Identification des états équivalents dans l'approche modale

Gregory FARAUT — Laurent PIÉTRAC — Eric NIEL

Laboratoire Ampère, INSA-Lyon
Bât. St-Exupery, 20 av Albert Einstein
69621 Villeurbanne, France
prenom.nom@insa-lyon.fr

RÉSUMÉ. *Ce papier présente une extension de la fonction de suivi de trajectoire pour identifier les états compatibles entre modes. La fonction de suivi de trajectoire a pour but d'assurer que les commutations entre modes soient sûres. Utilisée dans une démarche d'aide à la conception se basant sur la Théorie de Contrôle par Supervision (TCS), les spécifications sont formellement assurées. Cependant, les modèles manipulés sont des automates à états émondés et il a été mise en évidence que certains états non-accessibles, donc supprimés pour obtenir un automate émondé, étaient accessibles depuis un autre mode par une commutation. L'extension que nous proposons s'appuie sur les automates non-émondés et propose une comparaison formelle sur les noms des états pour déterminer les états accessibles depuis d'autres modes.*

ABSTRACT. *This paper presents an extension of the process tracking in order to identify the compatible states among modes. The aim of the process tracking is to ensure the switching between modes is reliable. Used in a framework based on Supervisory Control Theory (SCT), the specifications then are formally ensured. However, the handle models are trimmed automata and then some inaccessibles states, removed by the trim computing, could be accessible from another mode by a switch event. The proposed extension of the process tracking uses the non-trimmed automata and a formal comparison on the name of states in order to identify the accessible states from other modes*

MOTS-CLÉS : *Systèmes à Événements Discrets (SED), Théorie de contrôle par supervision (TCS), approche modale, aide à la conception, approches formelles*

KEYWORDS: *Discrete Events Systems (DES), Supervisory Control Theory (SCT), modal approach, assisted design, formal approaches*

1. Introduction

Introduite par Ramadge et Wonham (Ramadge *et al.*, 1987; Ramadge *et al.*, 1989), la Théorie de Contrôle par Supervision (TCS) est utilisée dans le domaine des systèmes à évènements discrets (SED). Le cadre théorique des SED permet l'étude et la conception d'une variété importante d'applications industrielles qu'elles soient de production, de transport ou informatiques. L'objectif de la TCS est de fournir un cadre formel permettant la synthèse de lois de commande du système étudié au travers de propriétés fondamentales telles que la contrôlabilité, l'atteignabilité ou l'observabilité. Notamment dans le cas de la propriété de contrôlabilité, la TCS permet, en s'appuyant sur les langages, de déterminer le plus grand sous-langage qui respecte les spécifications. Ce langage est appelé "le suprême contrôlable" (Wonham *et al.*, 1987). Ainsi, dans le pire des cas - celui où le suprême contrôlable serait vide - la TCS prouve formellement qu'aucune solution n'existe pour que le système respecte les spécifications en toutes circonstances.

Si l'un des avantages de la TCS est de disposer d'outils formels de conception, la théorie implique deux inconvénients majeurs. Le premier inconvénient est celui de l'explosion combinatoire lors du passage à l'échelle de systèmes complexes. Le second problème est celui de l'interprétation des modèles même dans le cas où l'explosion combinatoire est maîtrisée.

Pour réduire l'impact de l'explosion combinatoire sur les modèles manipulés, il a été proposé différentes approches qui permettent de décomposer les modèles tout en utilisant la TCS. Bien que ces approches soient appropriées pour réduire la complexité des modèles, leur utilisation implique également la vérification de propriétés supplémentaires pour assurer le calcul du suprême contrôlable. Ces propriétés sont, par exemple, la propriété de non-blocage entre superviseurs pour l'approche modulaire (Wonham *et al.*, 1988; Komenda *et al.*, 2008) ou la propriété de consistance hiérarchique pour l'approche hiérarchique (Zhong *et al.*, 1990; Leduc *et al.*, 2005). Ces approches réduisent l'explosion combinatoire pour le calcul mais n'améliorent pas l'interprétation des modèles car elles continuent de manipuler le procédé entier et toutes les spécifications en même temps.

Dans nos travaux précédents (Faraut *et al.*, 2009b; Faraut *et al.*, 2009a), nous avons proposé une démarche d'aide à la conception, basée sur l'approche modale, qui prend en compte plusieurs modes de fonctionnement, avec un modèle par mode. Cela permet de manipuler des modèles plus petits que dans l'approche centralisée et de garder ainsi une bonne interprétation de ceux-ci. L'utilisation de la TCS dans cette approche permet également de prouver formellement que les modèles construits respectent toutes les spécifications.

Néanmoins, bien que l'approche modale améliore significativement l'interprétation des modèles, il reste difficile de vérifier formellement que le langage obtenu corresponde au suprême contrôlable équivalent à l'approche centralisée. En effet, dans la TCS, ce langage est bien maximal car il prend en compte toutes les trajectoires possibles et supprime uniquement les trajectoires interdites. Lorsque les modèles sont

décomposés, quelle que soit l'approche utilisée, des trajectoires qui existaient dans l'approche centralisée sont supprimées lors du calcul du suprême contrôlable. Ceci vient de la dernière étape du calcul du suprême contrôlable qui supprime les états qui ne sont ni accessibles, ni co-accessibles. Autrement dit, après avoir supprimé les trajectoires interdites, le calcul du suprême contrôlable enlève les états dans lesquels il n'est plus possible que le système évolue. Si cela n'a aucune influence dans l'approche centralisée, cela réduit les trajectoires possibles entre modèles lorsque plusieurs modèles représentent le comportement du système, car des états inaccessibles dans un mode, peuvent l'être à partir d'un autre mode. Ceci n'a, à notre connaissance, jamais été pris en compte dans les approches basées sur une décomposition de modèles.

Dans ce papier, nous proposons une extension de la fonction de suivi de trajectoire, quatrième étape de notre démarche d'aide à la conception. Le rôle de cette extension est de permettre d'identifier les états compatibles entre modes, c'est-à-dire ceux qui sont à première vue inaccessibles mais qui sont néanmoins accessibles d'un autre mode. Pour réaliser cette identification, nous proposons d'une part de nous appuyer sur les automates à états non-émondés, ce qui permet de garder les états inaccessibles dans les modèles ; d'autre part, car il n'est pas possible d'utiliser les langages pour ces états, nous proposons d'utiliser les noms de ceux-ci pour en établir des équivalences. Ainsi, nous continuons à maîtriser la taille des modèles, nous gardons une bonne interprétation de ceux-ci, et nous identifions des trajectoires de commutations qui étaient précédemment supprimées lors du calcul des suprêmes contrôlables.

La section 2 concerne la TCS et contient deux parties consacrées respectivement à l'approche centralisée et à notre démarche d'aide à la conception pour l'approche modale. Nous rappelons dans cette section les principales équations de construction des modèles. Dans la section 3, nous présentons notre contribution, à savoir la définition des états équivalents et la nouvelle procédure de suivi de trajectoire qui identifie tous les états compatibles. Enfin, nous illustrons en section 4 notre contribution sur un exemple.

2. Théorie de contrôle par supervision

2.1. *Approche centralisée*

La TCS repose sur l'exploitation de la modélisation formelle d'un SED par un procédé générateur d'évènements (automate G) bouclé sur un superviseur S (défini par une fonction). Ce système bouclé est représenté par un procédé sous contrôle H générateur d'évènements autorisés.

L'automate G est un 5-uplets tel que $G = \{Q, \Sigma, \delta, q_0, Q_m\}$, avec Q l'ensemble des états du système, Σ l'ensemble des évènements, δ représente la fonction de transition et est de la forme $\delta : Q \times \Sigma \rightarrow Q$, $q_0 \in Q$ est l'état initial et $Q_m \subseteq Q$ est le sous-ensemble des états marqués. Les états marqués représentent des états particuliers devant être atteignables. Le langage du procédé est noté $L(G)$ et est formellement défini par $L(G) = \{s \in \Sigma^* | \delta(q_o, s) \text{ est définie}\}$. De même, un langage

marqué contenant tous les mots conduisant à un des états marqués est défini par : $L_m(G) = \{s \in \Sigma^* | \delta(q_o, s) \in Q_m\}$.

Le superviseur S est une fonction telle que $S : L(G) \rightarrow 2^{\Sigma}$. Il résulte à partir du langage du procédé G de tous les ensembles des commutations possibles dans chaque état du système. Il n'existe pas qu'un seul S pour un langage de G donné, et chaque couple (G,S), représentant le procédé sous contrôle S/G, équivaut à un langage $L(S/G)$ sous-langage de $L(G)$. Pour obtenir ce sous-langage, le superviseur désactive dans $L(G)$ un certain nombre d'évènements afin de supprimer certaines trajectoires. Cependant, cette action ne peut être entreprise que si les évènements sont contrôlables. L'ensemble Σ est ainsi décomposé en deux sous-ensembles disjoints Σ_c et Σ_{uc} représentant respectivement l'ensemble des évènements contrôlables et l'ensemble des évènements non contrôlables.

Pour le système bouclé, l'objectif est de vérifier que le procédé sous contrôle, représenté par un automate $H = \{Y, \Sigma, \tau, y_0, Y_m\}$ est tel que le langage marqué de H soit inclus dans celui de G, et qu'il existe un superviseur S tel que $L(S/G) = L(H)$. H est obtenu par le produit (Cassandras *et al.*, 2007) du procédé G et des spécifications E. Ces spécifications, résultantes des exigences clients, représentent les contraintes de sécurité et de vivacité du système. Le modèle des spécifications E est également un automate tel que $E = \{X, \Sigma, \xi, x_0, X_m\}$. Le procédé sous contrôle construit, il reste à vérifier sa contrôlabilité en discutant de l'existence d'un superviseur S. Si celui-ci existe, alors H respecte les spécifications E. Dans le cas contraire, les contraintes ne sont plus garanties, et implémenter H en l'état peut aboutir à des comportements non désirés.

Dans une telle situation, le concepteur recherche le plus grand sous-langage contrôlable de $L(S/G)$, c.-à-d. qui assure l'existence d'un superviseur restreignant le comportement de G à $L(H)$. Ce plus grand sous-langage est appelé "suprême contrôlable" et l'opération est représentée $\uparrow c$. Cette fonction donnant le plus grand sous-langage contrôlable vérifie également les propriétés d'accessibilité et de co-accessibilité (Wonham *et al.*, 1987), pour la vérification de non-blocage. Le langage du procédé sous contrôle $L(H)$ est obtenu tel que $L(H) = [L(S/G)]^{\uparrow c}$. Dans le cas où $L(S/G)$ est contrôlable ($L(H) = L(S/G)$), la recherche du plus grand sous-langage contrôlable est directe $L(H) = L(S/G) = [L(S/G)]^{\uparrow c}$, et tel que ce procédé sous contrôle soit non-bloquant ($L(S/G) = \overline{L_m(S/G)}$.

2.2. *Approche modale*

Dans ce papier, nous utilisons une approche modale comme précédemment dans nos travaux (Faraut *et al.*, 2009b; Faraut *et al.*, 2009a). En effet, nous considérons qu'un système est composé par de nombreux composants (actionneurs, capteurs, etc.) et doit respecter différentes exigences exprimées dans le cahier des charges. Ainsi, un système doit utiliser une partie des composants présents pour accomplir certaines tâches tout en respectant un ensemble d'exigences. Nous considérons ceci comme un

mode du système. Un système possède alors plusieurs modes et il commute entre eux suivant les besoins ou contraintes. Le comportement dans un mode représente un comportement *temporaire* du système jusqu'à ce qu'un autre mode soit activé. L'objectif de l'approche modale est de calculer le procédé sous contrôle de chaque mode tel que la séquence d'activation des modes représente le comportement permanent du système et est équivalent au comportement généré par l'approche centralisée.

2.3. *Démarche d'aide à la conception*

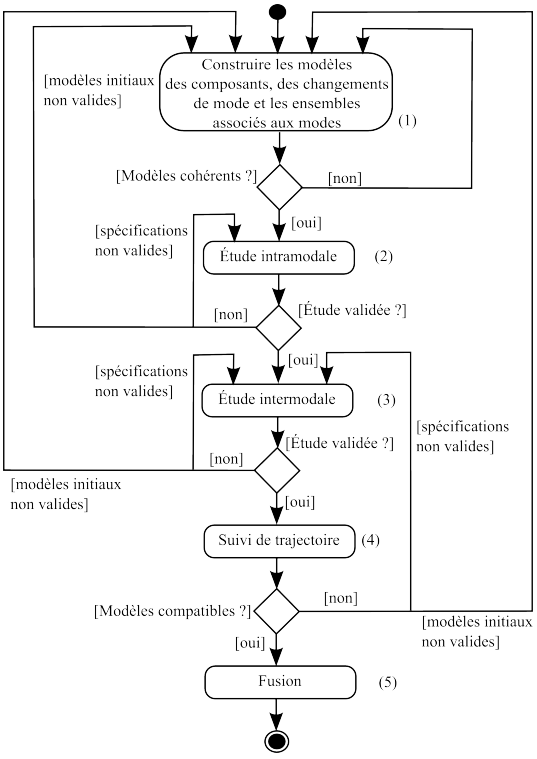

Figure 1. *Démarche d'aide à la conception pour l'approche modale*

Notre approche, illustrée figure 1, repose sur l'utilisation de la TCS et sur une démarche qui propose plusieurs étapes de construction des modèles. Notre démarche comporte cinq étapes de construction issues de nos travaux (Faraut *et al.*, 2009b; Faraut *et al.*, 2009a). La première regroupe la construction des différents modèles de composants et de spécifications. L'étape 2 concerne l'étude du comportement du procédé sous contrôle dans chaque mode, sans tenir compte des commutations de mode. Dans l'étape 3, un enrichissement des modèles avec les composants et les spécifications de commutation est réalisé, ce qui permet ensuite dans l'étape 4 d'étudier des

occurrences des évènements de commutation, et de vérifier que les changements de modes sont possibles. Enfin, dans la dernière étape, les modèles obtenus sont réduits et font apparaître un état représentant l'inactivité du mode.

Dans ce papier, nous nous intéressons principalement aux étapes 3 et 4. Néanmoins, nous souhaitons formellement exprimer les modèles manipulés. Ainsi, nous commençons par définir l'ensemble des composants utilisés dans un système, et les modèles qui les représentent :

Définition 1 (Ensemble de composants)
L'ensemble des composants est appelé $\mathcal{C} = \{C_1, C_2, \ldots, C_i\}$*, où* $i \in \mathbb{N}$ *et* $i \geq 1$*. Un composant* C_i *est modélisé par un automate* $G^{C_i} = (Q^{C_i}, \Sigma^{C_i}, \delta^{C_i}, q_0^{C_i}, Q_m^{C_i})$*, avec :*

– Q^{C_i}*,* $Q_m^{C_i}$ *et* $q_0^{C_i}$ *sont respectivement l'ensemble des états, l'ensemble des états marqués et l'état initial du composant* C_i *;*

– Σ^{C_i} *est l'ensemble des évènements du composant* C_i*, et inclut quatre sous-ensembles :*

- $\Sigma^{C_i} = \Sigma_c^{C_i} \cup \Sigma_{uc}^{C_i}$ *avec* $\Sigma_c^{C_i} \cap \Sigma_{uc}^{C_i} = \emptyset$*.* Σ_c *et* Σ_{uc} *sont respectivement les sous-ensembles disjoints des évènements contrôlables et incontrôlables du composant* C_i *;*

- $\Sigma^{C_i} = \Sigma_{\rightleftarrows}^{C_i} \cup \Sigma_{\circlearrowleft}^{C_i}$ *avec* $\Sigma_{\rightleftarrows}^{C_i} \cap \Sigma_{\circlearrowleft}^{C_i} = \emptyset$*.* $\Sigma_{\rightleftarrows}^{C_i}$ *est l'ensemble des évènements de commutation de* C_i*.* $\Sigma_{\circlearrowleft}^{C_i}$ *sont les autres évènements du composant.*

– δ^{C_i} *est la fonction de transition. Elle inclut* $\delta_{\rightleftarrows}^{C_i}$ *qui représente la fonction de transition de commutation. Une transition de commutation est une transition dans laquelle l'évènement est inclus dans* $\Sigma_{\rightleftarrows}^{C_i}$*.* ♦

La seconde définition concerne les ensembles des exigences :

Définition 2 (Ensemble d'exigences)
L'ensemble des exigences est appelé $\mathcal{R} = \{R_1, R_2, \ldots, R_l\}$*, où* $l \in \mathbb{N}$ *et* $l \geq 1$*. Une exigence* R_l*, est modélisée par une spécification* E^{R_l} *représentée par un automate* $E^{R_l} = (X^{R_l}, \Sigma^{R_l}, \xi^{R_l}, x_0^{R_l}, X_m^{R_l})$*, avec* X_l*,* $X_m^{R_l}$ *et* $x_0^{R_l}$ *qui sont respectivement l'ensemble des états, l'ensemble des états marqués et l'état initial de la spécification* E^{R_l} *;* Σ^{R_l} *est l'ensemble des évènements de la spécification* E^{R_l} *et* ξ^{R_l} *est la fonction de transition.*♦

De ces ensembles, nous sommes capables de définir les notions de mode :

Définition 3 (Ensemble de modes)
Un ensemble de modes est défini par $\mathcal{M} = \{M_0, M_1, \ldots, M_j\}$*, où* M_j *est le nom du mode représenté.* ♦

Définition 4 (Un mode)
En considérant que $\mathcal{C}^{M_j} \subset \mathcal{C}$ *soit l'ensemble des composants utilisés dans le mode* M_j*, et que* $\mathcal{R}^{M_j} \subset \mathcal{R}$ *soit l'ensemble des exigences à respecter dans le mode* M_j*, alors le mode* M_j *représente le comportement temporaire du système tel qu'il utilise seulement, et pour un temps fini, un sous-ensemble des composants du système et doit respecter un sous-ensemble des exigences :* $M_j = (\mathcal{C}^{M_j}, \mathcal{R}^{M_j})$*.* ♦

2.4. *Les procédés sous contrôle* H^{M_j}

À partir des définitions ci-dessus, et de l'étape 3 de la démarche d'aide à la conception illustrée figure 1, il est possible de construire les procédés sous contrôle dans chaque mode. Formellement, cette construction, basée sur la TCS, est définie comme suit :

Définition 5 (Procédé sous contrôle H^{M_j})
En considérant que $G^{M_j} = [||_{C_k \in \mathcal{C}^{M_j}} G^{C_k}]$, *et que* $E^{M_j} = [||_{R_k \in \mathcal{R}^{M_j}} E^{R_k}]$, *alors :*
$H^{M_j} = (Y^{M_j}, \Sigma^{M_j}, \tau^{M_j}, y_0^{M_j}, Y_m^{M_j})$ *avec :* $L_m(H^{M_j}) = [L_m(G^{M_j} \times E^{M_j})]^{\uparrow c}$ ♦

Cette définition utilise deux opérations sur les automates. La première est la composition parallèle dont la définition peut être trouvée dans (Cassandras *et al.*, 2007).

La seconde opération est l'algorithme de construction du suprême contrôlable, noté $\uparrow c$. Il existe actuellement, et à notre connaissance, seulement deux algorithmes qui permettent de le calculer (Wonham *et al.*, 1987; Kumar *et al.*, 1991). Dans ces deux algorithmes, la dernière étape est identique et concerne le calcul de l'automate émondé. Pour rappel, après avoir supprimé les trajectoires interdites, et incontrôlables, les algorithmes, réduisent la taille du modèle en supprimant les états non-accessibles. Dans la démarche d'aide à la conception, la fonction de suivi de trajectoire, correspondant à l'étape 4, est initialement exécutée sur les modèles construits par ces algorithmes.

2.5. *Suivi de trajectoire*

Les modèles H^{M_j} respectent ainsi les spécifications liées aux comportements internes désirés dans chaque mode et aux commutations permises. Cependant, à cause de la décomposition du comportement permanent du système en plusieurs comportements temporaires, représentés par des modes avec un modèle par mode, la commutation d'un mode à un autre n'est pas sûre. En effet, bien que respectant les spécifications dans un mode, le système, avec les modèles actuels, peut très bien sortir d'un mode et ne rentrer dans aucun mode. Il en résulte la désactivation d'un mode, l'activation d'aucun autre mode, et le blocage du système.

Il faut donc s'assurer que pour chaque commutation sortante d'un mode corresponde bien à *une et une seule* commutation entrante dans un autre mode. C'est le but de la fonction de suivi de trajectoire.

Pour réaliser cela, nous avons proposé une suite de définitions qui permettent de caractériser formellement le type de trajectoire menant à une commutation. Les trajectoires sont caractérisées en termes de *consistance* et de *compatibilité*.

Les trajectoires inconsistantes sont celles qui mènent à une commutation qui désactive un mode, mais qui pourraient activer plusieurs modes, au lieu d'un seul. Ceci est dû à une perte d'information lors de la commutation. Cette perte d'informa-

tion résulte de la fonction de projection, utilisée dans le suivi de trajectoire, qui a pour but d'identifier les trajectoires équivalentes entre modèles. Cependant, cette fonction *efface* les évènements non-communs entre modèles. Dans certains cas, cette information perdue peut causer de l'indéterminisme d'activation et doit être ajoutée.

Dans ce papier, nous nous intéressons principalement aux trajectoires qui sont *incompatibles*. Ces trajectoires sont celles qui mènent à une commutation qui désactive un mode et n'en active aucun autre. La raison principale est que la trajectoire équivalente dans le mode d'arrivée a été supprimée lors du calcul du suprême contrôlable pour respecter les spécifications.

Dans les définitions suivantes, nous considérons que le système se trouve dans le mode M_j et que l'évènement de commutation α se produit et conduit à commuter vers le mode M_k.

La première définition concerne les ensembles d'évènements de commutation dans les modes :

Définition 6 (Ensembles d'évènements de commutation)
$\Sigma^{M_i}_{\leftrightarrows} \subset \Sigma^{M_i}$ *est l'ensemble des évènements de commutation du mode* M_i *tel que :* $\Sigma^{M_i}_{\leftrightarrows} = \bigcup_{n \in \mathcal{C}^{M_i}_{\leftrightarrows}} \Sigma^{\mathcal{C}_n}_{\leftrightarrows}$.
$\Sigma^{M_i}_{\leftrightarrows} = \Sigma^{M_i}_{\leftarrow} \cup \Sigma^{M_i}_{\rightarrow}$ *où* $\Sigma^{M_i}_{\leftarrow}$ *(resp.* $\Sigma^{M_i}_{\rightarrow}$*) est l'ensemble des évènements qui activent (resp. désactivent) le mode* M_i*. Ils sont définis tels que :*

- $\Sigma^{M_i}_{\rightarrow} = \{\alpha \in \Sigma^{M_i}_{\leftrightarrows} | M_k \in Q^{\mathcal{M}}, \delta^{\mathcal{M}}(M_i, \alpha) = M_k \textit{ est défini}\}$;
- $\Sigma^{M_i}_{\leftarrow} = \{\alpha \in \Sigma^{M_i}_{\leftrightarrows} | M_k \in Q^{\mathcal{M}}, \delta^{\mathcal{M}}(M_k, \alpha) = M_i \textit{ est défini}\}$; ♦

De cette définition, nous pouvons définir l'ensemble d'états $Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}$ appartenant à H^{M_j} où un évènement de commutation α existe et implique une commutation du mode de départ M_j vers le mode d'arrivée M_k.

Définition 7 (État de commutation)
$Y^{M_j}_{M_j \xrightarrow{\alpha} M_k} = \{y \in Y^{M_j} \mid M_j, M_k \in Q^{\mathcal{M}},\ \alpha \in \Sigma^{M_i}_{\rightarrow} \cap \Sigma^{M_k}_{\leftarrow},\ \tau^{M_j}(y, \alpha) \textit{ est défini}\}$ ♦

Nous pouvons également définir le langage $L^{y}_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})$, un sous-langage de $L(H^{M_j})$, qui contient tous les mots menant de l'état initial y_0 de H^{M_j} vers un état y où un évènement de commutation α peut se produire.

Définition 8 (Langage de commutation)
$L^{y}_{M_j \xrightarrow{\alpha} M_k}(H^{M_j}) = \{s \in \Sigma^{M_j *} \mid y \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k},\ \tau(y_0, s) = y \textit{ est défini}\}$ ♦

Ce langage correspond à toutes les dynamiques dans le mode M_j qui mènent à un état y où un des évènements de commutation étudiés peut être généré. Il importe maintenant de savoir si ces dynamiques existent dans le mode d'arrivée M_k. Pour cela, nous utilisons la fonction de projection étendue définie par :

Définition 9 (Projection étendue)
Considérons $P_{M_j,M_k} : \Sigma^{M_j*} \rightarrow \Sigma^{M_k*}$ *tel que* $\forall \sigma \in \Sigma^{M_j}$ *et* $\forall s \in \Sigma^{M_j*}$ *:*

$$P_{M_j,M_k}(\varepsilon) = \varepsilon$$
$$P_{M_j,M_k}(s\sigma) = \begin{cases} P_{M_j,M_k}(s)\sigma \text{ si } \sigma \in \Sigma^{M_j} \cap \Sigma^{M_k} \\ P_{M_j,M_k}(s) \quad \text{si } \sigma \in \Sigma^{M_j} \backslash \Sigma^{M_k} \end{cases}$$

Littéralement, la fonction de projection étendue prend un langage défini sur l'alphabet utilisé dans le mode M_j (l'alphabet dans le mode de départ), et *efface* les évènements qui ne sont pas inclus dans l'alphabet utilisé dans le mode (d'arrivée) M_k.

Cependant, avoir identifié ces dynamiques par projection ne signifie pas que ces dynamiques existent dans le mode d'arrivée. La définition suivante recherche donc si les dynamiques existent dans le langage généré par le système dans le mode M_k.

Définition 10 (Langage équivalent)
En considérant que le mode de départ soit le mode M_j *et que le mode d'arrivée soit le mode* M_k, $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})$ *est le sous-langage de* H^{M_j} *menant à l'état y où un évènement de commutation* α *peut être généré.* $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k})$ *est le langage projeté sur l'alphabet du mode d'arrivée* M_k*, tel que :* $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) = P_{M_j,M_k}[L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})]$. ♦

En utilisant cette dernière définition, nous sommes en mesure de définir la propriété de compatibilité qui vérifie que si tous les langages $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k})$ sont inclus dans $L(H^{M_k})$, alors *au moins* un état de connexion existe dans $L(H^{M_k})$ pour chaque état y.

Définition 11 (Compatibilité entre langages)
Une trajectoire de commutation dans H^{M_k} *est compatible avec une trajectoire de commutation dans* H^{M_j} *ssi :* $\forall y \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}$, $(L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \subseteq L(H^{M_k}))$ ♦

Une trajectoire est consistante si chaque évènement de commutation dans le mode M_j correspond à un unique évènement de commutation dans le mode M_k.

Définition 12 (Consistance entre langages)
En considérant que les trajectoires soient compatibles entre les procédés sous contrôle H^{M_k} *et* H^{M_j}*. Les langages sont consistants ssi :*

$$\forall y_1, y_2 \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k},\ (y_1 \neq y_2 \Leftrightarrow L^{y_1}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \cap L^{y_2}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) = \emptyset) \qquad \blacklozenge$$

En résumé, les trajectoires qui sont consistantes sont également compatibles. En revanche, les trajectoires compatibles ne sont pas forcément consistantes. Quand le concepteur est face à des trajectoires compatibles mais inconsistantes, il ne peut que rajouter des spécifications afin de supprimer la perte d'information impliquant une inconsistance. C'était également le cas pour les trajectoires incompatibles. Dans ce papier, nous proposons une résolution des trajectoires incompatibles sans l'ajout de nouvelles spécifications.

3. Procédure et définitions

Nous proposons dans ce papier d'identifier les états compatibles entre modes dans le cas de trajectoires incompatibles. En effet, dans nos travaux précédents, les trajectoires incompatibles et inconsistantes devaient être supprimées par l'ajout de nouvelles spécifications. Cependant, il a été mis en évidence que la suppression des trajectoires incompatibles vient des spécifications. Une trajectoire permise et menant à une commutation dans un mode M_j n'existe plus dans le mode M_k car elle a été supprimée. Ainsi, la commutation et l'état où est générée cette commutation ont pu être supprimés uniquement car l'état était devenu inaccessible. En travaillant sur les automates non-émondés, nous maintenons les états inaccessibles dans l'automate. Mais comme il est impossible d'identifier la commutation par une trajectoire, nous proposons de définir les états équivalents entre modes suivant leur nom.

3.1. *Notions sur les suites*

Dans la TCS, l'opération élémentaire utilisée est celle de la composition parallèle. Dans cette opération, le nom de l'automate construit par composition parallèle est une *suite* des noms des automates qui le composent (Cassandras *et al.*, 2007). Dans le but de manipuler formellement le nom des automates, et pour être en mesure d'établir des équivalences, nous avons besoin de rappeler les notions sur les suites (Sipser, 2005).

Une suite d'éléments est une liste de ces éléments dans un ordre donné. Nous désignons une suite en écrivant la liste entre parenthèse. Par exemple, une suite d'éléments de l'ensemble $\{q_1, q_2, q_3\}$ peut être écrite : (q_3, q_1, q_2, q_1). Dans un ensemble, l'ordre des éléments n'a aucune importance, alors qu'il est important dans une suite. De même, la répétition a une importance dans une suite, mais pas dans un ensemble. Deux suites peuvent être concaténées : $(q_1, q_2) \cdot (q_3, q_4) = (q_1, q_2, q_3, q_4)$. De plus, une suite de suites renvoie mathématiquement une suite : $((q_1, q_2), (q_3, q_4), (q_5))$ = $(q_1, q_2, q_3, q_4, q_5)$. Ainsi, comme écrit dans la définition de la composition parallèle, le produit de deux ensembles Q_1 et Q_2, écrit $Q_1 \times Q_2$, est l'ensemble de toutes les paires dans lesquelles le premier élément appartient à l'ensemble Q_1 et le second élément appartient à l'ensemble Q_2 : $Q_1 \times Q_2 = \{(q_{10}, q_{20}), (q_{11}, q_{20}), \ldots, (q_i, q_j)\}$ avec $q_i \in Q_1$ et $q_j \in Q_2$. Dans le domaine des suites, la fonction *ran* permet de transformer une suite en un ensemble : $ran(q_1, q_2, q_3) = \{q_1, q_2, q_3\}$. Enfin, l'opérateur $\restriction$ permet de *filtrer* une suite par un ensemble dans le but de supprimer les éléments de la suite qui n'appartiennent pas à l'ensemble sur lequel le filtrage est effectué : $(q_1, q_2, q_3, q_4) \restriction \{q_2, q_4, q_5\} = (q_2, q_4)$. Le résultat de l'opération de filtrage est une sous-suite de la suite originale.

3.2. *Définition d'états équivalents*

Les états étudiés sont ceux qui sont inaccessibles dans le mode M_k. Cette inaccessibilité est identifiée par les trajectoires incompatibles. Nous considérons que deux états

- chacun dans un mode différent - sont équivalents si les composants et les spécifications communs aux deux modes sont dans les mêmes états. Comme nous considérons que le nom des états, dans le modèle H^{M_j}, est construit à partir du nom des états de chaque composant et de chaque spécification dans le mode M_j, alors pour établir l'équivalence entre deux états nous ne considérons qu'une partie du nom, celle qui correspond aux composants et aux spécifications communs aux deux modes. Formellement, nous définissons l'équivalence d'états comme suit :

Définition 13 (États équivalents)
En considérant que H_j^M et H_k^M soient respectivement les procédés sous contrôle dans le mode M_j et dans le mode M_k, en considérant que y_1 et y_2 soient deux états où peut être généré un évènement de commutation α, en considérant que $\mathcal{C}^{M_{jk}}$ soit l'ensemble des composants communs aux modes M_j et M_k, et en considérant que $\mathcal{R}^{M_{jk}}$ soit l'ensemble des spécifications qui doivent être respectées dans les modes M_j et M_k, alors les états y_1 et y_2 sont équivalents si :

$$\forall y_1 \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k} \text{ et } y_2 \in Y^{M_k}_{M_j \xrightarrow{\alpha} M_k},\ \mathcal{C}^{M_{jk}} = \mathcal{C}^{M_j} \cap \mathcal{C}^{M_k} \text{ et } \mathcal{R}^{M_{jk}} = \mathcal{R}^{M_j} \cap \mathcal{R}^{M_k},$$

$$ran(y_1 \upharpoonright (Q_{eq} \cup X_{eq})) = ran(y_2 \upharpoonright (Q_{eq} \cup X_{eq})) \text{ avec : } Q_{eq} = \bigcup_{C_i \in \mathcal{C}^{M_{jk}}} Q^{C_i} \text{ et } X_{eq} = \bigcup_{R_l \in \mathcal{R}^{M_{jk}}} X^{R_l}$$ ♦

3.3. *Procédure de suivi de trajectoire*

Avec les propriétés ci-dessus, nous sommes en mesure de détailler la procédure qui identifie les couples d'états équivalents qui connectent le mode de départ M_j au mode d'arrivée M_k.

Pour chaque évènement de commutation α dans H^{M_j} impliquant une désactivation du mode M_j et menant à une activation du mode M_k et pour chaque $y \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}$ [définition 7] :
Nous calculons $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})$ [définition 8] ;
Nous calculons $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k})$ [définition 10] ;
Nous vérifions la propriété de compatibilité [définition 11] :

1) pour chaque $(L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \subseteq L(H^{M_k})) : H^{M_k}$ la propriété de consistance est vérifiée [définition 12] :

a) $\exists y_1, y_2 \in Y^{M_k}$ $[L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j}) \subseteq L^{y_1}(H^{M_k})$ et $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j}) \subseteq L^{y_2}(H^{M_k})]$ ou $\exists y' \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}$ $(y \neq y' \Leftrightarrow L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \cap L^{y'}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \neq \emptyset)$: la trajectoire n'est pas consistante et la procédure s'arrête ici ;

b) sinon, la trajectoire de commutation est considérée comme consistante. $\delta^{\mathcal{M}}(M_j,\alpha) = M_k$ est donc considérée comme valide.

2) pour chaque $(L^{y}_{M_j \overset{\alpha}{\to} M_k}(H^{M_k}) \not\subseteq L(H^{M_k}))$: la propriété d'états équivalents est vérifiée [définition 13] :

a) un ensemble d'états potentiellement équivalents est déterminé afin que la propriété d'états équivalents soit satisfaite,

b) le concepteur choisit l'état équivalent dans cet ensemble. Si aucun état de l'ensemble n'est l'état équivalent recherché, la procédure s'arrête ici ;

3) Après le dernier $y \in Y^{M_j}_{M_j \overset{\alpha}{\to} M_k}$, si la procédure s'est déroulée complètement sans avoir été interrompue pour incompatibilité ou inconsistance, alors nous pouvons dire que H^{M_k} est consistant avec H^{M_j} au regard de l'évènement de commutation α. Nous pouvons étudier la commutation suivante.

4) l'automate final est non-émondé et tous les états équivalents ont été identifiés. Nous procédons au calcul de l'automate émondé tel qu'il ne supprime pas les états accessibles depuis un autre mode : (a) $\forall y \in Y^{M_j}_{M_j \overset{\alpha}{\to} M_k}$ sont considérés comme des états initiaux ; (b) nous calculons l'automate émondé ; (c) nous ne considérons plus les états y tel que $\forall y \in Y^{M_j}_{M_j \overset{\alpha}{\to} M_k}$ comme des états initiaux, sauf pour l'état initial $y_0^{M_j}$

À la fin de cette procédure, les modèles respectent les propriétés décrites. Les états inaccessibles dans le mode considéré mais accessibles depuis un autre mode seront ensuite supprimés lors de la fusion d'états, qui est la prochaine, et la dernière, étape de notre démarche d'aide à la conception.

4. Exemple applicatif

Dans cette section, un exemple est présenté pour illustrer la différence de comportement entre des automates émondés et non-émondés, ainsi que les états inaccessibles mais équivalents à d'autres états, autrement dit, accessible depuis un autre mode.

4.1. *Système et exigences*

L'exemple présenté est un système, illustré sur la figure 2(a). Ce système comporte quatre composants, dont les modèles sont représentés sur les figures 2(b) pour le composant C_1 et 2(c) pour les autres. Dans ces modèles, les évènements contrôlables s_i représentent des débuts de tâches, alors que les évènements incontrôlables e_i représentent des fins de tâches. Ces composants sont reliés à un stock noté B. Ce système a deux modes ; un mode nominal N et un mode dégradé D tel que $\mathcal{M} = \{N, D\}$. Dans le mode N, le système n'utilise que les composants C_1, C_2 et C_4, alors que dans le mode D il utilise C_2 et C_3. En effet, le composant C_1 peut tomber en panne, symbolisée par l'évènement f_1. Suite à cette panne, le composant C_1 est dans un état de

faute F_1 en attente d'une réparation symbolisée par l'évènement r_1 qui l'amène dans son état initial. L'évènement de commutation f_1 provoque la commutation du mode N vers le mode D et l'évènement de commutation r_1 provoque la commutation du mode D vers le mode N. Ainsi, le composant C_3 est en redondance de C_1. De plus, dans le mode D, le composant C_4 n'est pas utilisé pour produire. Cette spécification de commutation, et d'activation/désactivation des composants C_3 et C_4 suivant l'état de fonctionnement du composant C_1, est représentée sur la figure 2(d). Le stock B a une capacité maximale de 1 ; cette spécification est représentée sur la figure 2(e). Enfin, comme nous nous intéressons qu'aux trajectoires de commutations incompatibles, les spécifications représentées sur les figures 2(f) et 2(g) ont pour but de supprimer les trajectoires inconsistantes. Plus d'informations sur ces spécifications peuvent être trouvées dans (Faraut *et al.*, 2009b).

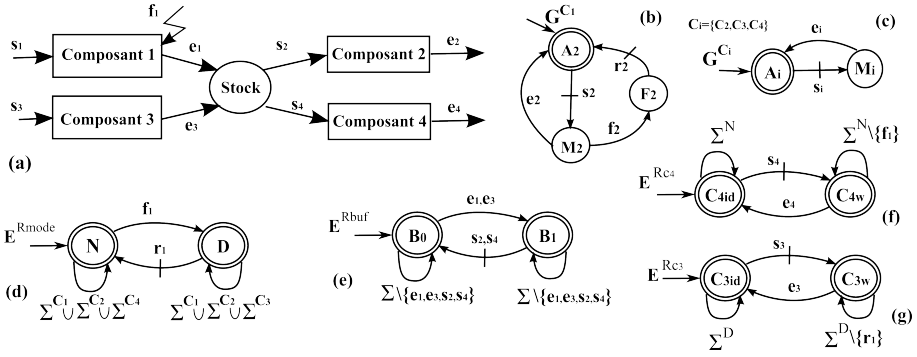

Figure 2. *Exemple : (a) le système étudié ; (b,c) modèles de composants C_i ; (d) modèle de la spécification de mode ; (e) modèle de al spécification du stock ; (f) modèle de la spécification sur C_4 ; (g) modèle de la spécification sur C_3*

4.2. *Suivi de trajectoire*

À partir des ensembles de modes et de spécifications, nous construisons les procédés sous contrôle dans chaque mode. La figure 3(a) représente le procédé sous contrôle dans le mode N et la figure 3(b) le procédé sous contrôle dans le mode D. Les parties en pointillées sont celles qui auraient été supprimées lors du calcul du suprême contrôlable. Comme nous travaillons sur des automates non-émondés, nous utilisons tous les comportements des automates. Ainsi, sur la figure 3(b), des trajectoires mènent aux états $(F_1, A_2, A_3, D, B_1, C_{3id})$ et $(F_1, M_2, A_3, D, B_1, C_{3id})$ où l'évènement de commutation r_1 peut se produire. Ces trajectoires sont incompatibles car elles n'appartiennent pas à $L(H^N)$, représenté sur la figure 3(a). En effectuant le suivi de trajectoire proposé dans cet article, avec notamment la définition 13, nous recherchons les états compatibles dans H^N qui permettraient la commutation depuis ces trajectoires. Ainsi nous avons, pour chacun des états où se produit une commutation incompa-

tible :$(F_1,A_2,A_3,D,B_1,C_{3id}) \upharpoonright (Q^{C_1} \cup Q^{C_2} \cup X^{R_{buf}} \cup X^{R_{mode}}) = (F_1,A_2,D,B_1)$ qui est équivalent, dans H^N, à l'état : $(F_1,A_2,A_4,D,B_1,C_{4id}) \upharpoonright (Q^{C_1} \cup Q^{C_2} \cup X^{R_{buf}} \cup X^{R_{mode}}) = (F_1,A_2,D,B_1)$.

Ces deux états sont donc équivalents et, même s'ils sont inaccessibles dans le mode N, sont accessibles depuis le mode D. Ainsi, en utilisant les automates non-émondés, une équivalence sur le nom des états entre modèles, et en ayant étendu le suivi de trajectoire avec cette équivalence, nous sommes en mesure d'identifier de nouvelles trajectoires possibles, précédemment supprimées.

5. Conclusion et perspectives

Dans cet article, nous discutons des trajectoires incompatibles entre modèles lorsque nous utilisons l'approche modale et la Théorie de Contrôle par Supervision. Cette incompatibilité est due à la décomposition du système en modes. Travailler sur des modèles plus petits réduit le comportement possible du système dans le mode considéré et mène, par conséquent, lors du calcul du suprême contrôlable, à supprimer les trajectoires inaccessibles qui pourraient cependant être accessibles depuis d'autres modes. Pour éviter l'ajout de spécifications dans le but de supprimer ces trajectoires incompatibles, nous proposons dans cet article de travailler sur des automates non-émondés et d'utiliser les noms des états pour identifier les états équivalents entre modèle. Il en résulte un plus grand nombre de commutations possibles.

Les perspectives de développement envisagées actuellement se rapportent à l'extension de cette équivalence d'état à toute la fonction de suivi de trajectoire dans le but d'automatiser par un programme la recherche de trajectoires. Enfin, travailler formellement sur le nom des états ouvre la voie vers la description de nouvelles spécifications exprimées sur les états et non uniquement sur les langages.

6. Bibliographie

Cassandras C. G., Lafortune S., *Introduction to discrete event systems [Second Edition]*, Springer, 2007.

Faraut G., Piétrac L., Niel E., « Démarche d'aide à la conception par approche multimode des SED », *Journal Européen des Systèmes Automatisés*, vol. 43/7-9, p. 837-853, 2009a.

Faraut G., Pietrac L., Niel E., « Formal Approach to Multimodal Control Design : Application to Mode Switching », *Industrial Informatics, IEEE Transactions on*, vol. 5, n° 4, p. 443-453, Nov., 2009b.

Komenda J., van Schuppen J., Gaudin B., Marchand H., « Supervisory control of modular systems with global specification languages », *Automatica*, vol. 44, n° 4, p. 1127-1134, April, 2008.

Kumar R., Garg V. K., Marcus S. I., « On Controllability and Normality of Discrete Event Dynamical Systems », *Systems and Control Letters*, vol. 17, n° 3, p. 157-168, 1991.

Leduc R., Lawford M., Wonham W., « Hierarchical interface-based supervisory control-part II : parallel case », *Automatic Control, IEEE Transactions on*, vol. 50, n° 9, p. 1336-1348, Sept., 2005.

Ramadge P. J., Wonham W. M., « Supervisory control of a class of discrete event processes », *SIAM J. Control Optim.*, vol. 25, n° 1, p. 206-230, 1987.

Ramadge P. J., Wonham W. M., « The control of discrete event systems », *Proceedings of the IEEE*, vol. 77, n° 1, p. 81-98, Jan, 1989.

Sipser M., *Introduction to the Theory of Computation, Second Edition*, Course Technology, February, 2005.

Wonham W. M., Ramadge P. J., « On the supremal controllable sublanguage of a given language », *SIAM Journal of Control and Optimization*, vol. 25, n° 3, p. 637-659, 1987.

Wonham W. M., Ramadge P. J., « Modular Supervisory Control of Discrete Event Systems », *Mathematics of Control, Signals and Systems*, vol. 1, n° 1, p. 13-30, 1988.

Zhong H., Wonham W., « On the consistency of hierarchical supervision in discrete-event systems », *Automatic Control, IEEE Transactions on*, vol. 35, n° 10, p. 1125-1134, Oct, 1990.

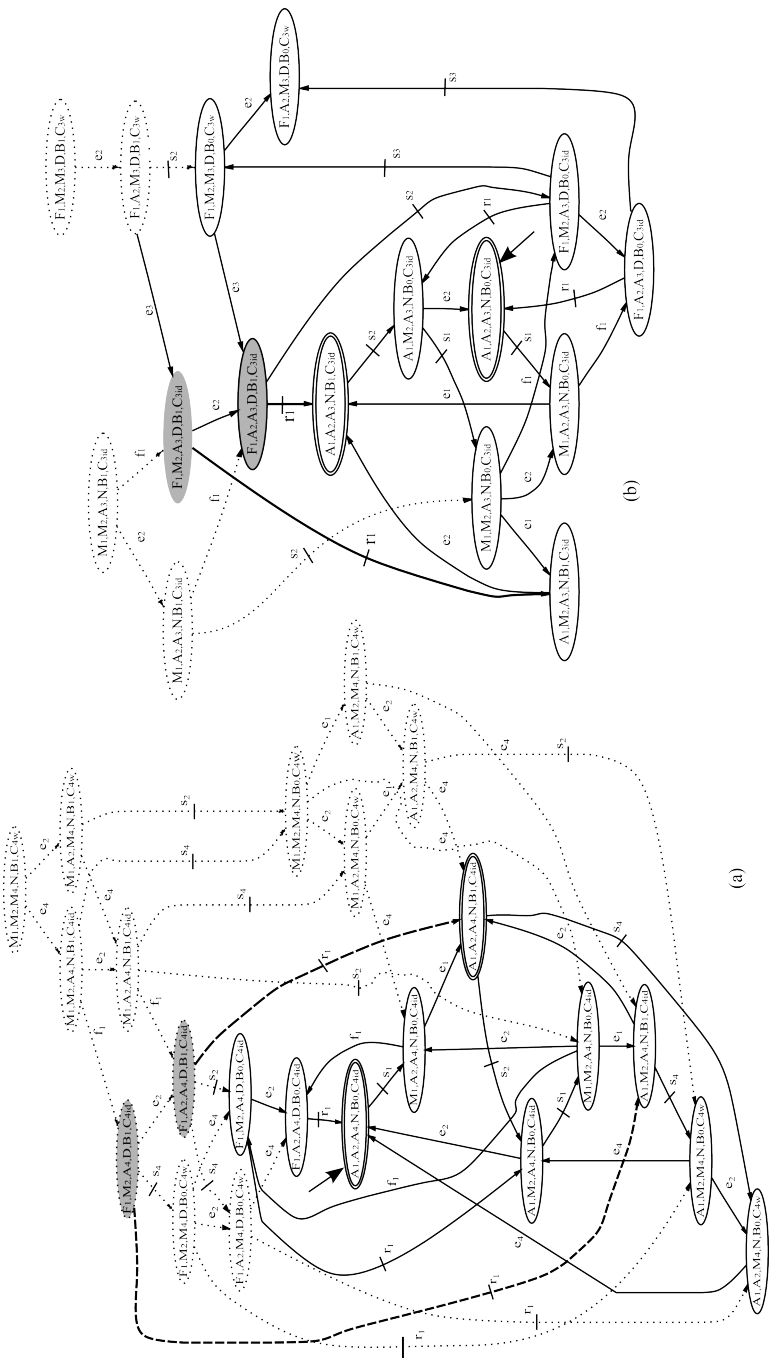

Figure 3. *(a) Procédé sous contrôle H^N ;(b) Procédé sous contrôle H^D. Les lignes pointillées représentent les comportements inaccessibles qui seraient supprimés par le suprême contrôlable. Les états grisés sont les états compatibles permettant de commuter du mode dégradé H^D vers le mode nominal H^N sur l'occurrence de l'évènement de commutation r_1. Les lignes discontinues représentent ces évènements de commutations.*