Réseaux d'automates hybrides à synchronisations typées pour la modélisation des Systèmes Dynamiques Hybrides

Zulema Juarez Orozco, Bruno Denis, Jean-Jacques Lesage

Réseaux d'automates hybrides à synchronisations typées pour la modélisation des SDH

Zulema JUÁREZ[1,2]*, Bruno DENIS[1], Jean-Jacques LESAGE[1]-IEEE member

[1]LURPA, ENS Cachan, UniverSud Paris
61, Avenue du President Wilson, F94230 Cachan, France

[2]Centro de Technología Avanzada (CIATEQ)
Calz. del Retablo #150, Col. Fovissste, 76150, Querétaro, Qro., Méxique

{juarez, denis, lesage}@lurpa.ens-cachan.fr
http://lurpa.ens-cachan.fr

Résumé— **Ce papier présente une classe d'automates hybrides à haut niveau d'expressivité pour faciliter la modélisation modulaire des Systèmes Dynamiques Hybrides (SDH) : les automates hybrides à entrées/sorties et synchronisations typées. Elle combine le concept de modularité issue des automates à entrées/sorties [1] avec plusieurs mécanismes de synchronisation par étiquette de transition. Afin d'exploiter un tel modèle avec les outils existants nous proposons (i) un opérateur de composition pour fournir un automate « mis à plat », et (ii) un algorithme de traduction des mécanismes de synchronisations typées en classiques synchronisations par rendez-vous. L'intérêt de notre classe d'automates hybrides pour la modélisation modulaire est mis en évidence sur un exemple de SDH qui nous permet également d'illustrer l'opérateur de composition et l'algorithme de traduction des mécanismes de synchronisations typées.**

Mots-clés— **systèmes hybrides, réseau d'automates hybrides, modélisation modulaire, synchronisations typées, opérateur de composition.**

I. Introduction

L'interaction de dynamiques continues et de dynamiques discrètes fait de la modélisation des Systèmes Dynamiques Hybrides (SDH) un problème difficile. L'introduction des automates hybrides [2] a largement contribuée à donner un cadre rigoureux à l'expression de cette interaction entre dynamique continue et dynamique discrète. Pour les systèmes complexes une approche modulaire est par ailleurs indispensable à la modélisation. Dans ce cas un module décrit le comportement d'un sous-système de taille réduite, ce qui le rend plus propice à la modélisation par un automate hybride. La difficulté de modélisation est alors pour une large part reportée sur la définition des mécanismes de coordination entre les modules.

Les deux principales approches pour représenter ces coordinations sont la synchronisation par les étiquettes de transition [3], et la coordination par partage de variables [4]. Avec l'approche par synchronisation par les étiquettes de transition, les variables sont partagées par tous les automates (pas d'encapsulation dans les modules) et les franchissements de transitions peuvent être synchronisés entre plusieurs automates (notion d'évènement). L'approche par coordination par partage de variables laisse les automates évoluer de manière asynchrone, l'influence d'un automate sur un autre se faisant par observation de ces variables. Cette coordination est caractéristique des automates hybrides à entrées/sorties [4] ($HIOA$: Hybrid Input Output Automata) qui renforcent le concept de modularité en typant les variables : d'entrée, de sortie ou locale. Il existe une classe d'automates proposés par G. Frehse, alliant des synchronisations par les étiquettes et la modularité [5], mais elle a été développée spécifiquement pour la vérification formelle par « Model-Checking » et non pas plus globalement pour la modélisation des SDH. Les contraintes propres à l'élaboration d'un « Model-Checker » ont ainsi conduit l'auteur à limiter le pouvoir d'expression des synchronisations entre automates, pour en faciliter la vérification.

Parallèlement aux travaux sur les SDH, la synchronisation entre automates a fait l'objet de nombreux développements dans le domaine des Systèmes à Évènements Discrets (SED). Depuis les travaux fondateurs autour des Communicating Sequential Processes [6] ou de l'algèbre de processus CCS [7], des mécanismes de synchronisations à caractère bloquant ou non bloquant, entre deux automates ou plus, ont été étudiés. Des formalismes tels que les « Net Condition Event Systems » (NCES) [8] proposent ainsi au modélisateur des mécanismes multiples et évolués pour représenter les communications et les synchronisations entres automates.

C'est dans ce contexte que nous proposons une nouvelle classe d'automates hybrides à entrées/sorties et synchronisations typées ($HIOA_{TS}$: Hybrid Input Ouput Automata with Typed Synchronisations), qui prolonge les capacités sémantiques des automates hybrides linéaires définis dans [5] grâce à des mécanismes évolués de synchronisation, inspirés de ceux définis pour les SED dans les NCES.

La modélisation d'un SDH n'est par ailleurs pas une fin en soit. L'évaluation de performances, l'analyse formelle, la recherche de régions atteignables ou le diagnostic sont parmi les activités qui motivent généralement l'élaboration du modèle. Pour rendre la modélisation par $HIOA_{TS}$ compatible avec ces différentes utilisations nous proposons deux voies : (i) la composition parallèle d'automates $HIOA_{TS}$

* Bourse financée par le CONACYT (Conseil National de Science et Technologie, Mexique)

qui génère un unique automate hybride composé exempt de synchronisations, (ii) la traduction d'un réseau d'$HIOA_{TS}$ en un réseau d'automates qui se coordonnent par synchronisations classiques (rendez-vous bloquant).

Cet article est organisé de la manière suivante. La partie II est consacrée à la définition des automates hybrides à entrées/sorties à synchronisations typées. Un exemple est utilisé pour illustrer la nécessité de définir différents types de synchronisations entre automates. Le comportement global d'un SDH modélisé par une approche modulaire étant donné par l'interaction d'automates, un opérateur de composition est également défini. La partie III présente un algorithme de traduction pour exprimer le comportement des automates hybrides à entrées/sorties à synchronisations typées avec à la sémantique d'un outil d'analyse existant. Enfin, l'exemple présenté en partie II est traduit puis analysé avec le « model-checker » polyédrique PHAVer.

II. Réseaux d'Automates Hybrides Linéaires à Synchronisations Typées

A. Les automates hybrides linéaires à entrées/sorties

La classe d'automates que nous proposons est une extension des automates hybrides linéaires à entrées/sorties ($HIOA$) retenus par G. Frehse dans ses travaux sur le « model-checker » PHAVer [5]. Un exemple d'$HIOA$ est donné sur la fig 1, nous allons en rappeler la définition formelle.

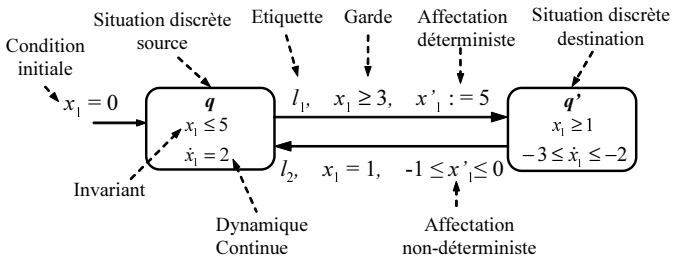


Fig. 1. Syntaxe d'un automate hybride linéaire $HIOA$

Un automate hybride linéaire à entrées/sorties $HIOA = \langle \mathbb{Q}, \mathbb{X}, \mathbb{L}, \mathbb{T}, \mathcal{A}, \mathcal{F}, \mathcal{I}, Init \rangle$ consiste en :

- un ensemble de situations discrètes $\mathbb{Q}$;
- un ensemble de variables réelles $\mathbb{X} = \mathbb{X}_I \cup \mathbb{X}_O \cup \mathbb{X}_L$, où $\mathbb{X}_I$ est l'ensemble des variables d'entrée, $\mathbb{X}_O$ l'ensemble des variables de sortie et $\mathbb{X}_L$ l'ensemble des variables locales. De plus, $\mathbb{X}_I \cap \mathbb{X}_O = \mathbb{X}_I \cap \mathbb{X}_L = \mathbb{X}_O \cap \mathbb{X}_L = \varnothing$. Les variables appartenant à $\mathbb{X}_L$ où à $\mathbb{X}_O$ sont les variables *contrôlées* par l'automate, tandis que les variables appartenant à $\mathbb{X}_I$ sont des variables *non-contrôlées* ;
- un ensemble d'étiquettes $\mathbb{L} = \mathbb{L}_I \cup \mathbb{L}_O \cup \mathbb{L}_L$ où $\mathbb{L}_I$ représente l'ensemble des étiquettes d'entrée, $\mathbb{L}_O$ l'ensemble des étiquettes de sortie et $\mathbb{L}_L$ l'ensemble des étiquettes locales ;
- un ensemble de transitions $\mathbb{T} \subseteq \mathbb{Q} \times \mathbb{L} \times \mathbb{G} \times \mathbb{Q}$, où si $(q, l, g, q') \in \mathbb{T}$ alors :
  - q et q' sont des situations discrètes source et destination, éléments de $\mathbb{Q}$,
  - l est une étiquette, élément de $\mathbb{L}$
  - g est une garde, élément de $\mathbb{G}$, c'est-à-dire, une condition sur la valeur des variables de $\mathbb{X}$ exprimée sous la forme d'une conjonction de prédicats éléments de $\mathbb{P}$. $\mathbb{P}$ est l'ensemble des prédicats définis sur $\mathbb{X}$. Un prédicat est une égalité ou une inégalité entre combinaisons linéaires des variables de $\mathbb{X}$ à coefficients constants et rationnels ;
- une fonction $\mathcal{A}$ qui associe à une transition (q, l, g, q') une liste d'affectations des variables contrôlées $\mathbb{X}_O$ et $\mathbb{X}_L$. Chaque affectation est spécifiée par une conjonction de prédicats de $\mathbb{P}$;
- une fonction $\mathcal{F}$ qui décrit la dynamique continue de chaque situation de $\mathbb{Q}$, en associant à chaque couple $(q, x_C) \in \mathbb{Q} \times (\mathbb{X}_O \cup \mathbb{X}_L)$ une conjonction de prédicats de la forme $\dot{x_C} \ge c(\mathbb{X})$, $\dot{x_C} \le c(\mathbb{X})$ ou $\dot{x_C} = c(\mathbb{X})$. Avec $c(\mathbb{X})$ une combinaison linéaire des variables de $\mathbb{X}$ à coefficients constants et rationnels ;
- une fonction $\mathcal{I}$ qui associe un invariant à chaque situation discrète de $\mathbb{Q}$. Cet invariant est une région de l'espace d'état des variables de $\mathbb{X}$ défini sous la forme d'une disjonction de conjonctions de $\mathbb{P}$;
- un état initial *Init* définissant une situation initiale $q_0 \in \mathbb{Q}$ et un domaine de l'espace d'état des variables de $\mathbb{X}_O \cup \mathbb{X}_L$ défini sous la forme d'une conjonction de prédicats de $\mathbb{P}$.

L'état courant d'un automate hybride linéaire est défini par un couple $(q, v) \in \mathbb{Q} \times \mathbb{R}^{|\mathbb{X}_O \cup \mathbb{X}_L|}$ où v est une valuation de l'ensemble des variables contrôlées. L'état d'un $HIOA$ peut donc être modifié de deux manières :

(i) par franchissement d'une transition $(q, l, g, q') \in \mathbb{T}$ qui change la situation courante q en q', et qui fait évoluer les variables contrôlées de $\mathbb{X}$ selon la fonction $\mathcal{A}$. Ce franchissement n'est possible que lorsque la garde l'autorise,

(ii) par l'écoulement du temps dans une situation qui fait évoluer la valeur des variables contrôlées selon la fonction $\mathcal{F}$. Un $HIOA$ peut demeurer dans une situation q tant que l'invariant $\mathcal{I}(q)$ est satisfait.

Un automate $HIOA$ doit quitter une situation discrète q si l'invariant $\mathcal{I}(q)$ n'est pas satisfait. Une transition doit alors être franchissable sous peine de blocage.

B. Réseau d'automates linéaires HIOA

Pour la modélisation des systèmes complexes, une approche modulaire est souvent indispensable. Elle conduit à la construction d'un ensemble de $HIOAs$ en interaction appelé Réseau d'$HIOA$, défini formellement comme : $R = \{i \in [1, n] \mid HIOA_i = \langle \mathbb{Q}_i, \mathbb{X}_i, \mathbb{L}_i, \mathbb{T}_i, \mathcal{A}_i, \mathcal{F}_i, \mathcal{I}_i, Init_i \rangle\}$ $\forall (i, j) \in [1, n]^2$ avec $i \neq j$: les ensembles des situations sont disjoints deux à deux $\mathbb{Q}_i \cap \mathbb{Q}_j = \varnothing$, les variables locales n'appartiennent à aucun autre automate $\mathbb{X}_{L_i} \cap \mathbb{X}_j = \varnothing$, les variables de sorties ne peuvent pas être des variables contrôlées dans un autre automate $\mathbb{X}_{O_i} \cap (\mathbb{X}_{O_j} \cup \mathbb{X}_{L_j}) = \varnothing$.

Pour définir le comportement d'un système modélisé par un réseau d'automates il est de plus nécessaire de définir comment les étiquettes associées aux transitions permettent la synchronisation des évolutions entre automates. Les automates d'un réseau d'$HIOA$ se synchronisent par rendez-vous bloquant, mais dans la littérature on trouve

de manière plus générale différents types de synchronisation que l'on peut classer selon :

- deux structures de synchronisation : les synchronisations biparties (1 automate émetteur, 1 automate récepteur) (CCS [7], UPPAAL [9]) et les synchronisations multiparties (1 automate émetteur, N automates récepteurs) (CSP [6], LOTOS [10]) ;
- quatre types de synchronisation :

(i) l'interaction asynchrone, où les étiquettes ne sont pas utiles et où les interactions entre automates se font sur partages de variables réelles, comme dans les automates à entrées/sorties [1] ;

(ii) la synchronisation par rendez-vous non-bloquant, où un automate émetteur émet une étiquette de synchronisation sans être bloqué par le comportement des récepteurs, alors que les récepteurs sont bloqués en attente de l'émission d'une étiquette de synchronisation. Ce type de synchronisation est par exemple utilisé dans les automates UPPAAL [11] ;

(iii) la synchronisation par rendez-vous bloquant où tous les automates qui possèdent la même étiquette l doivent s'attendre pour franchir simultanément leur transition étiquettée l. Par contre, ils évoluent de manière asynchrone pour les transitions sans étiquette partagée. Ce type de synchronisation très répandu est décrit dans [12] et utilisé par exemple dans HYTECH [3] et dans les *HIOA* de PHAVer [5] ;

(iv) l'interaction totalement synchrone où tous les automates du réseau n'évoluent que par franchissement des transitions possédant une étiquette partagée par tous. Aucune des autres transitions ne pourra être franchie [12].

C. Besoins du modélisateur en synchronisations typées

Lorsqu'il effectue une modélisation d'un système complexe, le concepteur se heurte souvent à des difficultés pour exprimer la coordination entre les automates de modules. Le besoin d'étendre les capacités sémantiques des automates hybrides linéaires pour offrir de multiples capacités d'exprimer différents types de synchronisation entre les automates est bien réel. Nous proposons pour cela une classe d'automates qui offre les trois premiers types de synchronisation et qui autorise la structure multipartie (1-émetteur, N-récepteurs). Par contre le dernier type de synchronisation ne peut pas cohabiter avec les trois autres car il bannit tout asynchronisme même partiel. Pour illustrer l'intérêt de cette classe, nous allons traiter un exemple volontairement simple mais illustratif des besoins de modélisation en terme de communication inter-automates. Il s'agit de deux véhicules circulant sur deux voies se croisant à un carrefour protégé par deux feux de signalisation (fig 2).

Le modèle de ce système peut être obtenu à l'aide de quatre *HIOA*s, un par véhicule et un par feu. Une vue globale du modèle ainsi obtenu, ne décrivant que sa structuration en un réseau d'*HIOA*s et les échanges de données et de synchronisation entre *HIOA*s, est donné sur la figure 3(b). Nous avons pour cela retenu une représentation graphique, inspirée des « Function Blocs » de la norme CEI 61499 [13] (figure 3(a)). Ces quatre automates n'échangent pas de variables continues et ils se coordonnent uniquement par synchronisation. Le Feu_X émet le signal de synchronisation start_x à destination de tous véhicule qui serait à l'arrêt sur la voie X pour signaler son passage au vert. Cette synchronisation doit être non-bloquante car le feu ne doit pas attendre de rendez-vous avec un véhicule pour passer au vert. Il est donc nécessaire ici d'utiliser une synchronisation non-bloquante. Il en va de même pour le signal de synchronisation stop_x issu du Feu_X. Cependant stop_x est par ailleurs un signal de synchronisation d'entrée pour Feu_Y où il est légitime d'utiliser cette fois une synchronisation par rendez-vous bloquant pour des raisons évidentes de sécurité. Cela fait apparaître le besoin d'une *synchronisation mixte* pour stop_x entre trois automates : un émetteur (Feu_X), un récepteur non-bloquant (Véhicule_X) et un récepteur bloquant (Feu_Y).

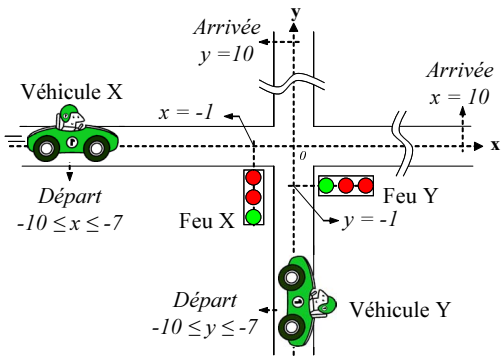
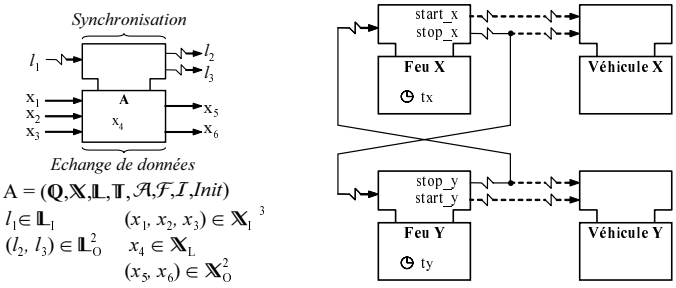


Fig. 2. Exemple : deux véhicules se croisant à un carrefour



(a) un exemple d'*HIOA* et de ses interactions avec d'autres *HIOA*s

(b) réseau d'*HIOA* modélisant le carrefour

Fig. 3. Représentation d'un réseau d'automates

C'est l'identification de ce besoin de modélisation qui nous a conduit à proposer une extension de *HIOA* : les *HIOA* à synchronisations typées ($HIOA_{TS}$). Dans un $HIOA_{TS}$, le type de synchronisation souhaité (émetteur / récepteur bloquant / récepteur non-bloquant) est associé aux transitions à l'aide des étiquettes.

D. Les automates hybrides linéaires à entrées/sorties et à synchronisations typées $HIOA_{TS}$

Un automate hybride à entrées/sorties et à synchronisations typées ($AHES_{TS}$) est de la forme $H_{HIOA_{TS}} = \langle H_{HIOA}, \mathcal{R} \rangle$ avec :

- H_{HIOA} un automate hybride linéaire à entrées/sorties tel que défini en section II-A,
- $\mathcal{R}$ une fonction qui associe un *rôle* aux étiquettes, avec $R : \mathbb{L} \rightarrow \{!, ?, ¿\}$.

L'association d'un rôle aux étiquettes impose deux conditions suplémentaires aux $HIOA$s d'un réseau :

- $\mathbb{L}_I \cap \mathbb{L}_O = \mathbb{L}_O \cap \mathbb{L}_L = \mathbb{L}_L \cap \mathbb{L}_I = \varnothing$;
- $\mathbb{L}_{L_i} \cap \mathbb{L}_j = \varnothing$ et $\mathbb{L}_{O_i} \cap (\mathbb{L}_{O_j} \cup \mathbb{L}_{L_j}) = \varnothing$.

La multiplicité des types de synchronisation entre automates est obtenue en associant un rôle aux étiquettes : le rôle émetteur symbolisé par « ! », le rôle récepteur bloquant symbolisé par « ? » et le rôle récepteur non-bloquant symbolisé par « ¿ ». Par construction toute les étiquettes de sortie ont le rôle d'émetteur. La table I présente les trois types de synchronisation entre trois automates et leur implication sur le rôle des étiquettes.

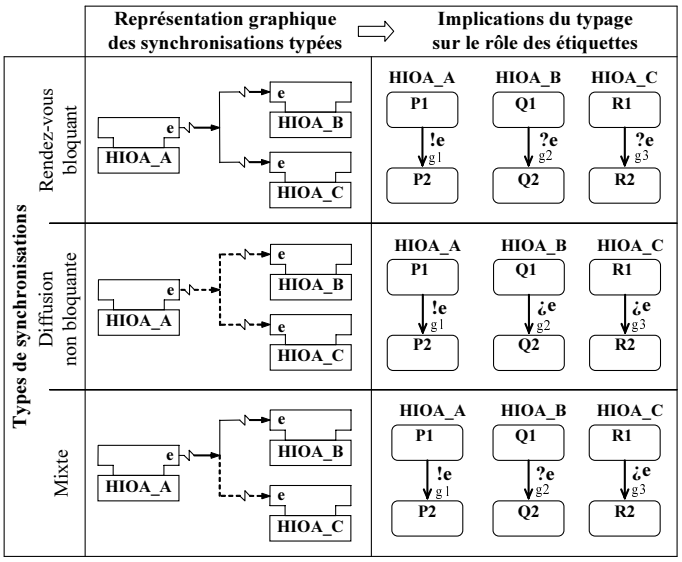


TABLE I
EXEMPLES DE SYNCHRONISATIONS TYPÉES

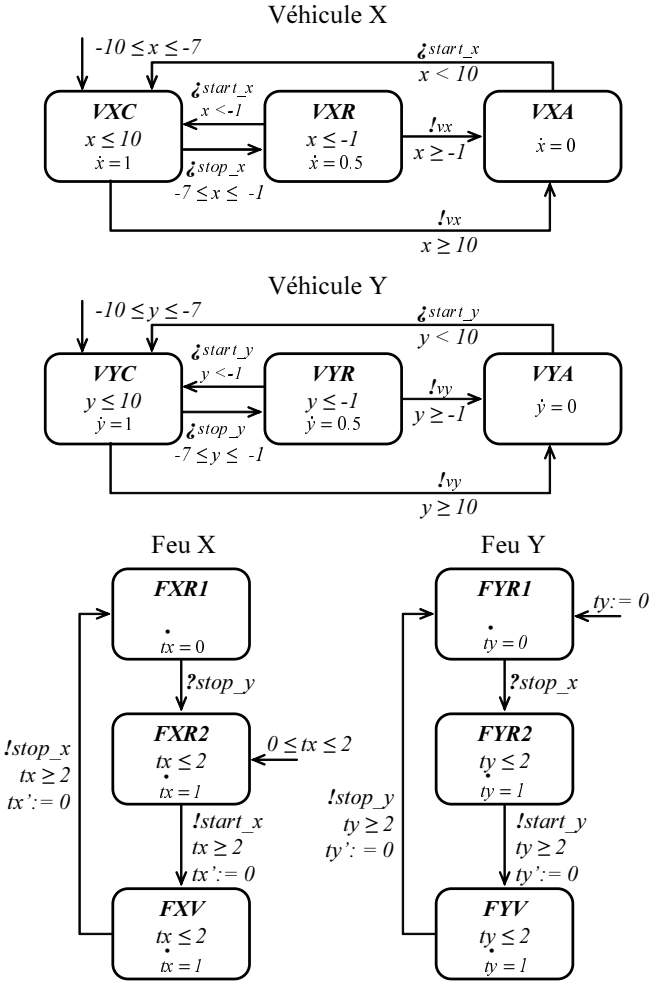


Fig. 4. Modèles $HIOA_{TS}$ de l'exemple

La figure 4 donne in extenso les quatre $HIOA_{TS}$ qui modélisent le carrefour où l'on peut observer le rôle donné à chaque étiquette. Par exemple, l'évènement stop_x est utilisé dans les trois automates avec à chaque fois un rôle différent :

- Feu_X : FXV $\xrightarrow{!stop\_x,\ldots}$ FXR1 (rôle émetteur)
- Feu_Y : FYR1 $\xrightarrow{?stop\_x,\ldots}$ FYR2 (rôle récepteur bloquant)
- Véhicule_X : VXC $\xrightarrow{¿stop\_x,\ldots}$ VXR (rôle récepteur non bloquant)

E. Composition des automates d'un réseau d'$HIOA_{TS}$

Soient H_1 et H_2 deux $HIOA_{TS}$. L'automate composé $H = H_1 \| H_2 = \langle \mathbb{Q}, \mathbb{X}, \mathbb{L}, \mathbb{T}, \mathcal{A}, \mathcal{F}, \mathcal{I}, Init, \mathcal{R} \rangle$ est défini comme suit :

- $\mathbb{Q}$ = partie atteignable de $\mathbb{Q}_1 \times \mathbb{Q}_2$;
- $\mathbb{X}_O = \mathbb{X}_{O1} \cup \mathbb{X}_{O2}$, $\mathbb{X}_I = (\mathbb{X}_{I1} \cup \mathbb{X}_{I2}) - \mathbb{X}_O$, et $\mathbb{X}_L = (\mathbb{X}_1 \cup \mathbb{X}_2) - (\mathbb{X}_O \cup \mathbb{X}_I)$;
- $\mathbb{L}_O = \mathbb{L}_{O1} \cup \mathbb{L}_{O2}$, $\mathbb{L}_I = (\mathbb{L}_{I1} \cup \mathbb{L}_{I2}) - \mathbb{L}_O$ et $\mathbb{L}_L = (\mathbb{L}_1 \cup \mathbb{L}_2)$;
- $R : \mathbb{L} \to \{!, ?, ¿\}$

$$l \mapsto \begin{cases} \text{"!"} & \text{si} \quad (\mathcal{R}_1(l) = \text{"!"} \vee \mathcal{R}_2(l) = \text{"!"}) \\ \text{"?"} & \text{si } \neg(\mathcal{R}_1(l) = \text{"!"} \vee \mathcal{R}_2(l) = \text{"!"}) \wedge \\ & \qquad (\mathcal{R}_1(l) = \text{"?"} \vee \mathcal{R}_2(l) = \text{"?"}) \\ \text{"¿"} & \text{sinon} \end{cases}$$

- $\mathbb{T} \subseteq \mathbb{Q} \times \mathbb{L} \times \mathbb{G} \times \mathbb{Q}$ avec $\mathbb{G} \subseteq \mathbb{P}(\mathbb{X})$.
  Nous utiliserons indifféremment les notations $q \xrightarrow{l,g} q'$ et (q, l, g, q') pour représenter les transitions. $\forall (t_1, t_2) \in \mathbb{T}_1 \times \mathbb{T}_2$ telles que $t_1 = q_1 \xrightarrow{l_1,g_1} q_1'$ et $t_2 = q_2 \xrightarrow{l_2,g_2} q_2'$, alors

  Cas 1 si $l_1 \neq l_2$ alors,
  $(q_1, q_2) \xrightarrow{l_1,g_1} (q_1', q_2) \in \mathbb{T}$ et $\mathcal{A}(t) = A(t_1)$ et $(q_1, q_2) \xrightarrow{l_2,g_2} (q_1, q_2') \in \mathbb{T}$ et $A(t) = A(t_2)$;

  Cas 2 si $l_1 = l_2$ (noté l) et $(\mathcal{R}_1(l_1), \mathcal{R}_2(l_2)) = (!, ?)$ ou $(?, !)$ ou $(?, ?)$ alors,
  $(q_1, q_2) \xrightarrow{l,g_1 \wedge g_2} (q_1', q_2') \in \mathbb{T}$ et $A(t) = A(t_1) \cup A(t_2)$;

  Cas 3 si $l_1 = l_2$ (noté l) et $(\mathcal{R}_1(l_1), \mathcal{R}_2(l_2)) = (!, ¿)$ ou $(?, ¿)$ alors,
  $(q_1, q_2) \xrightarrow{l,g_1 \wedge g_2} (q_1', q_2') \in \mathbb{T}$ et $A(t) = A(t_1) \cup A(t_2)$ et $(q_1, q_2) \xrightarrow{l,g_1 \wedge (\neg g_2)} (q_1', q_2) \in \mathbb{T}$ et $A(t) = A(t_1)$;

  Cas 4 si $l_1 = l_2$ (noté l) et $(\mathcal{R}_1(l_1), \mathcal{R}_2(l_2)) = (¿, !)$ ou $(¿, ?)$ alors,
  $(q_1, q_2) \xrightarrow{l,g_1 \wedge g_2} (q_1', q_2') \in \mathbb{T}$ et $A(t) = A(t_1) \cup A(t_2)$ et $(q_1, q_2) \xrightarrow{l,(\neg g_1) \wedge g_2} (q_1, q_2') \in \mathbb{T}$ et $A(t) = A(t_2)$;

  Cas 5 si $l_1 = l_2$ (noté l) et $(\mathcal{R}_1(l_1), \mathcal{R}_2(l_2)) = (¿, ¿)$ alors,

$(q_1, q_2) \xrightarrow{l, g_1 \wedge (\neg g_2)} (q_1', q_2) \in \mathbb{T}$ et $A(t) = A(t_1)$ et
$(q_1, q_2) \xrightarrow{l, (\neg g_1) \wedge g_2} (q_1, q_2') \in \mathbb{T}$ et $A(t) = A(t_2)$ et
$(q_1, q_2) \xrightarrow{l, g_1 \wedge g_2} (q_1', q_2') \in \mathbb{T}$ et $A(t) = A(t_1) \cup A(t_2)$.

- $\forall (q_1, q_2) \in \mathbb{Q}$,
  $\forall x_{C1} \in (\mathbb{X}_{L1} \cup \mathbb{X}_{O1}), \mathcal{F}((q_1, q_2), x_{C1}) = \mathcal{F}_1(q_1, x_{C1})$ et
  $\forall x_{C2} \in (\mathbb{X}_{L2} \cup \mathbb{X}_{O2}), \mathcal{F}((q_1, q_2), x_{C2}) = \mathcal{F}_2(q_2, x_{C2})$;
- $\forall (q_1, q_2) \in \mathbb{Q}, \mathcal{I}((q_1, q_2)) = \mathcal{I}_1(q_1) \wedge \mathcal{I}_2(q_2)$;
- Si $Init_1 = (q_{01}, d_{01})$ et $Init_2 = (q_{02}, d_{02})$ alors $Init = ((q_{01}, q_{02}), d_{01} \wedge d_{02})$.

Pour illustrer la composition parallèle que l'on vient de définir, la composition des automates Feu_X et Véhicule_X de l'exemple est présentée dans la figure 5. L'automate résultant de cette composition par synchronisations typées est constitué de 8 situations et 16 transitions.

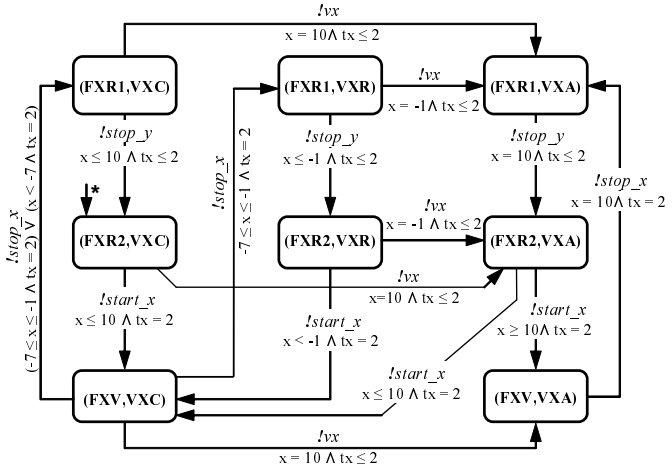


Fig. 5. Composition des automates Véhicule_X et Feu_X

III. Traduction d'un réseau d'$HIOA_{TS}$

Toutes les utilisations d'un réseau d'automate $HIOA_{TS}$ ne requièrent pas en entrée l'automate composé. C'est par exemple le cas de la vérification par « Model-checking » où le « model-checker » réalise lui même la composition des automates à vérifier. Dans ce cas , il est plus facile de passer par une traduction du réseau d'automate $HIOA_{TS}$ dans la classe d'entrée de l'outil utilisé. Pour cette communication nous avons retenu PHAVer comme outil de vérification. Dans ce cas, les enrichissements que nous avons apportés à la classe d'automate $HIOA$ doivent donc être traduits à l'aide des seules synchronisations multiparties à rendez-vous bloquant dont dispose PHAVer. Pour un automate possédant des transitions avec une étiquette l au rôle non bloquant, sa traduction implique la mise en place depuis toutes les situations d'une transition de type « self loop » avec l'étiquette l et sans affectation. Si avant traduction, une situation était déjà source d'une ou plusieurs transitions t_i avec l'étiquette l, alors la garde de cette nouvelle transition doit être le complément logique de la conjonction des gardes des transtions t_i (fig 6), si non la garde est sans restriction.

D'une manière plus générale, la traduction systématique d'un réseau d'automate $HIOA_{TS}$ en un réseau d'automates synchronisés par rendez-vous bloquant tels qu'utilisés dans PHAVer est décrite par l'algorithme 1. Pour illustrer cette traduction, le résultat de la traduction des modules Feu_X et Véhicule_X est présenté dans la figure 7.

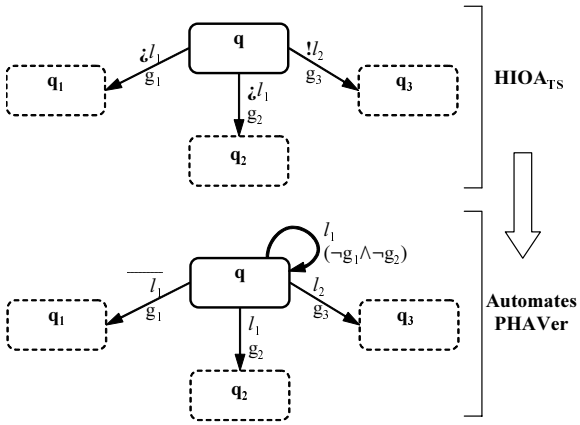


Fig. 6. Traduction des automates $HIOA_{TS}$ à la syntaxe du « model-checker » PHAVer

Algorithme 1 : Traduction d'un réseau d'automate $HIOA_{TS}$

Data : R_{TS}, un réseau d'automates $HIOA_{TS}$
Result : R_{tr}, un réseau d'automate PHAVer
$R_{tr} \leftarrow \emptyset$
forall $\langle \mathbb{Q}_c, \mathbb{X}_c, \mathbb{L}_c, \mathbb{T}_c, \mathcal{R}_c, \mathcal{A}_c, \mathcal{F}_c, \mathcal{I}_c, Init_c \rangle \in R_{TS}$ **do**
  $\mathbb{T}_{tr} \leftarrow \emptyset$
  forall $l_c \in \{l \in \mathbb{L}_c \mid \mathcal{R}(l) = "¿"\}$ **do**
    forall $q_c \in \mathbb{Q}_c$ **do**
      $T_c \leftarrow \{q \xrightarrow{l,g} q' \in \mathbb{T}_c \mid q = q_c \text{ et } l = l_c\}$
      if $T_c = \emptyset$ **then**
        $\mathbb{T}_{tr} \leftarrow \mathbb{T}_{tr} \cup \{q_c \xrightarrow{l_c, vrai} q_c\}$
      else
        $\mathbb{T}_{tr} \leftarrow \mathbb{T}_{tr} \cup T_c$, $g_{self\_loop} \leftarrow vrai$
        forall $q_c \xrightarrow{l_c, g_c} q_c' \in T_c$ **do**
          $g_{self\_loop} \leftarrow g_{self\_loop} \wedge (\neg g_c)$
        $\mathbb{T}_{tr} \leftarrow \mathbb{T}_{tr} \cup \{q_c \xrightarrow{l_c, g_{self\_loop}} q_c\}$
  forall $t \in \mathbb{T}_{tr}$ **do**
    if $t \in \mathbb{T}_c$ **then**
      $\mathcal{A}_{tr}(t) \leftarrow \mathcal{A}_c(t)$
    else
      $\mathcal{A}_{tr}(t) \leftarrow \emptyset$
  $R_{tr} = R_{tr} \cup \langle \mathbb{Q}_c, \mathbb{X}_c, \mathbb{L}_c, \mathbb{T}_{tr}, \mathcal{A}_{tr}, \mathcal{F}_c, \mathcal{I}_c, Init_c \rangle$

A titre d'exemple d'utilisation de la traduction d'$HIOA_{TS}$ selon l'algorithme 1, nous proposons l'étude de la région atteignable du réseau d'automates $HIOA_{TS}$ modélisant le carrefour (fig 4). Le « model-checker » PHAVer est alors utilisé pour rechercher la région atteignable par le modèle traduit. La figure 8 répresente la projection de cette région atteignable dans le plan X et Y (les variables des automates véhicules_X et véhicule_Y).

L'objectif des feux de croisement, tel qu'il est décrit par le réseau d'automates $HIOA_{TS}$ de la figure 4, est d'éviter la présence simultanée des deux véhicules à l'intersection. La surface du carrefour correspond à la région $-1 \leq x \leq 1$ et $-1 \leq y \leq 1$. On constate sur la figure 8

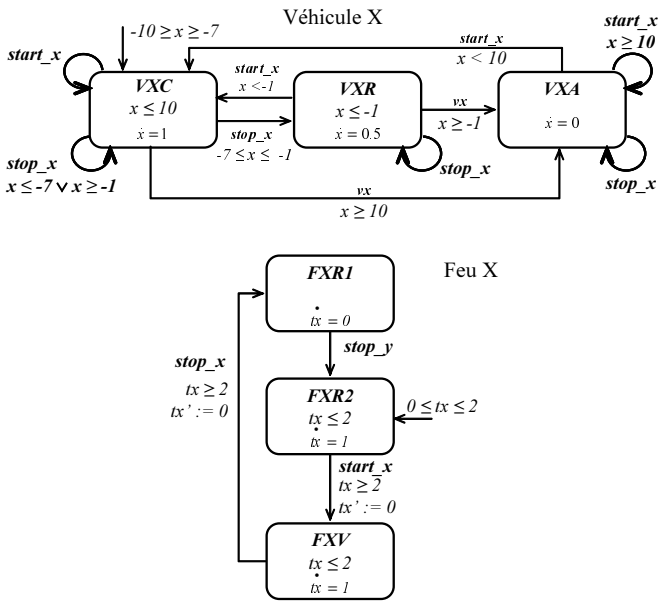


Fig. 7. Traduction des automates Véhicule_X et Feu_X

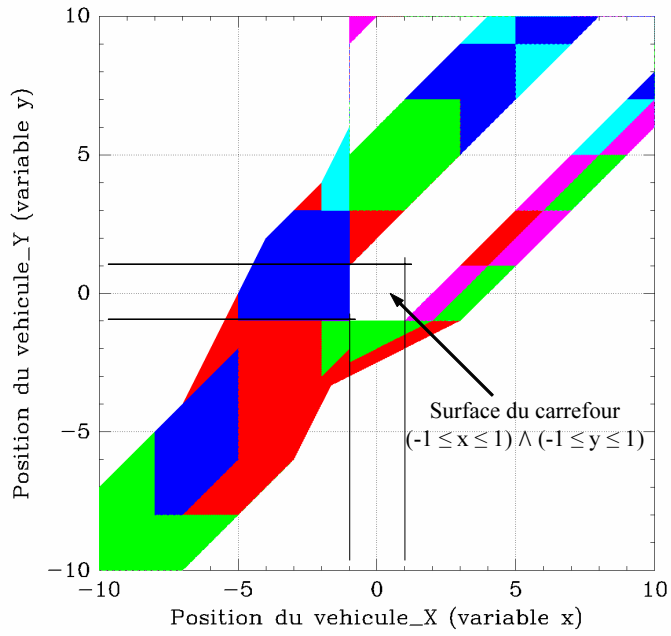


Fig. 8. Positions atteignables par les deux véhicules

que l'intersection entre la surface du carrefour et la region atteignable par les deux variables x et y est vide. Une fois traduit par l'algorithme 1, les synchronisations typées entre les différents automates ont donc parfaitement jouées leur rôle au sein du modèle.

IV. Conclusion

Dans ce papier nous avons présenté une classe d'automates hybrides linéaires à synchronisations typées pour la modélisation modulaire des SDH complexes : les $HIOA_{TS}$. Elle offre plusieurs mécanismes de synchronisation entre automates : des synchronisations multiparties (1 automate émetteur, $N \geq 1$ automates récepteurs) sur rendez-vous bloquant, non bloquant ou mixte. Bien que construite sur la base des automates hybrides à entrées/sorties, notre proposition de mécanismes de synchronisation peut se transposer à toutes les classes d'automates incluant la notion d'étiquette de transition, puisque c'est le rôle attribué à l'étiquette qui est porteur de la sémantique de synchronisation.

De manière corollaire, en proposant de puissants mécanismes de synchronisation pour satisfaire les besoins des modélisateurs, un écart s'est creusé entre le formalisme $HIOA_{TS}$ et les formalismes d'entrée des principaux outils d'analyse ou de diagnostic des SDH. Pour combler cet écart deux propositions complémentaires ont été présentées. La première est la définition d'un opérateur de composition des $HIOA_{TS}$. Il permet l'obtention de façon systématique d'un seul automate composé à partir d'un réseau d'$HIOA_{TS}$. Cet automate composé n'est donc plus porteur d'aucune marque de moduarité ni de synchronisation, ce qui lui confère une large compatibilité pour tout usage ultérieur. La deuxième proposition pour faciliter l'exploitation d'un réseau d'$HIOA_{TS}$ est un algorithme de traduction des mécanismes de synchronisations typées à l'aide du mécanisme le plus répandu de synchronisation : par rendez-vous bloquant.

Références

[1] Nancy A. Lynch et Mark R. Tuttle. Hierarchical correctness proofs for distributed algorithms. *Proc. of the 6th Annual ACM Symposium on Principles of Distributed Computing, PODC'87*, pages 137–151, Vancouver, BC, Canada, August 1987.

[2] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, et Pei-Hsin Ho. Hybrid automata : An algorithmic approach to the specification and verification of hybrid systems. *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 1993.

[3] T.A. Henzinger, P.H. Ho, et H. Wong-Toi. Hytech : a model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1) :110–122, 1997.

[4] Nancy A. Lynch, Roberto Segala, et Frits W. Vaandrager. Hybrid i/o automata. *Inf. Comput.*, 185(1) :105–157, 2003.

[5] Goran Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations*. PhD thesis, Radboud Universiteit Nijmegen, October 10 2005.

[6] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[7] Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[8] M. Raush et H.-M. Hanisch. Net condition/event systems with multiple condition outputs. *Emerging Technologies and Factory Automation, ETFA'95*, volume 1, pages 592–600, Paris, France, October 1995.

[9] Pavel Krcál et Wang Yi. Communicating timed automata : The more synchronous, the more difficult to verify. *CAV*, pages 249–262, 2006.

[10] ISO (International Organization of Standardization). *Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour*, 1989.

[11] Kim Guldstrand Larsen, Paul Pettersson, et Wang Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2) :134–152, 1997.

[12] Christos G. Cassandras et Stéphane Lafortune. *Introduction to Discrete Event Systems*, chapter 2, pages 83–91. Kluwer Academic Publishers, 1999.

[13] IEC (International Electrotechnical Commission). *IEC Standard 61499 : Function blocks for industrial-process measurement and control systems*, 2004.