakis, editors, *Proceedings of the 9th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'11)*, volume 6919 of *Lecture Notes in Computer Science*, pages 97–112, Aalborg, Denmark, September 2011. Springer.
12. Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robustness in timed automata. In Parosh Aziz Abdulla and Igor Potapov, editors, *Proceedings of the 7th Workshop on Reachability Problems in Computational Models (RP'13)*, volume 8169 of *Lecture Notes in Computer Science*, pages 1–18, Uppsala, Sweden, September 2013. Springer.
13. Martin De Wulf, Laurent Doyen, Nicolas Markey, and Jean-François Raskin. Robust safety of timed automata. *Formal Methods in System Design*, 33(1-3):45–84, 2008.
14. Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Systematic implementation of real-time models. In *Formal Methods (FM'05)*, volume 3582 of *LNCS*, pages 139–156. Springer, 2005.
15. Paul Gastin, Madhavan Mukund, and K. Narayan Kumar. Reachability and boundedness in time-constrained MSC graphs. In Kamal Lodaya, Madhavan Mukund, and R. Ramanujam, editors, *Perspectives in Concurrency Theory*, IARCS-Universities, pages 157–183. Universities Press, January 2009.
16. Anca Muscholl and Doron Peled. Message sequence graphs and decision problems on mazurkiewicz traces. In *MFCS*, pages 81–91, 1999.
17. Anuj Puri. Dynamical properties of timed automata. *In DEDS*, 10(1-2):87–113, 2000.
18. Ocan Sankur. Untimed language preservation in timed systems. In *Proc. of MFCS'11*, LNCS. Springer, 2011.
19. Alexander Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1998.
20. Mani Swaminathan and Martin Fränzle. A symbolic decision procedure for robust safety of timed systems. In *TIME 2007*, page 192. IEEE Computer Society, 2007.
21. Mani Swaminathan, Martin Fränzle, and Joost-Pieter Katoen. The surprising robustness of (closed) timed automata against clock-drift. In *TCS*, pages 537–553. Springer, 2008.