Timed pattern diagnosis in timed workflows: a model checking approach
Yannick Pencolé, Audine Subias

To cite this version:
Yannick Pencolé, Audine Subias. Timed pattern diagnosis in timed workflows: a model checking approach. 14th Workshop on Discrete Event Systems (WODES 2018), May 2018, Sorrento Coast, Italy. 2018. <hal-01827362>

HAL Id: hal-01827362
https://hal.laas.fr/hal-01827362
Submitted on 2 Jul 2018
Timed pattern diagnosis in timed workflows: a model checking approach

Yannick Pencolé * Audine Subias. *

* LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
(y pencol@laas.fr, subias@laas.fr).

Abstract
In this article we introduce the problem of timed pattern diagnosis in timed system and propose a method to solve a subclass of these problems: the problem of timed pattern diagnosis of workflows. It consists in searching for all the possible evolutions of the workflow that are consistent with a timed sequence of observations and determining whether these evolutions match a pattern of timed events instead of a single fault event. The formal characterization of the diagnosis problem is based on the notion of pattern matching. Defined as a reachability problem, the diagnosis problem is then solved by model checking.

Keywords: Diagnosis, Time Petri Net, Timed Pattern, Model checking

1. INTRODUCTION
The problem of fault diagnosis in DESs was introduced in Sampath et al. (1995). Later, extensions of the problem handling time have been proposed using timed automata (Tripakis (2002)) and more recently Petri nets with several interpretations of time and methods, for instance: Chatain and Jard (2005), Jiroveanu and Boel (2006), Basile et al. (2007), Bonhomme (2015), Boel and Jiroveanu (2013), Basile et al. (2015), Liu (2014), Wang et al. (2015). These contributions describe some methods for the same problem that is the diagnosis of single faults in a timed system. In this article we propose to extend the diagnosis problem to more complex behaviors than single events called timed patterns and to apply it to timed workflows. Timed patterns extend with time the classical notion of pattern (Jéron et al. (2006); Gougam et al. (2017)). Beyond faulty behaviors, classical patterns can represent any untimed normal behavior of interest or any behavior associated with a system specification as in Jiang and Kumar (2004). In Bozzano et al. (2015), such patterns are called diagnosis conditions and are represented with LTL (Linear Time Logic) properties with past operators.

As classical patterns, the proposed timed pattern models more complex behaviors than a single event (sequences, concurrent events,...) but it also add the possibility to model a time interval between the occurrence of two events in the patterns. Timed pattern can then naturally model unexpected/unacceptable faulty time delays between two unobservable events. In this article we firstly propose to formally define the general problem of timed pattern diagnosis in timed system. We then propose a method based on model-checking techniques that solves a subclass of this new problem that is the timed pattern diagnosis in timed workflows. Given the observable part of the run of a workflow, the diagnosis problem consists in determining whether the considered timed pattern has occurred or not during the run of the workflow. The proposed method first converts the problem into a reachability problem and then use model checking tool to solve it. Section 2 formally defines the general problem. The proposed method is presented in Section 3. An illustrative example about a service workflow is described in Section 4.

2. GENERAL PROBLEM STATEMENT
The objective of the approach is to diagnose the occurrence of a timed behavior (i.e. a timed pattern) in a timed system. It supposes the implementation of a diagnosis function that takes as input a sequence of observations and returns one of the following three results:

(1) The pattern’s behavior has certainly occurred.
(2) The pattern’s behavior is certainly absent.
(3) The pattern’s behavior may have occurred, ambiguity.

This section defines the different parts of the problem that is: how to model timed systems/timed patterns, how to formally define the occurrence of a timed pattern, what is finally the diagnoser function.

2.1 Preliminaries
This paper deals with timed sequences. Let $\Sigma$ be an alphabet that is a finite set of symbols, each symbol represents an event. A timed sequence over $\Sigma$ is an element of $T(\Sigma) = (\mathbb{N}^+ \times (\Sigma \cup \lambda))^*$. Suppose for instance that $\Sigma = \{a, b, c\}, \rho = (a, 3), (\lambda, 1), (c, 1), (c, 1), (b, 2), (\lambda, 5)$ also denoted $3a1\lambda1c1c2b5\lambda$ is a timed sequence, $\rho' = 1b14\lambda0\lambda$ is another one. Symbol $\lambda$ defines a timepoint (basically $1\lambda2\lambda$ is the same as $3\lambda$). The empty sequence of $T(\Sigma)$ is given by $\emptyset$.

For any timed sequence it is possible to define a canonical form $\delta_1 e_1 \ldots \delta_n e_n \delta_{n+1} \lambda, e_i \in \Sigma$ that is obtained by removing useless time points. For instance the canonical form of $\rho = 3a1\lambda1c1c2b5\lambda$ is $3a2c1c2b5\lambda$. In this article and without loss of generality only canonical forms are considered. The projection of a timed sequence $\rho$ over a sub-set $\Sigma_{\rho}$ of $\Sigma$ (denoted $P_{\Sigma_{\rho}}(\rho)$) is the timed sequence obtained by removing any symbols that does not belong to $\Sigma_{\rho}$ and by summing the durations between two symbols that are still present after the removal. Let us consider the canonical form of $\rho$, it corresponds...
to an event \( a \) that occurs at date 3 after the origin of time, an event \( c \) that occurs 2 units of time after \( a \) (so \( c \) occurs at date 5), followed by an event \( b \), 1 unit of time later (so at date 6), and so on. Given \( \Sigma_p = \{a, b\} \) then \( P_{\Sigma_p}(\rho) = 3a5b5a\).

Any sub-set \( (\text{canonical sub-set}) \) of \( T(\Sigma) \) is a timed language over \( \Sigma \). Let \( \text{time}(\rho) \) denote the sum of all durations in \( \rho \), then \( \text{time}(3a0b2a1b2b4\lambda) = 3 + 4 + 2 + 1 + 2 + 4 = 14 \). \( \rho' \) is a prefix of \( \rho \) if \( \text{time}(\rho') \leq \text{time}(\rho) \) and there exists \( \rho'' \) such that \( \rho' \rho'' \) and \( \rho \) have the same canonical form.

### 2.2 Matching

Like untimed discrete event system, a timed discrete event system generates a finite set of events \( \Sigma \). The difference is that in the timed case, the possible time durations between the occurrence of two successive events are defined, whereas in the untimed case, the events are only connected by a precedence relationship. Of course in both cases (timed and untimed), two events may not be related i.e even the order of their occurrence is not known. An evolution of a timed discrete event system is a timed sequence on its alphabet. The set of possible evolutions of a timed system is thus a set of timed sequences, that is, a timed language.

**Definition 2.1.** (Timed system). A timed discrete event system whose set of events is \( \Sigma \), is a timed language \( L \subseteq T(\Sigma) \) such that if \( \rho \in L \) then any of its prefixes is in \( L \).

The set of events \( \Sigma \) generated by the system is partitioned into two subsets: observable events \( \Sigma_o \) and unobservable events \( \Sigma_u \). Two timed sequences can be linked by the notion of matching that states whether a sequence is present inside another.

**Definition 2.2.** (sequence matching). A sequence \( \rho \in T(\Sigma) \) matches a sequence \( \sigma \in T(\Sigma) \), denoted \( \rho \propto \sigma \), if:

- \( \sigma = \delta \lambda \) and \( \text{time}(\rho) \geq \delta \), with: \( \delta \in \mathbb{N} \); or
- \( \sigma = \delta_0 \epsilon_0 \sigma_1 \), such that: \( \delta_0 \in \mathbb{N} \), \( \epsilon_0 \in \Sigma \), \( \sigma_1 \in T(\Sigma) \) and there exist two timed sequences \( \rho_0, \rho_1 \in T(\Sigma) \) and a duration \( \delta \in \mathbb{N} \) such that:
  1. \( \rho = \rho_0 \delta \epsilon_0 \rho_1 \);
  2. \( \delta_0 = \text{time}(\rho_0) + \delta \);
  3. \( \rho_1 \propto \sigma_1 \).

In other words, \( \rho \) matches \( \sigma \) if \( \sigma \) is a sub-word of \( \rho \). Figure 1 illustrates how the sequence \( \rho = 2b3a2a5a1c1c3a3b3 \) matches the sequence \( \sigma = 7a7c7d7\lambda \). The first event of \( \sigma \) is \( a \) and occurs at date 7. In \( \rho \), there is also an event \( a \) that occurs at date 7, 2 + 3 + 2. The second event of \( \sigma \) is \( c \) that occurs 7 time units later than \( a \). This event is matched with the third event \( c \) of \( \rho \) that occurs 5 + 1 + 1 + 7 = 14 time units after \( a \). Finally, \( \sigma \) endsinstantaneously after event \( c \) while \( \rho \) lasts 3 time units more so \( \rho \) matches \( \sigma \).

We define a timed pattern as a set \( \mathcal{R} \) of timed sequences which define a behavior of interest. For example, it may be the occurrence of a single event in a given time period, the occurrence of several events that occur in any order, but whose occurrence dates have constraints or the occurrence of several events that occur in a fixed order with specific time constraints. As in the original diagnosis problem that considers unobservable fault event, we consider that the behavior of the pattern is not observable i.e no event of the pattern is observable in the system.

**Definition 2.3.** (Timed pattern). A timed pattern over a set of events \( \Sigma_R \subseteq \Sigma_u \), is a timed language \( R \subseteq T(\Sigma_R) \) such that if

\[\Delta_R : T(\Sigma_o) \rightarrow \{\text{R-certain}, \text{R-absent}, \text{R-ambiguous}\}\]

such that:

- \( \Delta_R(\sigma) = \text{R-certain} \) if for any sequence \( \rho \in L \) such that \( P_{\Sigma_o}(\rho) = \sigma \), \( \rho \) is \( \mathcal{R} \)-matching;
Considering a set of patterns $\mathcal{R}_1, \ldots, \mathcal{R}_n$, the diagnoser function is defined by:

$$\Delta : \mathcal{T}(\mathcal{N}_0) \rightarrow \prod_{r=1}^{n}(\mathcal{R}_r \rightarrow \text{certain}, \mathcal{R}_r \rightarrow \text{absent}, \mathcal{R}_r \rightarrow \text{ambiguous})$$

such that $\Delta(\sigma) = (\Delta_{\mathcal{R}_1}(\sigma), \ldots, \Delta_{\mathcal{R}_n}(\sigma))$.

3. DIAGNOSIS BY MODEL CHECKING

The proposed method defines the problem of timed pattern diagnosis as a problem of reachability and uses model checking techniques to solve it. This resolution relies on the TINA toolbox (Time Petri Net Analyzer) (Berthomieu et al. (2004)). Given a system and a pattern, the method proceeds as follows.

1. The timed languages of the system and of the pattern are modeled by L-type Labeled Time Petri nets respectively denoted $\Theta$ and $\Omega$.
2. From the two previous nets, a new net is built ($\Theta_{||\Omega}$) that captures the matching relation between the system and the pattern.
3. The input observation sequence is transformed into an L-type Labeled Time Petri net (denoted $\text{Obs}$).
4. The result ($\Theta_{||\Omega}$) is then synchronised with the Petri net model of the observation sequence ($\text{Obs}$) to obtain the net ($\Theta_{||\Omega}||\text{Obs}$).
5. The model checker of TINA is used to verify the matching property between the system and the pattern according to the input observation sequence on the net ($\Theta_{||\Omega}||\text{Obs}$).
6. The diagnosis result is thus given according to the answer of the model checker.

All the steps are detailed hereafter.

3.1 Timed languages modeling

The approach relies on the following class of Time Petri Nets (see Berthomieu et al. (2006) and Peterson (1977)).

**Definition 3.1. (LTLTPNp).** An L-type Labeled Time Petri Net with Priorities (LTLTPNp for short) is a tuple $\mathcal{N} = (P, T, A, \succ, t, \ell, \Sigma, I, S, Q, M_0)$ with:

- $P$: a finite set of nodes called places;
- $T$: a finite set of nodes called transitions and $P \cap T = \emptyset$;
- $A \subset (P \times T) \cup (T \times P)$ is the set of arcs between nodes.
- $\succ \subset T \times T$: the priority relation, not reflexive, not symmetrical and transitive ($t_1 \succ t_2$ means that if $t_1$ and $t_2$ are both enabled then only $t_1$ is firing).
- $\ell : P \cup T \rightarrow L \cup \Sigma \cup \{\lambda\}$: the deterministic labelling function where $L$ (resp. $\Sigma$) is the set of place labels (resp. transition labels). For $r = \delta_1 t_1 \delta_2 t_2 \cdots \in (\mathbb{N} \times T)^*$, $\ell(r) = \delta_1 \ell(t_1) \delta_2 \ell(t_2) \cdots$;
- $I_s : T \rightarrow I^+$: the static time interval function, where $I^+$ is the non empty set of time intervals with non negative rational bounds.
- $Q$: the set of final markings and $M_0$ the initial marking.

A marking $M$ is a function from $P$ to $\mathbb{N}$ which maps each place with a number of tokens. For an LTLTPNp $\mathcal{N}$, the functions $\text{pre}(t) = \{ p \in P : (p, t) \in A \}$ and $\text{post}(t) = \{ p \in P : (t, p) \in A \}$ are defined. A state of $\mathcal{N}$ is a couple $(M, I)$, with: $M$ a marking and $I$ the firing interval of the transition.

Priorities $(\succ)$ aim at managing transition conflicts and modify the classical transition firing rules. Transition $t$ is fireable from state $(M, I)$ after a duration $\delta$, denoted $(M, I) \rightarrow [\delta, I]$, if:

1. $M \geq \text{pre}(t)$;
2. $\delta \in I(t)$;
3. $\forall t' \notin t : M \geq \text{pre}(t') \implies \delta \leq \sup(I(t'))$;
4. $\forall t' \in T : t \cdot M \geq \text{pre}(t') \lor \delta \notin I(t')$.

Intuitively, $t$ is fireable after $\delta$ time units from state $(M, I)$, if (1) $t$ is enabled, (2) $\delta$ belongs to the time interval associated with $t$ and is not greater than the upper bound of any other enabled transition $t'$, (3) at time $\delta$, $t$ has the highest priority among the transitions that could be fired without the defined priorities. $\mathcal{R}(N, M_0, I_0)$ is the set of reachable states from $(M_0, I_0)$, with $M_0$ the initial marking and $I_0$ the restriction of the static time interval function to the enabled transitions at $M_0$. As a L-type net (see Peterson (1977)), an LTLTPNp $\mathcal{N}$ generates the timed language $L(\mathcal{N})$ that is the set of timed sequences issued from the transition sequences generated by the net and leading to a final marking $M \in \mathcal{Q}^+\cap L(\mathcal{N}, M_0, I_0)$.

**Definition 3.2.** (Timed system model). A timed system of timed language $L$ is an LTLTPNp $\Theta = (P_\Theta, T_\Theta, A_\Theta, \succ_\Theta, \ell_\Theta, L_\Theta, \Sigma_\Theta, I_\Theta, M_0, Q_\Theta, M_0^*)$ such that $Q_\Theta$ is the set of reachable markings and $L = L(\Theta)$.

Figure 5 presents an LTLTPNp model of a system used in the application section (see section 4).

3.2 Timed pattern diagnosis of timed workflows

The general problem of timed pattern diagnosis in timed system as defined above may lead to decidability issues (Popova-Zeugmann (2013)). To remain computable, we now restrict the problem to a subclass of timed systems that are timed workflows.

**Definition 3.4.** (Timed workflow). A timed workflow $\Theta$ is a timed system where any $e \in \Sigma_\Theta$ occurs at most once in a run of the workflow.

Such a timed system can be represented by an acyclic and safe Petri net. Also, to remain computable, a pattern $\Omega$ is considered safe and conflict-free and must describe a behavior that is not reduced to the elapsed time (i.e. the sequence $0\lambda$ is not part of $L(\Omega)$ which means $M_{0\lambda} \not\in L(\Omega)$. Finally, to keep the causality relation described by the underlying untimed Petri net, any transition not enabled by the initial marking has a static time interval with a strictly positive bound.

3.3 A model to capture the matching relation

The diagnosis function defined in Section 2.3 is based on the matching relation between the evolutions of the system that are consistent with the input observation sequence and the pattern. To capture this relation, $\Theta$ and $\Omega$ models are composed to obtain a new Petri net model denoted $\Theta_{||\Omega}$. This
model represents the synchronized evolutions of the system and the pattern but also the evolutions of the system alone as a pattern evolution corresponds potentially (in case of matching) to a sub-word of the system. The combination of both nets is obtained by performing the union of the system places and the pattern places. The transitions of the system that do not share any event with the pattern are kept with their initial time intervals. The other ones must be synchronized with the pattern’s transitions. As in Time Petri nets compositionality is kept only when synchronizations involve transitions with an interval \([0, +\infty]\) (Berthomieu et al. (2006)), we need to proceed to a time decomposition of the transitions to synchronize (see Figure 3). Transition \(t_1\) (on the left) is labeled by \(a\) with a finite time interval \([\alpha, \beta]\). \(t_1\) is replaced by a sub-net (on the right) composed by a silent transition (labeled by \(\lambda\) not indicated on the figure) of interval \([\alpha, \lambda]\) whose firing enables both a transition labeled by \(a\) (\(t_2\)) with the interval \([0, +\infty]\) and a silent transition (\(t_2\)) of interval \([\beta - \alpha + 1, \beta - \alpha + 1]\). At \(\beta + 1\) (i.e. \(\beta\) times unit + one time unit) as this transition has no priority, the silent transition (\(t_2\)) is fired leading to an unacceptable behavior represented by the marking of a specific place \(unAcc\). This transformation applies to transitions of the system as well as transitions of the pattern. Therefore in the system the specific place is named \(sys\_unAcc\) whereas in the pattern it is \(pat\_unAcc\).

3.4 Integration of the observation sequence

The observation sequence is first modeled by an L-Type Labeled Time Petri net denoted \(Obs\). For instance for an observation sequence \(1o_13o_2\), \(Obs\) has one transition \(t_1\) labeled \(o_1\) assigned with a static time interval \([1, 1]\). The output place of this transition enables a second transition \(t_2\) labeled by \(o_2\) with an interval \([3, 3]\). The output place of \(t_2\) is denoted \(p_{Obs}\). Afterwards, the combination system-pattern (i.e the net \(\Theta_{Ω}\)) is synchronized (Hack (1975)) with the Petri model of the observation sequence (\(Obs\)) to obtain the net \(\Theta_{Ω}||Obs\). To perform the synchronization the transitions of the observation sequence net (\(Obs\)) and the transitions of the combination (\(\Theta_{Ω}\)) are transformed so that the synchronizations involve only transitions with \([0, +\infty]\) intervals as explained in Section 3.3. The verification of the matching property between the system and the pattern according to the input observation sequence on \(\Theta_{Ω}||Obs\) is then performed by the model checker. Then the diagnosis function (\(R\)-diagnosis) generates its results according to the answers of the model checker as presented now.

3.5 Implementation of the \(R\)-diagnoser function by Model checking

Back to the description of the method in Section 3, the overall computational complexity of the method lies in step 5 and we propose to use an off-the-shelf model checking tool to solve this part: the TINA toolkit (Berthomieu et al. (2004)). TINA generates enriched labeled Kripke structures from which it is possible to check properties written in SE-LTL (State/Event Linear Temporal Logic) which extends LTL in the way presented hereafter. A formula \(ψ\) is a SE-LTL formula if it is a universally quantified formula : \(ψ := \forall \varphi\) such that:

\[1\] This property is true because the systems that are considered here are workflows (see Definition 3.4).
The proposed method is illustrated on an example denoted as the foodshop example. This example is inspired from the Web services’ systems that were studied in the European project WSDIAMOND (Di Nitto et al., 2009). Figure 5 presents the workflow of a foodshop service. In this type of application, fault patterns usually represent misbehaviors not compatible with the service specification instead of physical faults on hardware. In this workflow, a foodshop first receives the order of a client (sh_req_whcheck). Within the next 3 hours, the shop must start two activities (sh_st[1,2]) in parallel. The first activity dispatches the order into two categories: the storable products (not perishable) and the only accepting marking \( M(\Omega) \notin Q_0 \). The foodshop then checks (wh_check) and provides a feedback (available wh_send_av, unavailable wh_send_unav). For non-storable products, the shop sends directly a request to a supplier (sh_reg_suppcheck) that also provides feedbacks (sp_send_av, sp_send_unav). The shop requires that the feedbacks are given within the next 4 hours. After the reception of the feedbacks, the shop decides to implement or not the order based on the availabilities (sh_decide). If one product is missing the shop cancels the order (sh_cancel), if not the shop requests the preparation for delivery to the warehouse and the supplier (sh_reg_deliv). When preparations are finished (wh_end_prep_deliv and sp_end_prep_deliv), it is finally delivered. In Figure 5 (on the left), observable events are in bold. They are events from the shop that manages the whole delivery process. To illustrate the method, we consider two patterns.

\[
\varphi := cst \mid r \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \circ \varphi \mid \square \varphi \mid \Diamond \varphi \mid \varphi U \varphi
\]

with \( p \) a place symbol, \( a \) a transition symbol, \( r \) a proposition, \( c \in \mathbb{N} \), \( \Delta \in \{<, >, \leq, \geq\} \) and \( \varnothing \in \{+,-,\ast,/\} \). The operators \( \circ \) (next), \( \square \) (always), \( \Diamond \) (eventually) and \( U \) (until) have their usual LTL semantics.

The \( R \)-diagnoser is based on two questions written in SE-LTL concerning the Petri net model \( \Theta_{\Omega} \). (Obs):

(1) Question \( \varphi_{\text{CERTAIN}} \): Are we sure that the pattern is recognized in each system’s behavior that produces the observations? If the result is false the following question is asked to the model checker:

(2) Question \( \varphi_{\text{ABSENT}} \): Are we sure that the pattern is never recognized in each system’s behavior that produces the observations?

More formally, each question corresponds to a SE-LTL formula. Given \( M \) a marking of \( \text{Obs} \mid \Theta_{\Omega} \), \( M(\Omega) \) denotes the restriction of \( M \) to \( \Omega \) i.e. the marking of the pattern’s places.

(1) \( \varphi_{\text{CERTAIN}} \equiv \Box (\text{sys_unAcc} = 0 \land \text{pob} = 1) \Rightarrow M(\Omega) \in Q_0 \), in other words, it is always true (\( \Box \)) that if the system generates \( \text{Obs} \) (\( \text{sys_unAcc} = 0 \land \text{pob} = 1 \)) then (\( \Rightarrow \)) it matches the pattern (\( M(\Omega) \in Q_0 \)).

(2) \( \varphi_{\text{ABSENT}} \equiv \Box (\text{sys_unAcc} = 0 \land \text{pob} = 1) \Rightarrow M(\Omega) \notin Q_0 \), in other words, it is always true (\( \Box \)) that if the system generates \( \text{Obs} \) \( \text{sys_unAcc} = 0 \land \text{pob} = 1 \) then (\( \Rightarrow \)) it does not match the pattern (\( M(\Omega) \notin Q_0 \)).

4. APPLICATION TO THE DIAGNOSIS OF A TIMED WORKFLOW OF A FOODSHOP.

The proposed method is illustrated on an example denoted as the foodshop example. This example is inspired from the Web services’ systems that were studied in the European project WSDIAMOND (Di Nitto et al., 2009). Figure 5 presents the workflow of a foodshop service. In this type of application, fault patterns usually represent misbehaviors not compatible with the service specification instead of physical faults on hardware. In this workflow, a foodshop first receives the order of a client (sh_req_whcheck). Within the next 3 hours, the shop must start two activities (sh_st[1,2]) in parallel. The first activity dispatches the order into two categories: the storable products (not perishable) and the only accepting marking \( M(\Omega) \notin Q_0 \).

TINA’s response is negative: there exists at least one run of the system that produces \( \sigma_1 \) and for which \( \text{sp}_\text{send}_\text{unav} \) has not happened. Then we check whether these runs do not match (i.e. contain) \( \text{sp}_\text{send}_\text{unav} \) with the following query \( \varphi_{\text{ABSENT}} \):

\[
\emptyset ((\text{sys}_\text{unAcc} = 0 \land p9 = 1) \Rightarrow (pp1 = 1))
\]

This is a positive answer. So, the diagnosis of \( \sigma_1 \) is that \( \text{sp}_\text{send}_\text{unav} \) did not happen. Let us consider now the observable sequence \( \sigma_2 \) composed of the first five events of \( \sigma_1 \) followed by \( 8 \text{sh} \text{decide} 2 \text{sh} \text{cancel} \). In this case \( \varphi_{\text{CERTAIN}} \) is true: any run generating \( \sigma_2 \) contains \( \text{sp}_\text{send}_\text{unav} \). Querying TINA with \( \varphi_{\text{ABSENT}} \) is useless: the diagnosis is certain.

4.2 Occurrence of a sequence pattern

Consider \( \text{wh}_\text{send}_\text{av}[0,\infty[ \rightarrow \text{wh}_\text{end}_\text{prep}_\text{deliv}[70,200] \) .

This second pattern illustrates a misbehaviour: if the warehouse sends as a feedback that the goods are available then the preparation ends within 70 and 200 time units which is considered as an unacceptable duration (quality of services is impacted). Let us consider the following observed sequence: \( \sigma_3 = 50 \text{sh} \text{recv} \text{g} 1 \text{sh} \text{st} 1 \text{sh} \text{end} \text{spl} 1 \text{sh} \text{reg}_\text{whcheck} 1 \text{sh} \text{reg}_\text{suppcheck} 3 \text{sh} \text{decide} 2 \text{sh} \text{reg}_\text{deliv} 50 \text{deliv} \). TINA concludes that the pattern has not happened. This is due to the fact that the delivery date is time 50 after the order request. Now if we modify \( \sigma_3 \) by increasing the date 50 to 80, the diagnosis becomes certain. Only the observation of time is discriminating here.

5. CONCLUSIONS ET PERSPECTIVES

This work extends the original problem of the diagnosis of simple faults to the diagnosis of timed patterns in timed systems. The diagnoser function is based on the notion of matching between the system evolutions that are consistent with an input sequence of observations and the pattern. A resolution of the problem based model checking techniques is proposed to solve the problem on timed workflows. This work must be completed by a diagnosability analysis inspired from Gougam et al. (2017). We are also planning to investigate other subclasses of Time Petri nets where the problem is decidable.

REFERENCES