Diagnostic de motifs de comportements dans les systèmes temporels

Yannick Pencolé and Audine Subias

CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Université de Toulouse, INSA, LAAS, F-31400 Toulouse, France
pencole@laas.fr,subias@laas.fr

Résumé

Dans cet article nous proposons une formulation du problème de diagnostic de motifs d'un système temporel. Il s'agit de rechercher à partir d'une séquence d'observations du système, toutes les évolutions de celui-ci qui non seulement produisent la séquence observée mais aussi qui sont concordantes avec les motifs d'intérêt considérés. La caractérisation formelle du problème de diagnostic s'appuie sur la notion de *pattern matching* (concordance de motifs). Posé sous forme d'un problème d'atteignabilité, le problème de diagnostic est résolu par *model checking*.

1 Introduction

Le problème du diagnostic dans les systèmes à événements discrets a été à l'origine défini en s'appuyant sur un modèle du système et un ensemble explicite d'événements de faute [25]. Ainsi posé ce problème a été résolu par l'utilisation de différents formalismes comme les automates, les automates communicants et plus récemment par l'utilisation de réseaux de Petri [3],[13],[19] [11],[8],[2],[7],[29],[31].

Les motifs de fautes permettent d'étendre la notion d'événements de faute en introduisant des comportements plus complexes [16]. Un système peut en effet avoir un comportement de faute en raison d'une séquence d'événements spécifique i.e. un ordre particulier d'événements se produisant sur le système. N'importe quel événement de la séquence peut, indépendamment des autres ne pas être un événement de faute, mais un agencement particulier de ces événements est la cause d'un comportement de faute. Au delà des comportements de faute, les motifs peuvent modéliser également n'importe quel comportement normal d'intérêt ou comportement associé à des spécifications sur le système [17]. Les motifs de comportement permettent en ce sens d'appréhender un large spectre de problèmes de diagnostic : faute simple, faute multiple, fautes répétitives ...et plus globalement tout problème de diagnostic d'un comportement d'intérêt (normal ou de faute). Un autre avantage de l'utilisation des motifs de comportements réside dans la séparation claire du modèle des comportements du système et des objectifs de diagnostic puisque ceux-ci sont définis par le motif lui-même [30]. La réutilisabilité des motifs est ainsi renforcée pour aborder de nouveaux problèmes de diagnostic sur d'autres systèmes. Le pouvoir descriptif des motifs de comportements peut aussi être renforcé par l'intégration de la dimension temporelle exprimée notamment sous formes de contraintes entre les dates d'occurrence des événements mis en jeu dans le motif. Le problème de diagnostic peut alors être étendu aux systèmes temporels [12],[26],[14],[2],[7],[24],[20],[29].

Dans cet article nous proposons une approche de diagnostic de motifs de comportements temporels. Le système et les motifs sont modélisés par des langages temporels. L'approche proposée prend ses fondements dans le domaine du *pattern matching* (concordance de motifs). Initialement posé comme la recherche dans une séquence d'entrée d'une sous-chaîne spécifique (*string matching problem* ou encore *single keyword pattern problem*)[18], le problème de *pattern*

matching a évolué vers la recherche d'un ensemble de sous-chaînes spécifiques (*multiple keyword pattern matching problem*), pour ensuite être généralisé à la recherche de toutes les sous-chaînes qui satisfont un motif plus complexe décrit par une expression régulière [1]. Des extensions au contexte temporel (*timed pattern matching*) ont également été développées sur la base de motifs spécifiés par des expressions régulières temporelles ou des automates temporisés [27],[28].

Les travaux présentés dans cet article portent sur une formulation du problème de diagnostic de motifs d'un système temporel. Il s'agit de rechercher à partir d'une séquence d'observations du système, toutes les évolutions de celui-ci qui non seulement produisent la séquence observée mais aussi qui sont concordantes avec les motifs d'intérêt considérés. Le positionnement du problème de diagnostic et les notations sont introduits dans la section 2 qui se conclue par une caractérisation formelle du problème de diagnostic. La section 3 propose une résolution par *model checking* du problème de diagnostic posé sous forme d'un problème d'atteignabilité. Enfin un exemple d'application est donné en section 4.

2 Positionnement du problème

L'objectif de l'approche est de diagnostiquer l'occurrence d'un motif de comportement temporel lors de l'évolution d'un système. Il s'agit donc pour un motif donné de mettre en place une fonction de diagnostic qui prend en entrée une séquence d'observation temporelle et qui en retour renvoie dans un délai borné un des trois résultats suivant :

1. *Le comportement décrit par le motif s'est produit pendant l'évolution du système.*
2. *Le comportement décrit par le motif ne s'est pas produit pendant l'évolution du système .*
3. *Il est possible que le comportement décrit par le motif se soit produit pendant l'évolution du système.*

Cette vision intuitive de la fonction de diagnostic amène à se poser plusieurs questions : comment le système est-il modélisé ? qu'est-ce qu'un motif de comportement temporel et comment est-il modélisé ? qu'est-ce qu'un délai ? Après une présentation rapide des notations de base, les sections suivantes répondent à ces différentes questions.

2.1 Notations

Soit Σ un alphabet, c.-à-d. un ensemble de symboles (généralement des lettres) la clôture de Kleene de Σ, notée $\Sigma^\star$, est définie comme étant l'ensemble des séquences finies issues de Σ (la séquence vide ε incluse). Σ^+ est l'ensemble des séquences finies et non vides issues de Σ($\Sigma^+ = \Sigma^\star \backslash \{\varepsilon\}$). Un sous-ensemble $\mathcal{S}$ de $\Sigma^\star$ est appelé un *langage* (atemporel) issu de Σ. La continuation sur $\mathcal{S}$ d'une séquence ρ — notée $\mathcal{S}/\rho$ — est l'ensemble de séquences : $\{\zeta \in \Sigma^\star : \rho\zeta \in \mathcal{S}\}$. Les langages temporels étendent cette notion de langage en y adjoignant une information temporelle. On distingue un symbole particulier — $\lambda \notin \Sigma$ — qui représente un point temporel. Une séquence temporelle sur Σ est un enchainement de délais temporels et d'événements de Σ, plus précisément une séquence temporelle sur Σ est un élément de $\mathcal{T}(\Sigma) = (\mathbb{R}^+ \times (\Sigma \cup \lambda))^+$. Considérons par exemple l'alphabet $\Sigma = \{a, b, c\}$, $\rho = 3a1\lambda 1c1c2b5\lambda$ est une séquence temporelle, $\rho' = 1b1b40\lambda$ en est une autre. La séquence *vide* est la séquence 0λ. Pour toute séquence temporelle, on peut définir une représentation canonique de la forme $\delta_1 e_1 .. \delta_n e_n \delta_{n+1}\lambda, e_i \in \Sigma$ de la séquence en supprimant les points temporels inutiles, par exemple la forme canonique de $\rho = 3a1\lambda 1c1c2b5\lambda$ est $3a2c1c2b5\lambda$. Par la suite et sans perte de généralité, nous ne considérerons que des formes canoniques. La projection d'une séquence ρ sur un sous-ensemble Σ_p de Σ notée $P^t_{\Sigma_p}(\rho)$ est la séquence temporelle obtenue après l'effacement des éléments n'appartenant pas à

Σ_p et sommation des délais successifs. Comme la forme générale d'une séquence temporelle est une succession de délais et d'événements, l'effacement d'événements résultera par l'apparition de suite de délai sans événements entre eux. Pour revenir à une forme de séquence temporelle, ces délais successifs sont sommés. Reprenons la séquence canonique de ρ, elle correspond à un événement a qui se produit 3 unités de temps après l'origine, à un événement c qui se produit 2 unités de temps après a (soit à la date 5), suivi à 1 unité de temps par un autre événement c, suivi par l'occurrence d' un événement b, 2 unités de temps après et s'arrête ensuite après 5 unités de temps. Prenons $\Sigma_p = \{a, b\}$ alors $P^t_{\Sigma_p}(\rho) = 3a5b5\lambda$. On appelle langage temporel sur Σ tout sous-ensemble (canonique) de $\mathcal{T}(\Sigma)$. On note durée(ρ) la somme de tous les délais temporels de ρ, par exemple si $\rho = 3a4c1c2b4\lambda$, durée(ρ) = 3+4+1+2+4 = 14.

2.2 Concordance d'un système et d'un motif temporels

Comme un système à événements discrets atemporel, un système temporel génère un ensemble fini d'événements constituant son alphabet Σ. La différence est que dans le cas temporel, les délais temporels possibles entre l'occurrence de deux événements successifs sont connus, alors que dans le cas atemporel, les événements ne sont reliés que par une relation de précédence. Notons évidemment, que dans les deux cas (temporel et atemporel), deux événements peuvent ne pas être reliés, c.-à-d. que même l'ordre de leur occurrence n'est pas connu. Une évolution d'un système à événements discrets temporel est une *séquence temporelle* sur son alphabet. L'ensemble des évolutions possibles d'un système temporel est donc un ensemble de séquences temporelles c'est à dire un langage temporel. Deux séquences temporelles peuvent être reliées par la notion de concordance (*matching*). Cette notion permet de déterminer si une séquence est présente dans une autre [18].

Définition 2.1 (concordance de séquence (*matching*))**.** *Une séquence $\rho \in \mathcal{T}(\Sigma)$ est en concordance avec une autre séquence $\omega \in \mathcal{T}(\Sigma)$, noté $\rho \Subset \omega$, si :*
— *$\omega = \delta\lambda$, avec : $\delta \in \mathbb{R}^+$, et* durée(ρ) $\geq \delta$ *; ou*
— *$\omega = \delta_0\sigma_0\omega_1$, telle que : $\delta_0 \in \mathbb{R}^+$, $\sigma_0 \in \Sigma$, $\omega_1 \in \mathcal{T}$ et il existe deux séquences $\rho_0 \in \mathcal{T}(\Sigma), \rho_1 \in \mathcal{T}(\Sigma)$ et une durée $\delta \in \mathbb{R}^+$ telles que :*

1. *$\rho = \rho_0\delta\sigma_0\rho_1$;*
2. *$\delta_0 =$* durée(ρ_0) $+ \delta$ *;*
3. *$\rho_1 \Subset \omega_1$.*

La figure 1 représente schématiquement la concordance $\rho = 2b3c2a5a1b1c4a2c0\lambda$ avec $\omega = 7a7c6c0\lambda$. La vérification de $\rho \Subset \omega$ se fait par récursivité. Lors de la première récursion, ω est subdivisée en trois sous-séquences $\omega = \delta_0\sigma_0\omega_1$, avec : $\delta_0 = 7, \sigma_0 = a$ et $\omega_1 = 7c6c$. De même, ρ est subdivisée en quatre sous-séquences, $\rho = \rho_0\delta\sigma_0\rho_1$, avec : $\rho_0 = 2b3c, \delta = 2, \sigma_0 = a$ et $\rho_1 = 5a1b1c4a2c$. La subdivision de ρ doit vérifier durée(ρ_0) $+ \delta = \delta_0$, ce qui est bien le cas ici $(2+3)+2 = 7$. Dans la deuxième récursion, le même procédé est utilisé en prenant comme base $\rho = 5a1b1c4a2c0\lambda$ et $\omega = 7c6c0\lambda$. Enfin, lors de la dernière récursion, nous avons les subdivisions de ce qui reste de ρ et ω. Avec : $\rho_0 = 4a, \delta = 2, \sigma_0 = c, \delta_0 = 6$ et $\rho_1 = \omega_1 = 0\lambda$. Cette division vérifie bien durée(ρ_0) $+ \delta = \delta_0$ $(4+2 = 6)$. De plus, comme $\rho_1 = \omega_1 = 0\lambda$, par définition $\rho_1 \Subset \omega_1$ et donc $\rho \Subset \omega$. À noter que le deuxième événement a dans la séquence ρ ne peut pas être utilisé comme base pour la première récursion. En effet, si nous subdivisons ρ en quatre séquences $\rho = \rho_0\delta\sigma_0\rho_1$, où $\rho_0 = 2b3c2a, \delta = 5, \sigma_0 = a$ et $\rho_1 = 1b1c4a2c0\lambda$, la deuxième condition de la définition de la concordance ne sera plus satisfaite (durée(ρ_0) $+ \delta = (2+3+2)+5 = 12 \neq \delta_0$).

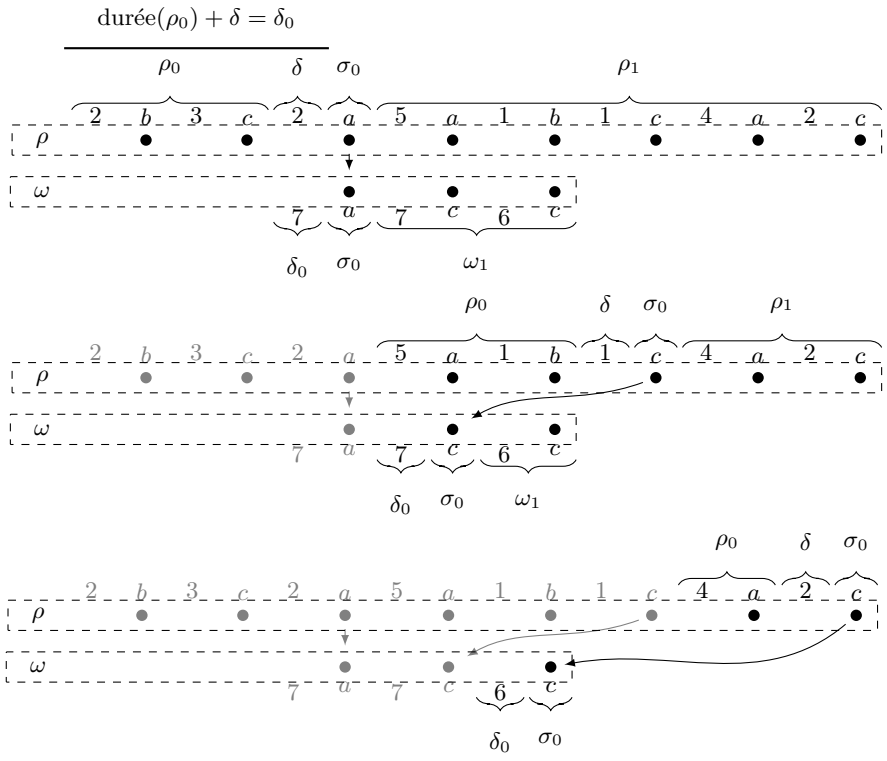

FIGURE 1 – Illustration de la notion de concordance (*matching*).

Définition 2.2 (Système temporel)**.** *Un système à événements discrets temporel, dont l'ensemble des événements est* Σ*, est un langage temporel* $\mathcal{L} \subseteq \mathcal{T}(\Sigma)$ *tel que si* $\rho = \rho_1\rho_2$ *et* $\rho \in \mathcal{L}$*, avec* $\rho_1, \rho_2 \in \mathcal{T}(\Sigma)$*, alors* $\rho_1 \in \mathcal{L}$*.*

L'ensemble Σ des événements générés par le système est partitionné en deux sous-ensembles : Σ_o et Σ_u représentant respectivement l'ensemble des événements observables et l'ensemble des événements non observables. On définit un *motif temporel* comme un ensemble $\mathcal{R}$ de séquences temporelles — c.-à-d. un langage temporel — qui définissent un comportement auquel on s'intéresse. Un tel motif est un comportement du système analysé jugé intéréssant à diagnostiquer. Il peut s'agir par exemple de l'occurrence d'un événement simple dans un délai temporel donné, de l'occurrence de plusieurs événements qui se produisent dans un ordre quelconque mais dont les dates d'occurrence respectent des contraintes temporelles données, ..., ou bien encore de l'occurrence de plusieurs événements qui se produisent selon un ordre et des contraintes temporelles fixés.

Définition 2.3 (Motif temporel)**.** *Un motif sur un ensemble d'événements* $\Sigma_{\mathcal{R}}$ *est un langage temporel* $\mathcal{R} \subseteq \mathcal{T}(\Sigma_{\mathcal{R}})$ *tel que si* $\rho \in \mathcal{R}$ *alors seules des continuations de* ρ *de la forme* $\rho\delta\lambda, \delta \in \mathbb{R}^+$ *sont dans* $\mathcal{R}$*.*

Dans l'approche présentée ici, le langage temporel d'un motif de comportement ne modélise que le comportement désiré et ne prend pas en compte tous les événements possibles du système. Par exemple, si le motif considéré est *l'événement a arrive après b* il sera défini par le langage temporel $\{\delta_0 b \delta_1 a \delta_2 \lambda, \delta_0, \delta_1, \delta_2 \in \mathbb{R}^+, \delta_1 > 0\}$, qui ne prend pas en compte tous les

événements de Σ pour un système donné. Cette représentation est en intension par opposition à la représentation en extension proposée par [16] pour des motifs atemporels et qui dépend d'un ensemble d'événements Σ donné.

Définition 2.4 ($\mathcal{R}$-concordance)**.** *Soit $\mathcal{R}$ un motif de comportement. Une séquence $\rho \in \mathcal{T}(\Sigma)$ est $\mathcal{R}$-concordante s'il existe une séquence $\omega \in \mathcal{R}$ tel que $\rho \ni \omega$, c.-à-d. ρ matche ω.*

2.3 Diagnostic de motifs temporels

Le diagnostic temporel de fautes simples s'appuyant sur un modèle complet (automates temporels) des comportements du système a été introduit dans [26]. Nous proposons ici d'étendre la définition pour considérer des comportements plus complexes que les fautes simples, comportements décrits par des motifs temporels. Pour un motif de comportement donné $\mathcal{R}$, le problème de diagnostic consiste à définir une fonction $\mathcal{R}$-diagnostiqueur qui prend en entrée une séquence temporelle d'événements observables et qui produit l'un des trois symboles $\{\mathcal{R}-certain, \mathcal{R}-sain, \mathcal{R}-ambigu\}$[22].

Définition 2.5 ($\mathcal{R}$-diagnostiqueur)**.** *Soit $\mathcal{L}$ le langage temporel du système sur l'ensemble d'événements $\Sigma = \Sigma_u \cup \Sigma_o$, un $\mathcal{R}$-diagnostiqueur est une fonction*

$$\Delta_{\mathcal{R}} : \mathcal{T}(\Sigma_o) \rightarrow \{\mathcal{R}-certain, \mathcal{R}-sain, \mathcal{R}-ambigu\}$$

telle que :
- *$\Delta_{\mathcal{R}}(\sigma) = \mathcal{R}-certain$ si pour toute séquence $\rho \in \mathcal{L}$ qui est cohérente avec σ (i.e.$P^t_{\Sigma_p}(\rho) = \sigma$), ρ est $\mathcal{R}$-concordante ;*
- *$\Delta_{\mathcal{R}}(\sigma) = \mathcal{R}-sain$ si pour toute séquence $\rho \in \mathcal{L}$ qui est cohérente avec σ, ρ n'est pas $\mathcal{R}$-concordante ;*
- *$\Delta_{\mathcal{R}}(\sigma) = \mathcal{R}-ambigu$ dans tout autre cas.*

Pour un ensemble de motifs de comportements $\mathcal{R}_1, \ldots, \mathcal{R}_n$, la fonction diagnostic d'un système peut être define par : $\Delta : \mathcal{T}(\Sigma_o) \rightarrow \prod_{i=1}^{n}\{\mathcal{R}_i-certain, \mathcal{R}_i-sain, \mathcal{R}_i-ambigu\}$ tel que $\Delta(\sigma) = (\Delta_{\mathcal{R}_1}(\sigma), \ldots, \Delta_{\mathcal{R}_n}(\sigma))$.

3 Mise en œuvre du problème à l'aide d'un outil de Model-Checking

Dans cet article nous proposons de poser le problème de diagnostic de motif temporel comme un problème d'atteignabilité et d'utiliser une méthode de vérification de modèle (*model checking*) pour le résoudre. Cette résolution va s'appuyer sur l'outil TINA (TIme petri Net Analyzer) [5]. Cet outil analyse des réseaux de Petri temporels et vérifie des propriétés représentées par des formules de la logique SE-LTL (*State/Event Linear Temporal Logic*). Le principe est donc dans un premier temps de modéliser les langages temporels du système et du motif de comportement par des réseaux de Petri temporels et dans un deuxième temps de les combiner afin de mettre en œuvre la relation de concordance entre système et motif. Dans un troisième temps, le résultat de cette combinaison sera lui-même synchronisé avec un réseau de Petri temporel représentant une séquence d'oservations donnée. Les différentes opérations qui vont être définies reposent sur une extension des réseaux de Petri temporels : les *réseaux de Petri temporels étiquetés avec priorités et marquages accepteurs* (RdPTÉPrA).

3.1 Modélisation des langages temporels

L'approche proposée repose sur le formalisme des réseaux de Petri temporels étiquetés avec priorités et marquages accepteurs (RdPTÉPrA) dont la définition est donnée ci-après [21].

Un RdPTÉPrA est un 9-uplet $\Theta = \langle P, T, A, \succ, \ell, L, \Sigma, I_s, Q, M_0 \rangle$ [4], avec :

— P : un ensemble de places ;
— T : un ensemble de transitions, avec $P \cap T = \varnothing$;
— $A \in (P \times T) \cup (T \times P)$: une relation représentant les arcs entre les nœuds ;
— $\succ \subset T \times T$: la relation binaire de priorité, non réflexive, non symétrique et transitive [6] ;
— $\ell : P \cup T \to L \cup \Sigma \cup \{\lambda\}$: l'application d'étiquetage, où L (resp. Σ) est l'ensemble des étiquettes de places (resp. de transitions) ; pour $r = \delta_1 t_1 \delta_2 t_2 \cdots \in (\mathbb{R} \times T)^\star$, $\ell(r) = \delta_1 \ell(t_1) \delta_2 \ell(t_2) \ldots$
— $I_s : T \to I^+$: l'application intervalle statique, où I^+ est l'ensemble des intervalles temporels non vides (la borne inférieure est inférieure ou égale à la borne supérieure) avec des bornes rationnelles non négatives ;
— Q : les marquages accepteurs et M_0 : le marquage initial.

Un marquage M est une application de P vers $\mathbb{N}$ qui associe à chaque place un nombre de jetons. Pour un RdPTÉPrA Θ, on définit les fonctions $\text{pre}_\Theta(t) = \{p \in P : (p,t) \in A\}$ et $\text{post}_\Theta(t) = \{p \in P : (t,p) \in A\}$. Un état du RdPTÉPrA est donné par $\langle M, I \rangle$, où : M est un marquage et I est l'intervalle de franchissement de la transition. Une transition t est franchissable à partir de $\langle M, I \rangle$ après un délai δ, noté $\langle M, I \rangle \to [\delta t]$, si :

1. $\text{pre}_\Theta(t) \subseteq M$;
2. $\delta \in I(t)$;
3. $\forall t' \neq t : \text{pre}_\Theta(t') \subseteq M \implies \delta \leq \sup(I(t'))$;
4. $\forall t' \in T_\Theta \succ t : \text{pre}_\Theta(t') \nsubseteq M \vee \delta \notin I(t')$.

Dans un RdPTÉPrA, une transition est donc franchissable après δ unités de temps si toutes les places d'entrée de cette transition sont marquées, le délai δ appartient à l'intervalle temporel de franchissement de la transition, aucune autre transition vérifiant ces conditions n'est arrivée à la limite supérieure de son intervalle de franchissement et enfin aucune transition prioritaire n'est franchissable. L'ensemble des transitions instantanément franchissables à partir d'un état $\langle M, I \rangle$ est noté franchissable(Θ, M, I). On note $\text{R}(\Theta, M_0, I_0)$ l'ensemble des états accessibles d'un RdPTÉPrA Θ à partir de l'état $\langle M_0, I_0 \rangle$, où : M_0 est le marquage initial et I_0 est la restriction de la fonction intervalle statique (I_s) aux transitions sensibilisées par le marquage initial $(\text{post}(M_0))$. Un RdPTÉPrA Θ génère un langage temporel $\mathcal{L}(\Theta)$ contenant l'ensemble des séquences temporelles résultant des séquences de transitions franchissables menant à un marquage accepteur $M \in Q$.

Définition 3.1 (Modèle du système temporel)**.** *Le système temporel de langage $\mathcal{L}$ est un réseau de Petri $\Theta = \langle P, T, A, \succ, \ell, L, \Sigma, I_s, Q, M_0 \rangle$ tel que $\mathcal{L} = \mathcal{L}(\Theta)$.*

Il existe plusieurs réseaux pouvant représenter le même langage. Ici, nous nous limiterons aux réseaux pour lesquels :

1. le réseau est sauf (pas plus de un jeton par place dans tout marquage accessible) ;
2. la borne inférieure de chaque intervalle temporel statique est strictement positive (préservation de la causalité entre transitions) ;
3. l'ensemble des marquages accepteurs est l'ensemble des marquages accessibles.

La figure 4 présente le modèle d'un tel système (voir exemple d'application en section 4).

Le langage du motif de comportement est également modélisé par un RdPTÉPrA noté $\Omega = \langle P_\Omega, T_\Omega, A_\Omega, \succ_\Omega, \ell_\Omega, L_\Omega, \Sigma_\Omega, I_{s\Omega}, Q_\Omega \rangle$ avec Q_Ω l'ensemble des marquages accepteurs. Dans l'approche proposée, ce modèle doit vérifier les conditions suivantes :

1. $M_0 \notin Q_\Omega$, le motif ne contient pas la séquence 0λ ;
2. tout événement du motif n'est pas observable dans le système considéré ;
3. le réseau de Petri Ω est sauf et le réseau de Petri Ω ne contient pas de cycles ;
4. le réseau n'a pas de conflit structurel et à tout instant, si une transition t d'étiquette $\ell_\Omega(t) = e$ est franchissable alors aucune autre transition de même étiquette e ne l'est ;
5. toute transition non sensibilisée par le marquage initial M_0 a un intervalle statique temporel dont la borne inférieure est strictement positive ;
6. tout marquage accepteur est bloquant (voir la définition d'un motif temporel).

3.2 Formulation du problème de diagnostic par model checking

3.2.1 Model checking

L'outil *TINA* permet de générer à partir d'un réseau de Petri la structure de Kripke [9] sur laquelle repose le problème de model checking. En réalité, TINA construit une structure de Kripke enrichie sur laquelle peut être vérifiée une formule ψ de la logique SE-LTL (*State/Event Linear Temporal Logic*) qui est une extension de la logique Logique Temporelle Linéaire (LTL). Pour plus de détails nous invitons le lecteur à consulter `http://www.laas.fr/tina/`. La formule logique $\psi ::= \forall\varphi$ est telle que :

$$\begin{array}{rcl} \varphi & ::= & cst|r|\neg\varphi|\varphi \vee \varphi|\varphi \wedge \varphi| \bigcirc \varphi|\Box\varphi|\Diamond\varphi|\varphi U \varphi \\ r & ::= & e|e \triangle e \\ e & ::= & p|a|c|e \bigtriangledown e \\ cst & ::= & dead|div|sub \end{array}$$

avec p le symbole d'une place, a le symbole d'une transition, $c \in \mathbb{N}$, $\triangle \in \{=, <, >, \leq, \geq\}$ et$\bigtriangledown \in \{+, -, *, /\}$. Les opérateurs $\bigcirc$(next), $\Box$(always), $\Diamond$(eventually) et U(until) ont la sémantique usuelle de la logique *LTL* (Logique Linéaire Temporelle).

L'approche proposée dans la suite de cet article fait l'hypothèse d'un temps discret. Ainsi la structure de Kripke générée par TINA prend en compte cette hypothèse qui revient à ne considérer que les états *essentiels* obtenus par le franchissement des transitions pour des valeurs entières des horloges locales [23].

3.2.2 Représentation de la concordance entre un système et un motif

La fonction de diagnostic définie section 2.3 s'appuie sur la notion de concordance des séquences d'évolution du système (qui sont cohérentes avec une séquence d'observation) avec un motif de comportement. La concordance entre un système et un motif peut être mise en évidence par la construction d'un modèle réseau de Petri représentant les évolutions du système seul et les évolutions synchronisées du système et du motif. Ce modèle peut être obtenu en réalisant une *combinaison asymétrique* ($\Theta \ltimes \Omega$) entre le réseau modélisant le système Θ et celui associé au motif de comportement Ω. La combinaison asymétrique nécessite de synchroniser des transitions du système et des transitions du motif or ce n'est possible du point de vue temporel que si les transitions en question sont associées à des intervalles du type $[0, +\infty[$. Avant de procéder à une quelconque opération de synchronisation, il faut donc au préalable transformer les réseaux en effectuant une décomposition temporelle qui nécessite l'usage de priorité (voir la figure 2). Pour chaque transition du réseau initial impliquée dans une synchronisation, on

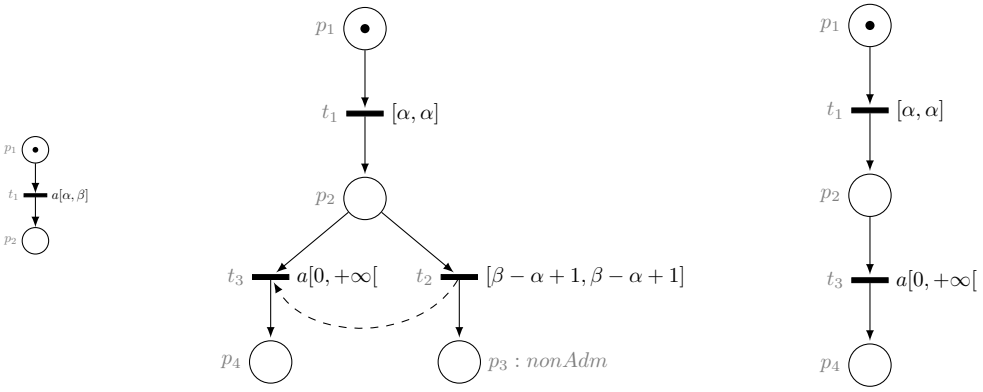

FIGURE 2 – Décomposition des contraintes temporelles $[\alpha, \beta]$ associées à une transition étiquetée par un événement a : à gauche le réseau initial, au milieu la décomposition et à droite la décomposition dans le cas où $\alpha > 0$ et β est infini.

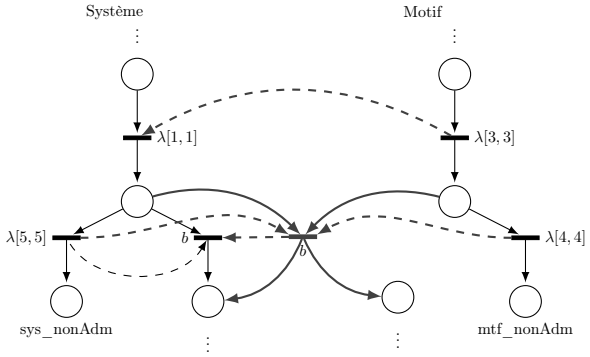

FIGURE 3 – Illustration de la combinaison asymétrique d'une transition $b[1, 4]$ d'un système avec une transition $b[3, 6]$ d'un motif.

procède comme suit. Si la transition d'étiquette a est associée à un intervalle de temps $[\alpha, \beta]$, on va la remplacer par un sous-réseau constitué d'une transition silencieuse d'intervalle $[\alpha,\alpha]$ dont le tir sensiblisera une transition d'étiquette a (entre les instants α et β) à laquelle on aura associé un intervalle $[0, +\infty[$. A $\beta + 1$ (i.e β unités de temps plus 1 unité de temps) cette transition n'étant pas prioritaire (si β est fini seulement) la transition silencieuse d'intervalle $[\beta - \alpha + 1,\beta - \alpha + 1]$ est franchie traduisant un comportement non admissible représenté dans chaque réseau décomposé par une place spéciale *nonAdm*. Ainsi dans le réseau du système décomposé, cette place aura pour nom *sys_nonAdm* et dans le modèle du motif décomposé, cette place sera nommé *mtf_nonAdm*.

Après l'étape de décomposition temporelle appliquée sur le système et le motif, la combinaison asymétrique est construite en réalisant l'union des places du système et de celles du motif, les transitions du système sont conservées à l'identique et celles du motif qui sont labellisées par des événements sont remplacées par des transitions synchronisées prioritaires avec un intervalle temporel $[0, +\infty[$ (voir figure 3).

Définition 3.2 (Combinaison asymétrique). *Soit* $\Theta_1 = \langle P_1, T_1, A_1, \succ_1, \ell_1, L_1, \Sigma_1, I_{s1}, Q_1\rangle, \Theta_2 = \langle P_2, T_2, A_2, \succ_2, \ell_2, L_2, \Sigma_2, I_{s2}, Q_2\rangle$ *tels que* $P_1 \cap P_2 = \varnothing$, $T_1 \cap T_2 = \varnothing$ *et* $\Sigma_2 \subseteq \Sigma_1$, *la combinaison asymétrique,* $\Theta = \Theta_1 \ltimes \Theta_2$, *est le RdPTÉPrA* $\langle P, T, A, \succ, \ell, L, \Sigma, I_s, Q\rangle$ *défini par :*

— $P = P_1 \cup P_2$;

— $T = T_1 \cup \hat{T_2} \cup T_s$, *avec :* $\hat{T_2} = \{t \in T_2 : \ell_2(t) = \lambda\}$; *et* $T_s = \bigcup_{l \in \Sigma_1 \cap \Sigma_2} \{t_1 \parallel t_2 : \exists (t_1, t_2) \in (T_1 \times T_2) : \ell_1(t_1) = \ell_2(t_2) = l\}$;

— $A = A_1 \cup \hat{A_2} \cup A_s$,
$\hat{A_2} = A_2 \setminus [(P_2 \times (T_2 \setminus \hat{T_2})) \cup ((T_2 \setminus \hat{T_2}) \times P_2)]$; *et*

$$\begin{aligned} A_s =& \{(p,t) \in (P \times T) : p \in P_1, t = t_1 \parallel t_2, (p, t_1) \in A_1\} \\ & \cup \{(p,t) \in (P \times T) : p \in P_2, t = t_1 \parallel t_2, (p, t_2) \in A_2\} \\ & \cup \{(t,p) \in (T \times P) : p \in P_1, t = t_1 \parallel t_2, (t_1, p) \in A_1\} \\ & \cup \{(t,p) \in (T \times P) : p \in P_2, t = t_1 \parallel t_2, (t_2, p) \in A_2\} \end{aligned}$$

— $\succ = \succ_1 \cup \hat{\succ_2} \cup \succ_s \cup \succ_\lambda$, *avec :* $\hat{\succ_2} = \succ_2 \cap (\hat{T_2} \times \hat{T_2})$, $\succ_\lambda = \bigcup_{(t_1,t_2) \in (T_1 \times \hat{T_2}) : \ell_1(t_1) = \lambda} \{(t_2, t_1)\}$; *et*

$$\succ_s = \bigcup_{t_s = t_1 \| t_2} \left\{ \bigcup_{i=1}^{2} \left\{ \bigcup_{(t_i,t) \in \succ_i} \{(t_s, t)\} \cup \bigcup_{(t,t_i) \in \succ_i} \{(t, t_s)\} \right\} \cup \{(t_s, t_1)\} \right\};$$

— $I(t) = \begin{cases} [0, +\infty[& si\ t = t_1 \parallel t_2 \in T_s \\ I_1(t) & si\ t \in T_1 \\ I_2(t) & si\ t \in T_2 \end{cases}$;

— $\ell(p) = \begin{cases} \ell_1(p) & si\ p \in P_1 \\ \ell_2(p) & si\ p \in P_2 \end{cases}$; $\ell(t) = \begin{cases} \ell_1(t_1) & si\ t = t_1 \parallel t_2 \in T_s \\ \ell_1(t) & si\ t \in T_1 \\ \ell_2(t) & si\ t \in T_2 \end{cases}$;

— $L = L_1 \cup L_2$ *et* $\Sigma = \Sigma_1 \cup \Sigma_2$;

— $Q = \{q_1 \cup q_2 : (q_1, q_2) \in Q_1 \times Q_2\}$ *et* $M_0(p) = \begin{cases} M_{10}(p) & si\ p \in P_1 \\ M_{20}(p) & si\ p \in P_2 \end{cases}$.

Les transitions non synchronisées gardent leur intervalle statique originel (issu des réseaux initiaux) et à chaque transition synchronisée est affecté un intervalle statique non contraint (c.-à-d. $[0, +\infty[$). Pour un système Θ et un pattern Ω, chaque marquage accessible M de $\Theta_\Omega = \Theta \ltimes \Omega$ est alors l'union de deux marquages, l'un accessible dans Θ et l'autre accessible dans Ω. $\forall \langle M, I\rangle \in \mathrm{R}(\Theta_\Omega, \langle M_{0\Theta_\Omega}, I_{0\Theta_\Omega}\rangle) : \exists \langle M_1, I_1\rangle \in \mathrm{R}(\Theta, \langle M_{0\Theta}, I_{0\Theta}\rangle), \exists \langle M_2, I_2\rangle \in \mathrm{R}(\Omega, \langle M_{0\,\Omega}, I_{0\,\Omega}\rangle) : M = M_1 \cup M_2$. De plus $\mathcal{L}(\Theta \ltimes \Omega) = \{\rho \in \mathcal{L}(\Theta) : \exists \omega \in \mathcal{L}(\Omega) : \rho \ni \omega\}$ [15]. On dispose ainsi d'un réseau $\Theta \ltimes \Omega$ qui détermine pour chaque exécution possible du système si cette exécution est concordante ou non avec le motif. Il ne reste plus qu'à exploiter le réseau $\Theta \ltimes \Omega$ en le confrontant à la séquence des observations produites par le système pour établir le diagnostic.

3.2.3 Diagnostic

Le problème de diagnostic de motif dans un système pour une séquence d'observations donnée peut finalement être résolu en déterminant l'intersection entre les séquences du système qui produisent cette séquence d'observations et celles qui sont concordantes avec le motif.

Le diagnostic peut donc être réalisé de la manière suivante. La séquence d'observation du système est d'abord modélisée par un réseau de Petri temporel (noté *Obs*). Par exemple pour une séquence d'observations $1o_1 3o_2$, *Obs* sera constitué d'une transition t_1 d'étiquette o_1 et d'intervalle $[1,1]$ suivi d'une transition t_2 d'étiquette o_1 et d'intervalle $[3,3]$, la place finale de *Obs* est notée p_{Obs}. Une décomposition temporelle des transitions est ensuite effectuée sur le réseau *Obs* ainsi que sur les transitions du réseau $\Theta \ltimes \Omega$ dont les étiquettes sont présentes dans *Obs*. Ensuite, l'opération consiste à fusionner les transitions de même étiquette entre $\Theta \ltimes \Omega$ et *Obs*, pour obtenir le produit synchronisé $Obs \parallel \Theta_\Omega$.

La fonction diagnostic $\mathcal{R}$-diagnostiqueur s'appuie alors sur deux questions posées au modèle checker `selt` et concernant le RdPTÉPrA $Obs \parallel \Theta_\Omega$:

1. Question $\varphi_{CERTAIN}$: est-on certain que le motif est reconnu dans chaque comportement du système qui produit les observations ? Si oui, la fonction de diagnostic renvoie $\mathcal{R}-certain$. Si non, la question suivante est alors posée :
2. Question φ_{SAIN} : est-on certain que le motif n'est jamais dans chaque comportement du système qui produit les observations ? Si la réponse est VRAI la fonction de diagnostic renvoie le résultat $\mathcal{R}-sain$, sinon le résultat de diagnostic est $\mathcal{R}-ambigu$.

Chaque question correspond à une formule SE-LTL. Soit M un marquage de $Obs \parallel \Theta_\Omega$, on note $M_{|\Omega}$ la restriction à Ω du marquage M,

1. $\varphi_{CERTAIN} \equiv \Box((sys\_nonAdm = 0 \wedge p_{Obs} = 1) \Rightarrow M_{|\Omega} \in Q_\Omega)$, autrement dit, est-il toujours vrai ($\Box$) que si le système produit *Obs* ($sys\_nonAdm = 0 \wedge p_{Obs} = 1$) alors ($\Rightarrow$) il est concordant avec le motif ($M_{|\Omega} \in Q_\Omega$).
2. $\varphi_{SAIN} \equiv \Box((sys\_nonAdm = 0 \wedge p_{Obs} = 1) \Rightarrow M_{|\Omega} \notin Q_\Omega)$, autrement dit, est-il est toujours vrai ($\Box$) que si le système produit *Obs* ($sys\_nonAdm = 0 \wedge p_{Obs} = 1$) alors ($\Rightarrow$) il n'est pas concordant avec le motif ($M_{|\Omega} \notin Q_\Omega$).

4 Application au Diagnostic d'un food-shop.

L'approche proposée est illustrée sur un magasin en ligne de produits alimentaires baptisé *foodshop* qui a été utilisé comme étude de cas dans le cadre du projet européen WSDIAMOND [10]. Dans cette application, les motifs correspondent davantage à des spécifications qu'à des défauts physiques d'un procédé et le diagnostic peut être vu davantage comme le diagnostic du modèle que le diagnostic appliqué au process. Ce système est représenté par le réseau de la figure 4. Le fonctionnement simplifié de ce système est le suivant : la boutique reçoit la commande d'un client (*sh_recvg*). Dans les 3 heures qui suivent la boutique doit démarrer deux activités (*sh_st[1,2]*) en parallèle : (1) une activité de tri des produits demandés en deux catégories : les produits qui peuvent être stockés (c'est à dire non périssables) et ceux qui ne le peuvent pas, cette activité se termine par l'événement de fin *sh_end_spl* ; (2) une activité de sélection d'un entrepôt. Pour les produits non périssables la boutique demande à l'entrepôt de vérifier leur disponibilité (*sh_req_whcheck*). Ce dernier fait la vérification (*wh_check*) et envoie une information de disponibilité (disponible *wh_send_av*, indisponible *wh_send_unav*). Pour les produits périssables la boutique s'assure de leur disponibilité directement auprès d'un fournisseur (*sh_req_suppcheck*) qui en vérifie ainsi la disponibilité (*sp_send_av*, *sp_send_unav*). Le foodshop exige que le fournisseur comme l'entrepôt donne une réponse sur la disponibilité des produits dans les 4 heures après avoir été sollicités. Quand la boutique reçoit les réponses, elle décide des suites à donner à la commande (*sh_decide*). Si un des produits est manquant la boutique annule les réservations (*sh_cancel*). Dans le cas contraire, la boutique demande

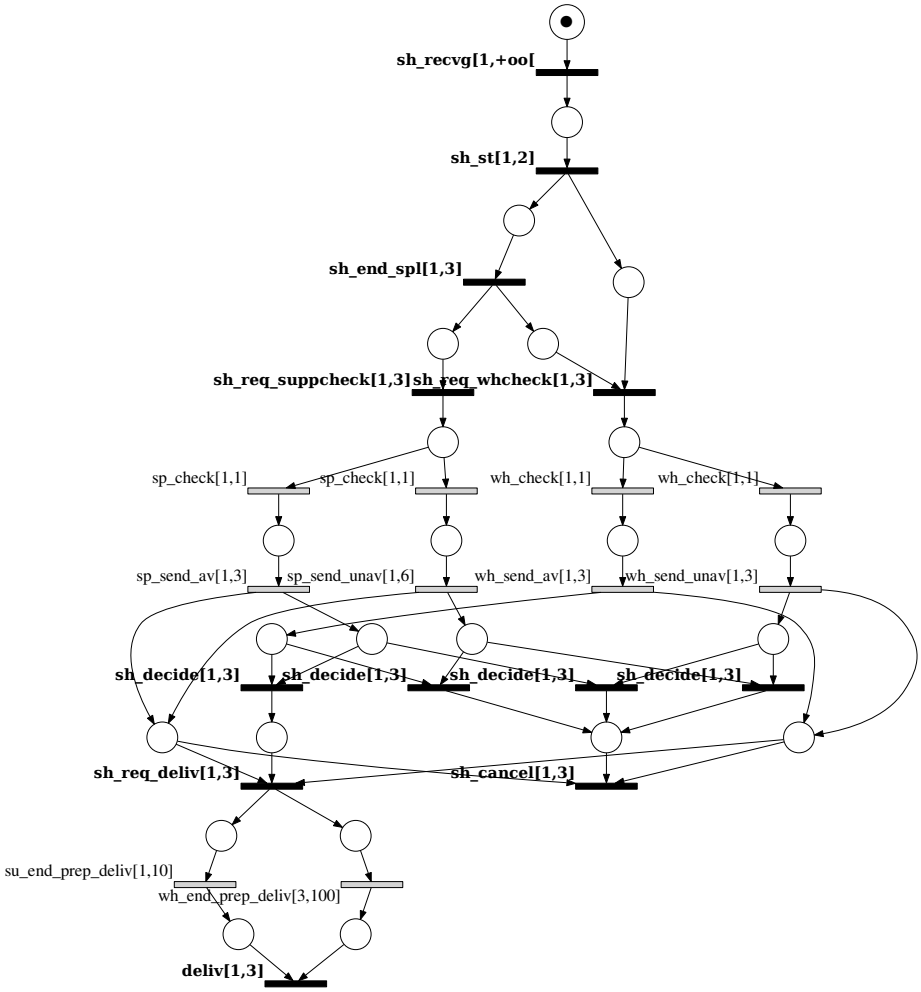

FIGURE 4 – Réseau du système *foodshop*.

la préparation de la commande (*sh_ req_ deliv*) auprès de l'entrepôt et du fournissseur. Quand les commandes sont prêtes (*wh_ end_ pre_ deliv* et *sp_ end_ prep_ deliv*) la commande est envoyée et la boutique, le fournisseur et l'entrepôt sont prêts pour traiter la commande du client suivant. [1] Sur la figure 4 Les événements observables sont indiqués en gras. Ils correspondent aux événements propres à la boutique qui supervise tout le processus et le suivi de commande, ceux liés au fournisseur et à l'entrepôt sont non observables. Pour illustrer l'approche nous

1. Dans cet exemple, seul l'agencement des activités pour la gestion d'un client est présenté. Un système réel met en œuvre bien évidemment plusieurs sessions parallèles de l'agencement présenté.

considérons deux motifs de comportements.

Occurrence de l'événement *sp_ send_ unav* : le premier motif est la simple occurrence de l'événement *sp_ send_ unav* représentant le fait que le fournisseur réponde à la boutique que les produits demandés ne sont pas disponibles. Ce simple motif nous permet de montrer que notre approche couvre le problème de diagnostic initialement défini pour de simples événements de fautes [25],[26]. Ce motif est composé de deux places pp_0 et pp_1 et d'une transition entre les deux. Le marquage initial M_0 est $M_0(pp_0) = 1$ et $M_0(pp_1) = 0$ et le seul marquage accepteur $M \in Q_\Omega$ est tel que $M(pp_0) = 0$ et $M(pp_1) = 1$. Supposons dans un premier temps que la séquence d'observations du système soit la suivante : $\sigma_1 =$ **50***sh_ recvg* **1***sh_ st* **1***sh_ end_ spl* **1***sh_ req_ whcheck* **1***sh_ req_ suppcheck* **3***sh_ decide* **2***sh_ req_ deliv* **10***deliv*.

La transformation de σ_1 en un réseau de Petri temporel consiste en une séquence de 8 transitions et de 9 places $\{p_0, \ldots, p_9\}$ avec pour marquage initial un jeton en place p_0 et pour marquage accepteur un jeton en place p_9. La transition t_1 entre p_0 et p_1 est étiquetée avec l'événement *sh_ recvg* et un intervalle temporel statique $[50, 50]$, et ainsi de suite pour les transitions $t_2, \ldots, t_8$. Pour résoudre le problème de diagnostic, on construit le produit entre le système et le motif qui est ensuite synchronisé avec le réseau de σ_1. Sur le réseau ainsi obtenu, on demande dans un premier temps à TINA de vérifier si le motif a toujours lieu dès lors que le système a produit la séquence observable σ_1 à l'aide de la formule suivante $\varphi_{CERTAIN}$:

```
[] ((sys_nonAdm = 0 /\ p9=1) => (pp1=1))
```

Dans cet exemple, la réponse de TINA est négative : il existe au moins une exécution du système qui produit σ_1 et pour laquelle *sp_ send_ unav* n'a pas eu lieu. On cherche alors à savoir si toutes les exécutions du système produisant σ_1 ne contiennent pas *sp_ send_ unav* à l'aide de la formule φ_{SAIN} :

```
[] ((sys_nonAdm = 0 /\ p9=1) => -(pp1=1))
```

La réponse est positive. Après l'observation de σ_1, le diagnostic est donc que *sp_ send_ unav* n'a pas eu lieu (pour s'en convaincre, il suffit de constater ici que σ_1 contient par exemple l'événement *sh_ req_ deliv* qui indique que tous les produits sont effectivement disponibles puisqu'il a été décidé de les livrer).

Supposons maintenant l'observation de la séquence : $\sigma_2 =$ **50***sh_ recvg* **1***sh_ st* **1***sh_ end_ spl* **1***sh_ req_ whcheck* **1***sh_ req_ suppcheck* **7***sh_ decide* **2***sh_ cancel*. Les cinq premiers événements de σ_2 sont ceux de σ_1. Dans σ_2, on voit que la décision est d'abandonner la livraison. Dans le cas de σ_2, TINA répond par la négative aux deux questions formulées : le diagnostic est ambigu, il n'est pas possible de savoir si la décision de non-livraison vient du fournisseur (occurrence du motif) ou seulement de l'entrepôt (non-occurrence du motif).

Enfin considérons maintenant la séquence : $\sigma_3 =$ **50***sh_ recvg* **1***sh_ st* **1***sh_ end_ spl* **1***sh_ req_ whcheck* **1***sh_ req_ suppcheck* **8***sh_ decide* **2***sh_ cancel*. La seule différence entre σ_2 et σ_3 est que dans σ_3, *sh_ decide* a lieu 8 unités de temps après *sh_ req_ suppcheck* et non pas 7. Dans ce cas, TINA répond positivement à $\varphi_{CERTAIN}$: toute exécution du système produisant σ_3 contient *sp_ send_ unav*. Il est alors inutile de vérifier φ_{SAIN} : le diagnostic est certain. Pour s'en convaincre, il suffit de remarquer sur la figure 4 que le temps de traitement par le fournisseur en cas de non disponibilité est parfois plus long que dans l'autre cas ($sp\_\ send\_\ unav[1, 6]$ et ($sp\_\ send\_\ av[1, 3]$), c'est l'observation de ce délai supplémentaire sur *sh_ decide* qui permet de lever l'ambiguité.

Occurrence de la séquence $wh\_\ send\_\ av[0, +\infty[\rightarrow wh\_\ end\_\ prep[70, 200]$: Nous illustrons l'emploi de notre méthode par un second motif de comportement qui fait apparaître plus

d'un événement. Le principe de ce motif est de superviser que dans le cas où l'entrepôt envoie une disponibilité des produits et réalise effectivement la préparation des produits, cette préparation se fait avec un délai supérieur à 70 unités de temps et inférieur à 200 unités de temps : on peut estimer que ce délai est intempestif et qu'il est nécessaire de le surveiller. Considérons la séquence d'observations : $\sigma_4 = \mathbf{50}sh\_recvg\ \mathbf{1}sh\_st\ \mathbf{1}sh\_end\_spl\ \mathbf{1}sh\_req\_whcheck\ \mathbf{1}sh\_req\_suppcheck\ \mathbf{3}sh\_decide\ \mathbf{2}sh\_req\_deliv\ \mathbf{50}deliv$.

TINA conclut ici que le motif n'a pas eu lieu (réponse négative $\varphi_{CERTAIN}$ et positive à φ_{SAIN}), la date de livraison étant à 50 après la requête, le motif ne peut effectivement pas avoir lieu. Maintenant si l'on modifie σ_4 en augmentant la date 50 en la date 80, le diagnostic change, il devient certain. Là encore c'est l'observation du temps qui discrimine.

5 Conclusions et perspectives

Ce travail s'inscrit dans le contexte du diagnostic de systèmes temporels. L'approche introduite permet d'étendre le problème originel du diagnostic de simples événements de fautes à des motifs de comportements plus complexes. L'objectif est de rechercher dans une séquence d'observation du système toutes les évolutions qui non seulement satisfont un motif de comportement temporel complexe mais qui également correspondent à un comportement du système surveillé. La fonction de diagnostic définie s'appuie sur la notion de concordance des séquences d'évolution du système qui sont cohérentes avec un motif de comportement. La concordance est mise en évidence par la construction d'un modèle réseau de Petri temporel qui détermine pour chaque execution possible du système si cette execution est concordante ou non avec le motif. Le diagnostic est alors établi en confrontant une séquence d'observation produite par le système avec ce modèle. Une résolution du problème de diagnostic s'appuyant sur l'outil $TINA$ et des techniques de model checking est proposée. Une des suites de ce travail concerne la résolution du problème de diagnostic dans le cadre d'un temps dense et l'étude des cas limites aux bornes.

Références

[1] Alfred V. AHO. Algorithms for finding patterns in strings. In Jan Van Leeuwen, editor, *Algorithms and Complexity*, Handbook of Theoretical Computer Science, pages 255 – 300. 1990.

[2] F. Basile, P. Chiacchio, and G. De Tommasi. Diagnosis of time petri nets using fault diagnosis graph. *IEEE Transactions on Automatic Control*, 54(4) :748–759, 2009.

[3] Albert Benveniste, Éric Fabre, Stefan Haar, and Claude Jard. Diagnosis of asynchronous discrete-event systems : a net unfolding approach. *Transactions on Automatic Control*, 48(5) :714–727, 5 2003.

[4] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H Roux. Comparison of different semantics for time Petri nets. In *Automated technology for verification and analysis*, pages 293–307. Springer, 2005.

[5] B. Berthomieu and F. Vernadat. *Traité IC2 "Systèmes temps réel 1 – Techniques de description et de vérification"*, chapter 1. Nicolas Navet, 2006.

[6] Bernard Berthomieu, Florent Peres, and François Vernadat. Model checking bounded prioritized time petri nets. In *Automated Technology for Verification and Analysis*. Springer, 2007.

[7] Patrice Bonhomme. Fault diagnosis of p-time labeled petri net systems. In *Proceedings of the 9th Workshop on Verification and Evaluation of Computer and Communication Systems, VECoS 2015, Bucharest, Romania, September 10-11, 2015.*, pages 11–22, 2015.

[8] Maria Paola Cabasino, Alessandro Giua, and Carla Seatzu. Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica*, 46(9) :1531–1539, 7 2010.

[9] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT press, 1999.

[10] Elisabetta Di Nitto, Anne-Marie Sassen, and Paolo Traverso. *WS-DIAMOND : Web Services ?DIAgnosability, MONitoring, and Diagnosis*, pages 213–240. MIT PRESS, 2009.

[11] M. Dotoli, M.P. Fanti, and A.M. Mangini. Fault detection of discrete event systems using petri nets and integer linear programming. In *17th IFAC World Congress*, pages 43–47, Seoul, Korea, 7 2008.

[12] C. Dousson, P. Gaborit, and M. Ghallab. Situation recognition : representation and algorithms. In *International Joint Conference on Artificial Intelligence*, pages 166–172, Chambéry, France, august 1993.

[13] Sahika Genc and Stéphane Lafortune. Distributed diagnosis of place-bordered Petri nets. *IEEE Transactions on Automation Science and Engineering*, 4(2) :206–219, 2007.

[14] Mohamed Ghazel, Armand Toguyéni, and Michel Bigand. A monitoring approach for discrete event systems based on a time Petri net model. In *16th IFAC World Congress*, pages 331 – 336, 2005.

[15] Houssam-Eddine Gougam. *Analyse de l'impact du temps sur la diagnosticabilité des systèmes à événements discrets*. PhD thesis, Université de Toulouse, INSA, France, 2015.

[16] Thierry Jéron, Hervé Marchand, Sophie Pinchinat, and Marie-Odile Cordier. Supervision patterns in discrete event systems diagnosis. In *8th International Workshop on Discrete Event Systems*, pages 262–268, Ann Arbor, MI, United States, 6 2006.

[17] Shengbing Jiang and Ratnesh Kumar. Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. *Transactions on Automatic Control*, 49(6) :934–945, 6 2004.

[18] Donald E Knuth, James H Morris, Jr, and Vaughan R Pratt. Fast pattern matching in strings. *SIAM journal on computing*, 6(2) :323–350, 1977.

[19] Dimitry Lefebvre and Catherine Delherm. Diagnosis of DES with Petri net models. *IEEE Transaction Automation Science and Engineering*, 4(1) :114–118, 2007.

[20] Baisi Liu, Mohammed Ghazel, and Armand Toguyéni. Diagnosis of labeled time Petri nets using time interval splitting. In *19th World Congress*, pages 43–47, Cape Town, South Africa, 8 2014.

[21] Philip Meir Merlin. *A Study of the Recoverability of Computing Systems*. PhD thesis, 1974.

[22] Yannick Pencolé, Anika Schumann, and Dmitry Kamenetsky. Towards low-cost fault diagnosis in large component-based systems. In *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 1473–1478, Beijing, China, 2006.

[23] Louchka Popova-zeugmann. Essential states in time petri nets. *Informatik-Berichte*, 96, 1998.

[24] R. Saddem, A. Toguyeni, and M. Tagina. Diagnosis of critical embedded systems : application to the control card of a railway vehicle braking systems. In *IEEE Conference on Automation Science and Engineering, CASE 2011, Trieste, Italy, Aug. 24-27, 2011*, pages 163–168, 2011.

[25] Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *Transactions on Automatic Control*, 40(9) :1555–1575, 9 1995.

[26] Stavros Tripakis. Fault diagnosis for timed automata. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, 7th International Symposium, FTRTFT 2002, Co-sponsored by IFIP WG 2.2, Oldenburg, Germany, September 9-12*, pages 205–224, 2002.

[27] Dogan Ulus, Thomas Ferrère, Eugene Asarin, and Oded Maler. *Timed Pattern Matching*, pages 222–236. Springer International Publishing, Cham, 2014.

[28] Masaki Waga, Takumi Akazaki, and Ichiro Hasuo. *A Boyer-Moore Type Algorithm for Timed Pattern Matching*, pages 121–139. Springer International Publishing, Cham, 2016.

[29] X. Wang, C. Mahulea, and M. Silva. Diagnosis of time petri nets using fault diagnosis graph. *IEEE Transactions on Automatic Control*, 60(9) :2321–2335, Sept 2015.

[30] Marina Zanella and Gianfranco Lamperti. Diagnosis of discrete-event systems by separation of concerns, knowledge compilation, and reuse. In *Proceedings of the 16th European Conference on*

Artificial Intelligence (ECAI04), pages 838–842, 2004.

[31] Janan Zaytoon and Stéphane Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37 :308–320, 2013.