UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

CONTRÔLE ET DIAGNOSTIC DÉCENTRALISÉS DES SYSTÈMES À ÉVÉNEMENTS DISCRETS: APPROCHE MULTI-DÉCISIONNELLE

Thèse de doctorat
Spécialité : génie électrique

Hicham CHAKIB

Jury : Ahmed KHOUMSI (directeur)
Mustapha NOURELFATH (examinateur externe)
Brahim HADJOU (rapporteur)
Richard ST-DENIS (évaluateur externe au programme)

Sherbrooke (Québec) Canada Mai 2011

IV 2146

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-83280-6

Our file Notre référence
ISBN: 978-0-494-83280-6

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distrbute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protege cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

RÉSUMÉ

De nos jours, les systèmes technologiques sont devenus très complexes (matériel informatique, logiciel, système de télécommunication, usine manufacturière, etc), et cette complexité croît continuellement de sorte que les anciennes techniques intuitives utilisées pour leur conception, leur étude et leur réalisation deviennent inadaptées. À cause de cette complexité croissante, la probabilité pour qu'une erreur (ou panne) inattendue survienne est de plus en plus grande. Plus encore, quelques erreurs peuvent provoquer des accidents très graves causant des pertes économiques ou humaines. C'est dans ce cadre que les méthodes formelles ont été développées pour l'analyse, la conception et la réalisation des systèmes logiciels et électroniques quelque soit leur complexité. Ainsi, l'étude des *systèmes à événements discrets* (SED) a été introduite avec l'objectif de développer des méthodes formelles pour répondre à des besoins pressants, tels que le contrôle, le diagnostic, le pronostic, le test et la vérification des comportements discrets des systèmes technologiques.

Cette thèse considère et généralise les études du contrôle et du diagnostic décentralisés des SED. Le principe commun du contrôle et du diagnostic décentralisés des SED est la *prise de décision décentralisée*, qui est basée sur l'utilisation d'une *architecture décentralisée*. Cette dernière est constituée de plusieurs *décideurs locaux* qui observent partiellement un SED et prennent des décisions locales qui sont ensuite fusionnées par un module de fusion D. Ce dernier, en se basant sur une fonction de fusion, calcule à partir des décisions locales une décision globale. Le système englobant les décideurs locaux et le module de fusion s'appelle un *décideur décentralisé*. L'ensemble de tous les décideurs décentralisés ayant D comme module de fusion est appelé D-*architecture*.

La principale contribution de cette thèse est de proposer une nouvelle approche de prise de décision décentralisée, appelée *multi-décision* et qualifiée de *multi-décisionnelle*. Le principe de la multi-décision est basé sur l'utilisation de plusieurs (disons p) décideurs décentralisés $(DD^j)_{j=1,\dots,p}$ qui fonctionnent simultanément et en parallèle. Chaque DD^j a une architecture décentralisée parmi celles qu'on trouve dans la littérature. C'est-à-dire que chaque DD^j est constitué d'un ensemble de décideurs locaux $(Dec_i^j)_{i=1,\dots,n}$ dont les décisions locales sont fusionnées par un module de fusion D^j afin d'obtenir une décision globale. Dans l'architecture multi-décisionnelle, les décisions globales des p $(DD^j)_{j=1,\dots,p}$ sont fusionnées par un module $\mathbf{D}$ afin d'obtenir une décision effective qui respecte une propriété désirée Pr. L'intérêt de la multi-décision est que l'architecture $((DD^j)_{j=1,\dots,p}, \mathbf{D})$ constituée des différents $(DD^j)_{j=1,\dots,p}$ et de $\mathbf{D}$ généralise chacune des architectures DD^j. C'est-à-dire que l'ensemble des SED auxquels on peut appliquer $((DD^j)_{j=1,\dots,p}, \mathbf{D})$ englobe les différents SED auxquels on peut appliquer les différents DD^j séparément.

Nous avons étudié l'approche multi-décisionnelle sur deux exemples de prise de décision : le contrôle supervisé et le diagnostic. On obtient alors le contrôle et le diagnostic multi-décisionnels. Dans les deux cas, l'approche multi-décisionnelle nécessite une décomposition de langages infinis (c.à.d., contenant un nombre infini de séquences), qui est connue comme étant un problème difficile. Pour résoudre ce problème, on a proposé, dans le cas particulier

des langages réguliers, une méthode qui transforme la décomposition d'un langage infini X en une décomposition d'un ensemble fini d'états marqués. Pour arriver à cela, on a dû s'imposer une restriction en ne considérant que les décompositions de X qui respectent une condition spécifique. Cette condition présente l'avantage de rendre les conditions d'existence de solutions vérifiables. Nous avons ainsi développé des algorithmes pour vérifier les conditions d'existence de solutions pour le contrôle et le diagnostic multi-décisionnels. Ces algorithmes ont le même ordre de complexité que les algorithmes qui vérifient les conditions d'existence de solutions pour le contrôle et le diagnostic décentralisés. Il est important de noter que les conditions d'existence obtenues pour une architecture multi-décisionnelle $((DD^j)_{j=1,\ldots,p}, \mathbf{D})$ sont moins contraignantes que celles obtenues pour chacune des architectures DD^j.

Mots-clés : systèmes à événements discrets (SED), automates à états finis, prise de décision décentralisée sur des SED, contrôle supervisé décentralisé de SED, diagnostic décentralisé de SED, prise de décision multi-décisionnelle sur des SED, contrôle supervisé multi-décisionnel de SED, diagnostic multi-décisionnel de SED.

REMERCIEMENTS

J'exprime toute ma reconnaissance à Monsieur Ahmed Khoumsi, Professeur à l'Université de Sherbrooke, mon directeur de thèse pour tout ce qu'il m'a apporté tant sur le plan scientifique que personnel, et notamment pour son enseignement, ses conseils, et sa disponibilité.

Je tiens à remercier très chaleureusement les membres du jury pour l'intérêt qu'ils ont porté à mes recherches en acceptant d'évaluer cette thèse. Je remercie Monsieur Mustapha NOURELFATH, Professeur à l'Université Laval, qui a accepté d'être évaluateur externe à l'université de ce travail, Monsieur Brahim HADJOU, Professeur à l'Université de Sherbrooke, pour avoir accepté d'être rapporteur de cette thèse, et Monsieur Richard ST-DENIS, Professeur à l'Université de Sherbrooke, d'avoir accepté d'être évaluateur externe au programme de ce travail.

Je voudrais également remercier mes collègues Amer AL-CANAAN et Abdelhamid MAMMERI pour leur amitié, leur aide et pour nos longues discussions intéressantes que nous avons eues ensemble.

Je remercie tous les membres du Département de génie électrique et de génie informatique. Plus particulièrement, je remercie vivement Mesdames Claudia CARBONNEAU et Danielle GAGNÉ du secrétariat du Département de génie électrique et de génie informatique pour leur aide, leur sympathie et leur disponibilité. Je tiens également à remercier Monsieur Sylvain PÉPIN du groupe de support en génie informatique pour sa disponibilité, et son aide précieuses.

Enfin, je tiens à exprimer toute ma reconnaissance à ma famille pour leur encouragement et leur soutien moral qu'ils m'ont apportés.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1

INTRODUCTION

Le sujet de la présente thèse porte sur le contrôle supervisé décentralisé et le diagnostic décentralisé des systèmes à événements discrets modélisés par des automates à états finis. Nous avons opté pour une thèse par articles. Dans cette introduction, nous présentons dans un premier temps les théories du contrôle supervisé et du diagnostic décentralisés des systèmes à événements discrets. Dans un deuxième temps, nous formulons la problématique de la thèse et ses objectifs. Enfin, nous présentons les résultats obtenus et nos contributions.

1.1 Systèmes à événements discrets (SED)

Le comportement des systèmes à événements discrets (SED) est caractérisé par une dynamique événementielle, dans le sens où le comportement du système présente une succession d'événements qui se manifestent pendant des intervalles de temps qui ne sont pas nécessairement réguliers. L'occurrence d'un événement est soit instantanée (pas de durée) ou un événement est interprété comme une action d'une certaine durée (courte ou longue).

Chaque événement se manifeste pendant un laps de temps très court pour signaler un changement d'état du système. Un événement se définit comme une action produite par le système ou par son environnement et subie par le système.

L'étude formelle des SED s'est avérée nécessaire avec la croissance de l'industrialisation, et avec l'apparition des systèmes informatiques de plus en plus complexes. En effet, c'est dans le but d'étudier les interactions des systèmes informatiques complexes, que l'étude formelle des SED a pris forme [Hoare, 1985; Milner, 1980]. La modélisation et l'étude des comportements discrets sont souvent effectuées par la théorie des langages, les machines à états finies, les réseaux de Petri et l'algèbre de processus. La modélisation peut être effectuée au niveau logique (en considérant seulement l'ordre logique des événements), au niveau temporel (en introduisant le temps), ou au niveau stochastique (en introduisant les probabilités).

Comme exemple de SED on peut citer :

Un protocole de communication, avec comme exemples d'événements : émission d'un message (par exemple demande de connexion), réception d'un message (par exemple

message d'acceptation de connexion) et demande de service. Nous pouvons citer comme exemple de séquence d'événements (comportement) : "émission message m"."émission message n"."réception message n"."réception message m".

Un système de téléphonie, avec comme exemples d'événements : décroche, raccroche, début de sonnerie et compose un chiffre. Nous pouvons citer comme exemple de séquence d'événements : "début de sonnerie"."décroche"."raccroche".

Une ligne d'assemblage d'un système de production, avec comme exemples d'événements : "pièce A prête", "pièce B prête", "détection de la présence d'une pièce par un capteur" et "assemblage des deux pièces A et B". Nous pouvons citer comme exemple de séquence d'événements : "pièce A prête"."pièce B prête"."détection de la présence des pièces A et B"."assemblage des pièces A et B".

L'étude formelle d'un SED est souvent élaborée en utilisant les *automates à états finis* (AEF) et les *réseaux de Petri*. Dans cette thèse, nous optons pour les AEF. Un AEF est un graphe de transitions permettant de modéliser un SED par des états (états du SED) et des transitions correspondant aux occurrences des événements. Formellement, un AEF G est défini comme un quintuple $G = (Q, \Sigma, \delta, q_0, Q_m)$, avec :

Q : ensemble fini d'états ;

Σ : ensemble fini d'événements (ou *alphabet*) ;

q_0 : état *initial* ;

Q_m : ensemble des états *finaux* ou *marqués* ($Q_m \subseteq Q$) ;

$\delta : Q \times \Sigma \rightarrow Q$: fonction de transition entre états.

À chaque état de G, la fonction de transition δ définit les événements dont l'occurrence est possible et l'état atteint suite à l'occurrence de chacun de ces événements. Ainsi, à partir de l'état initial, on peut parcourir une séquence d'états de l'AEF qui dépend de la séquence d'événements exécutés. L'ensemble de toutes les *séquences* (ou *traces*) d'événements exécutées par un AEF G est appelé *langage* de G, que l'on note $\mathcal{L}(G) \subseteq \Sigma^*$, où Σ^* est l'ensemble des mots sur Σ, mot vide ϵ inclus. L'ensemble des séquences qui atteignent les états marqués est appelé *langage marqué* et noté $\mathcal{L}_m(G)$. On a $\mathcal{L}_m(G) \subseteq \mathcal{L}(G)$.

Comme exemple, considérons le SED modélisé par l'AEF de la figure 1.1, qui contient trois états : au repos (R), en marche (M) et en panne (P). Les transitions entre ces états sont indiquées par les événements suivants : mettre en marche (m), finir le travail (f), tomber en panne (p) et être réparé (r). Au départ la machine est au repos (état R), la transition entre cet état et l'état où la machine entre en fonction (M) est indiqué par l'événement m. Si la machine finit son travail, elle retourne à l'état de repos R, ce qui est indiqué par l'événement f. Si par contre, la machine tombe en panne, alors elle se trouvera dans l'état

P par la manifestation de l'événement p. Si la machine est réparée, alors elle passera de l'état P à l'état R par la manifestation de l'événement r.

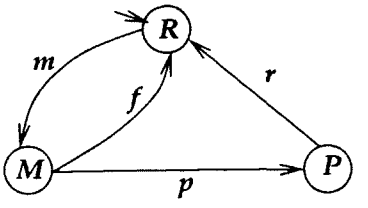

Figure 1.1 Exemple d'un SED modélisé par un AEF. "R" est l'état initial et $Q_m = \{R\}$.

1.2 Contrôle et diagnostic des SED

1.2.1 Prise de décision dans les SED

La prise de décisions est un sujet très important qu'on retrouve dans plusieurs domaines. Nous nous intéressons ici à la prise de décision dans les SED qui peut se résumer comme suit. On considère un SED qu'on appellera *procédé* qui évolue par l'exécution d'événements, et un module qu'on appellera *décideur* dont la tâche est d'observer l'évolution du procédé et de prendre des décisions respectant une propriété désirée Pr. Selon les exemples, les décisions prises par le décideur peuvent influencer ou pas le comportement du procédé. Voici trois exemples de prises de décisions :

Dans le contrôle [Cassandras et Lafortune, 1999; Kumar et Garg, 1995; Wonham, 2008] : le décideur est un superviseur, dont les décisions consistent à *inhiber* ou *autoriser* des événements contrôlables. La propriété désirée Pr habituellement utilisée est que le procédé contrôlé soit conforme à une spécification donnée, c.à.d., il ne doit exécuter que des séquences d'événements autorisées par la spécification.

Dans le diagnostic [Sampath *et al.*, 1995] : le décideur est un diagnostiqueur dont les décisions consistent à indiquer l'occurrence ou la non occurrence d'un comportement indésirable (par exemple, une panne ou une faute) du procédé. Un exemple de propriété Pr désirée est que toute faute exécutée par le procédé soit indiquée dans un délai borné (noter qu'il y a d'autres propriétés qui seront discutées dans cette introduction).

Dans le pronostic [Kumar et Takai, 2008] : le décideur est un pronostiqueur dont les décisions consistent à prédire l'occurrence ou la non-occurrence d'un comportement indésirable du procédé. Un exemple de propriété Pr désirée est que toute faute soit prédite.

Le décideur tente de respecter la propriété désirée à partir de l'information qu'il obtient en observant le comportement du procédé. Cette observation peut être *totale* (le décideur observe *tous* les événements exécutés par le procédé) ou *partielle* (seulement une partie des événements est observée). L'observation partielle est habituellement modélisée par un masque qui ne laisse passer que les événements observables (voir schéma de la figure 1.2).

Dans des récents travaux [Kumar et Takai, 2006, 2009; Takai et Kumar, 2006, 2008; Wang *et al.*, 2004, 2005, 2007], il a été montré que la technique de prise de décision utilisée dans le contrôle des SED peut être adaptée pour le diagnostic des SED. Dans [Wang *et al.*, 2004, 2005, 2007], les auteurs ont étudié le diagnostic des SED en considérant plusieurs architectures qui ont été utilisées dans le cadre du contrôle des SED [Yoo et Lafortune, 2002a, 2004]. Dans [Kumar et Takai, 2006, 2009; Takai et Kumar, 2006, 2008], les auteurs ont étudié le diagnostic des SED par le principe de l'inférence qui a été déjà étudié dans le contrôle [Kumar et Takai, 2005, 2007]. Dans ces différentes références, la même technique de prise de décision a été utilisée dans le contrôle et dans le diagnostic.

Notons que dans notre thèse, nous ne manquerons pas d'exploiter judicieusement cette analogie entre le contrôle et le diagnostic, ce qui nous a permis d'obtenir des résultats intéressants dans ces deux sujets. En effet, l'approche suivie dans cette thèse permettra de présenter le contrôle et le diagnostic dans des cadres formels assez similaires.

Après avoir expliqué que le contrôle et le diagnostic peuvent être vus comme deux cas particuliers de prises de décisions, nous allons dans les deux sections suivantes (1.2.2 et 1.2.3) présenter plus spécifiquement le contrôle et le diagnostic.

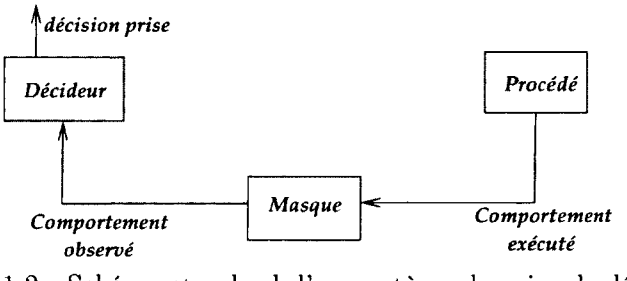

Figure 1.2 Schéma standard d'un système de prise de décision

1.2.2 Contrôle supervisé des SED

Principe général du contrôle

La théorie du contrôle s'intéresse principalement à l'analyse et à la conception de systèmes de contrôle. Contrôler un système veut dire influencer son comportement pour atteindre un

but désiré. De nos jours, les systèmes de contrôles sont partout, on les trouve par exemple dans les microprocesseurs, les robots et les avions. Le système à contrôler, aussi appelé *procédé*, exécute une certaine fonction qui à partir de signaux d'entrée génère des signaux de sortie. Le comportement externe du procédé est ainsi défini par des signaux de sortie en réponse à des signaux d'entrée. En général, le comportement d'un procédé consiste en une partie désirable et une partie indésirable. Dans ce cas, il est nécessaire de restreindre son comportement afin de n'exécuter que la partie désirable, parfois appelée *spécification*. Le problème du contrôle consiste à concevoir une loi de contrôle appropriée pour choisir les signaux d'entrée de telle sorte que le comportement engendré par le procédé sous contrôle (comportement contrôlé) satisfait la spécification. Le contrôle peut s'effectuer dans quatre principaux domaines, à savoir : (1) domaine temporel, (2) domaine en s ou transformée de Laplace, (3) domaine des fréquences et (4) domaine d'événements discrets. Pour plus de détails sur le contrôle dans les différents domaines, voir [Dorf et Bishop, 2001; Franklin *et al.*, 1994; Wonham, 2008]. Dans cette thèse, on s'intéresse au quatrième domaine qu'on appelle aussi le contrôle supervisé des SED.

Contrôle supervisé des SED

La théorie du contrôle supervisé (ou plus simplement contrôle) des SED a été élaborée par Ramadge et Wonham en utilisant la théorie des langages et les AEF [Ramadge et Wonham, 1987; Wonham et Ramadge, 1987] (pour plus de détails voir [Cassandras et Lafortune, 1999; Kumar et Garg, 1995; Wonham, 2008]). En résumé, cette théorie est basée sur le principe suivant. On dispose d'un SED à contrôler, que nous appellerons *procédé*, modélisé par un AEF G. Ce dernier décrit le comportement sans contrainte du procédé. Soient $\mathcal{L}(G)$ le langage préfixe-clos et $\mathcal{L}_m(G)$ le langage marqué associés à G. On utilisera le terme G aussi bien pour l'AEF lui-même que pour le procédé qu'il modélise. L'ensemble Σ des événements de G est divisé en deux sous-ensembles : Σ_c, ensemble des événements *contrôlables* et Σ_{uc}, ensemble des événements *incontrôlables*, tels que $\Sigma = \Sigma_c \cup \Sigma_{uc}$ et $\Sigma_c \cap \Sigma_{uc} = \emptyset$.

Le principe de la théorie du contrôle des SED est de contraindre (ou de contrôler) G afin qu'il respecte une *spécification* modélisée par un AEF S. Soient $\mathcal{L}(S)$ le langage préfixe-clos et $\mathcal{L}_m(S)$ le langage marqué associés à S. Le contrôle est accompli par un superviseur Sup modélisé par un AEF (ce dernier étant aussi désigné par Sup) qui, en interagissant avec G, le contraint à n'exécuter que les comportements acceptés par S. Le calcul de la décision d'un superviseur d'autoriser ou d'inhiber un événement $\sigma \in \Sigma_c$ est basé sur deux ensembles $\mathcal{E}_\sigma$ et $\mathcal{D}_\sigma$. Considérons une spécification S de langage marqué $K = \mathcal{L}_m(S) \subseteq \mathcal{L}_m(G)$. On a alors $\overline{K} = \mathcal{L}(S) \subseteq \mathcal{L}(G)$, où $\overline{K} = \{s \in \Sigma^* | \exists u \in \Sigma^* \text{ t.q. } su \in K\}$ est le préfixe clôture

de K. $\mathcal{E}_\sigma$ est l'ensemble des traces $s \in \overline{K}$ telles que $s\sigma$ est une trace du procédé G (c.à.d., $s\sigma \in \mathcal{L}(G)$) autorisée par la spécification (c.à.d., $s\sigma \in \overline{K}$). Et $\mathcal{D}_\sigma$ est l'ensemble des traces $s \in \overline{K}$ telles que $s\sigma$ est une trace du procédé G (c.à.d., $s\sigma \in \mathcal{L}(G)$) non autorisée par la spécification (c.à.d., $s\sigma \notin \overline{K}$). Après chaque séquence s d'événements exécutés par G, *Sup* autorise tout événement $\sigma \in \Sigma_c$ accepté par $\mathcal{L}(S)$ (c.à.d., σ tel que $s \in \mathcal{E}_\sigma$) et inhibe tout événement $\sigma \in \Sigma_c$ accepté par $\mathcal{L}(G)$ et non accepté par $\mathcal{L}(S)$ (c.à.d., σ tel que $s \in \mathcal{D}_\sigma$). Les événements incontrôlables sont toujours autorisés. Le contrôle effectué par *Sup* sur G a pour but de restreindre le comportement de G par l'interdiction d'événements contrôlables afin que G respecte S. Le système englobant le procédé G et le superviseur *Sup* forment un nouveau système, noté Sup/G. Soit $\mathcal{L}(Sup/G)$ le langage préfixe-clos et $\mathcal{L}_m(Sup/G)$ le langage marqué associés à Sup/G. Les objectifs de contrôle sont que Sup/G doit satisfaire les deux objectifs suivants :

Sup/G doit être *non bloquant*, c.à.d., qu'il doit toujours pouvoir atteindre un état marqué à partir de n'importe quel état de Sup/G. Formellement, $\overline{\mathcal{L}_m(Sup/G)} = \mathcal{L}(Sup/G)$.

Sup/G doit toujours respecter la spécification S, c.à.d., $\mathcal{L}_m(Sup/G) \subseteq \mathcal{L}_m(S)$.

Une spécification S (ou $\mathcal{L}_m(S)$) est dite *contrôlable* par rapport à G et Σ_{uc} (ou contrôlable, si le contexte est clair) si $\mathcal{L}(S).\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \mathcal{L}(S)$, c.à.d., $\mathcal{L}(S)$ peut être obtenu à partir de G en inhibant seulement les événements contrôlables. Une spécification S (ou $\mathcal{L}_m(S)$) est dite $\mathcal{L}_m(G)$-*close* si $\mathcal{L}(S) \cap \mathcal{L}_m(G) = \mathcal{L}_m(S)$, c.à.d., toute séquence s acceptée par S (c.à.d., $s \in \mathcal{L}(S)$) et atteignant un état marqué de G (c.à.d., $s \in \mathcal{L}_m(G)$) atteint aussi un état marqué de S (c.à.d., $s \in \mathcal{L}_m(S)$).

Dans [Ramadge et Wonham, 1987; Wonham et Ramadge, 1987], il a été prouvé que si S est contrôlable et $\mathcal{L}_m(G)$-clos, alors il existe un superviseur non bloquant *Sup* tel que $\mathcal{L}_m(Sup/G) = \mathcal{L}_m(S)$.

Dans le cas où S n'est pas contrôlable ou n'est pas $\mathcal{L}_m(G)$-clos, le problème qui se pose dans la théorie du contrôle des SED est d'obtenir un *Sup* non bloquant qui soit le plus permissif possible, à savoir qu'il permet d'obtenir le comportement le moins restrictif parmi tous les comportements contrôlables et $\mathcal{L}_m(G)$-clos de G qui respectent S (c.à.d., $\mathcal{L}_m(Sup/G) \subseteq \mathcal{L}_m(S)$. Comme $\mathcal{L}_m(G)$-clôture est fermée par l'union, et en montrant que la contrôlabilité est fermée par l'union, les auteurs dans [Ramadge et Wonham, 1987] ont montré l'existence et l'unicité du comportement le moins restrictif. Ils ont proposé un algorithme à point fixe permettant de construire le comportement le moins restrictif à partir de G et S.

La théorie développée par Ramadge et Wonham a donné lieu à diverses extensions. Parmi ces extensions, on trouve le *contrôle à observation partielle*, le *contrôle modulaire*, le

contrôle hiérarchique et le *contrôle décentralisé.* Dans cette thèse, on s'intéressera plus spécialement au contrôle décentralisé. Le contrôle à observation partielle a été intensément étudié par les chercheurs dans le domaine du contrôle des SED, et son étude présente un préliminaire important pour le contrôle décentralisé. On parlera dans la prochaine sous-section de ce type de contrôle. Le contrôle décentralisé sera présenté dans la section 1.3.2.

Contrôle à observation partielle des SED

Ce type de contrôle des SED se présente dans les situations où le superviseur doit prendre des décisions sans avoir accès à toutes les informations requises. C'est le cas de plusieurs systèmes, comme les systèmes répartis, où le superviseur n'a accès au système qu'à travers un ou plusieurs sites locaux, ce qui ne permet au superviseur d'avoir accès qu'à des informations partielles. Et quelques fois, ce sont des facteurs qui surgissent pendant le contrôle (par exemple une panne d'un capteur du SED) qui peuvent rendre quelques informations inaccessibles pour le superviseur. C'est dans ce contexte que la notion d'*observation partielle* intervient pour décrire la situation où le superviseur est incapable d'observer certains événements au cours du contrôle du SED [Cieslak *et al.*, 1988; Lin et Wonham, 1988b]. Par conséquent, l'alphabet Σ du SED se répartit en deux sous-ensembles, celui des événements observables Σ_o et celui des événements inobservables Σ_{uo}, tels que $\Sigma = \Sigma_o \cup \Sigma_{uo}$ et $\Sigma_o \cap \Sigma_{uo} = \emptyset$. Il s'agit alors de concevoir un superviseur *Sup* qui permet d'atteindre les deux objectifs déjà présentés (Sup/G non bloquant et $\mathcal{L}_m(Sup/G) \subseteq \mathcal{L}_m(S)$). Mais le problème qui se pose, c'est que *Sup* doit atteindre ces objectifs alors qu'il a une observabilité seulement partielle du procédé. Ce problème est ainsi appelé le *problème d'observation du contrôle supervisé avec tolérance zéro* (POCSTZ). Un schéma de contrôle en boucle fermée sous observation partielle résolvant le POCSTZ est représenté à la figure 1.3, où le masque P correspond à une *projection naturelle* $P : \Sigma \rightarrow \Sigma_o$ qui ne laisse passer que les événements de Σ_o, c.à.d., pour chaque séquence s, $P(s)$ est obtenue à partir de s en excluant les événements qui n'appartiennent pas à Σ_o (c.à.d., les événements de $\Sigma_{uo} = \Sigma \backslash \Sigma_o$). Toute séquence s sera donc observée par le superviseur sous la forme de sa projection dans Σ_o, $P(s)$. Deux séquences s_1 et s_2 qui ont la même projection dans Σ_o (c.à.d., $P(s_1) = P(s_2)$) ne seront pas distinguées par le superviseur. Par conséquent, les mêmes décisions d'inhibitions d'événements sont prises par *Sup* après les exécutions de s_1 ou s_2. Une condition sur la spécification est donc nécessaire pour tenir compte de cette contrainte sur *Sup*. C'est ainsi que la notion du *langage observable* est introduite pour décrire un langage K qui peut être obtenu en effectuant sur G des décisions d'autorisation/inhibition d'événements telles que lorsque deux séquences s_1 et s_2 de $\overline{K}$ ont la même projection ($P(s_1) = P(s_2)$),

alors les mêmes décisions sont prises après s_1 et s_2. Formellement, on dit qu'un langage K est *observable* par rapport à G et P (ou *observable*, si le contexte est clair) si et seulement si pour toutes séquences $s_1, s_2 \in \overline{K}$ telles que $P(s_1) = P(s_2)$, on a

$\forall \sigma \in \Sigma,\ s_1\sigma \in \overline{K} \wedge s_2\sigma \in \mathcal{L}(G) \Rightarrow s_2\sigma \in \overline{K}$, et

$s_1 \in K \ \wedge \ s_2 \in \overline{K} \cap \mathcal{L}_m(G) \ \Rightarrow \ s_2 \in K$, et

les deux propriétés ci-dessus sont aussi vraies si on intervertit s_1 et s_2.

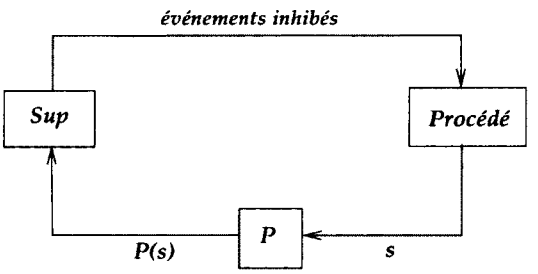

Figure 1.3 Action du superviseur en boucle fermée sous observation partielle

Les auteurs dans [Lin et Wonham, 1988a] ont démontré que le POCSTZ peut être résolu si et seulement si la spécification K est contrôlable par rapport à G et Σ_{uc}, observable par rapport à G et P et $\mathcal{L}_m$-close, mais aucune méthode de calcul de superviseur n'est proposée. Le problème qui se pose, c'est que l'observabilité n'est pas fermée par l'union, l'union de deux langages observables n'étant pas nécessairement observable. Une solution partielle a été proposée en suggérant la notion de *normalité* [Lin et Wonham, 1990]. Étant donné la projection $P : \Sigma \rightarrow \Sigma_o$, on dit qu'un langage $K \subseteq \mathcal{L}(G)$ est normal si $\overline{K} = \mathcal{L}(G) \cap P^{-1}(P(\overline{K}))$. Intuitivement, K est dit normal s'il est le plus grand langage parmi les langages qui ont la même projection que K. Vu que la normalité implique l'observabilité (alors que la réciproque n'est pas toujours vraie) [Brandt *et al.*, 1990; Lin et Wonham, 1988a] et que la normalité est fermée par l'union, on peut trouver pour une spécification K qui n'est pas normale, le plus grand comportement inclus dans K qui soit normal et donc observable. Toutefois, ce comportement normal n'est pas nécessairement le comportement observable le moins restrictif. Par contre, l'observabilité est fermée par l'intersection de langages préfixe-clos. Ainsi, si on considère l'ensemble $PO_{G,P}(K)$ des langages contenant K (K peut être non observable) qui sont préfixe-clos et observables, alors le plus petit langage contenant K, qui est préfixe-clos et observable par rapport à G et P, existe (unique) et on le note $\inf PO_{G,P}(K)$.

Jusque là, on a considéré que pour une spécification S donnée, un objectif essentiel du contrôle est d'avoir $\mathcal{L}_m(Sup/G) \subseteq \mathcal{L}_m(S)$. Il y a aussi eu d'autres travaux plus généraux ayant un objectif basé sur deux spécifications A (minimale) et E (maximale), c.à.d., $A \subseteq$

$\mathcal{L}_m(Sup/G) \subseteq E$. Pour plus de détail, voir [Brandt *et al.*, 1990; Kumar *et al.*, 1991; Lin et Wonham, 1988a].

Contrôle distribué des SED

Les SED distribués (ou concurrents) sont des systèmes qui sont obtenus par combinaison de composants interagissant entre eux. De tels SED sont généralement complexes au sens où ils possèdent un grand nombre d'états, concevoir un superviseur pour un tel système serait donc une tâche très complexe, voire impossible dans plusieurs cas. L'approche la plus souhaitable est de concevoir un superviseur localement pour chaque composant sans calculer explicitement le système global à contrôler. Dans ce cadre, étant donné un objectif de contrôle global, il faudrait que celui-ci soit atteint par la combinaison des différents superviseurs locaux. Pour plus de détails sur ce type de contrôle, voir [Abdelwahed, 2002; Abdelwahed et Wonham, 2002; Akesson *et al.*, 2002; de Queiroz et Cury, 2000a,b; Gaudin et Marchand, 2004; Jiang et Kumar, 2000; Khoumsi, 2005; Lafortune, 2007; Minhas et Wonham, 2003; Willner et Heymann, 1991].

1.2.3 Diagnostic des SED

Principe général du diagnostic

Le diagnostic des *défauts* (ou *fautes*) des systèmes industriels et informatiques a fait l'objet de nombreux travaux dans les dernières années. C'est une procédure qui permet la détection d'un défaut, de définir son origine et de déterminer ses causes. Un défaut d'un système industriel ou informatique est caractérisé par un changement inattendu du fonctionnement normal du système. Un tel défaut perturbe le comportement du système causant ainsi une détérioration de la performance et même amenant le système vers des situations dangereuses. Un système qui exécute un comportement fautif est dit *défaillant*, dans le cas contraire on dit que le système est *normal* ou *sain*.

Le diagnostic des défauts se base sur la comparaison des comportements réels du système avec les comportements sains ou défaillants. L'étude du diagnostic des défauts a connu un essor considérable avec la complexité croissante des systèmes industriels qui sont de plus en plus exigeants en termes de contraintes, de sécurité, de fiabilité, de disponibilité et de performances. De nos jours, les systèmes sont en effet de plus en plus complexes et ont donc tendance à tomber en panne en dépit des précautions de fabrication et manipulation. Ces pannes peuvent engendrer des conséquences très graves dont on peut citer comme exemples : des accidents d'avions, des fuites de radiations suite à des pannes dans les centrales nucléaires et des coupures de courant [Perrow, 1984]. L'une des solutions adoptées

pour remédier aux risques de pannes est de construire un module de diagnostic pour détecter les défaillances et pannes, empêcher leur propagation et limiter leurs conséquences.

Plusieurs méthodes ont été proposées dans la littérature pour étudier le diagnostic des défauts. Ces méthodes se divisent en deux sous-groupes.

Méthodes sans modèle Les méthodes non fondées sur un modèle mathématique sont dites des approches sans modèle. On trouve par exemple les tests statistiques et l'analyse des signatures, voir [Dubuisson, 2001; Hamscher *et al.*, 1992a; Pouliezos et Stavrakakis, 1994; Rich et Venkatasubramanian, 1987; Zwingelstein, 1995]. Dans ce type de méthodes, le système à diagnostiquer est considéré comme une "boîte noire" avec entrées et sorties. Elles utilisent uniquement un ensemble de mesures ou de connaissances heuristiques sur le système. En se basant ainsi sur des règles qui établissent des associations empiriques entre effets et causes, il est possible de lier les symptômes aux défauts.

Méthodes avec modèle Les méthodes fondées sur un modèle sont caractérisées par l'étude d'un modèle mathématique qui englobe un comportement normal et un autre défaillant. On peut avoir un modèle *quantitatif*, exprimé par exemple par des équations différentielles ou des fonctions de transfert [Frank, 1996; Gertler, 1998; Willsky, 1976]. On peut aussi avoir un modèle *qualitatif*, exprimé par exemple par des automates à états finis, des réseaux de Petri et des expressions logiques [Bavishi et Chon, 1994; Darwiche et Provan, 1996; Hamscher *et al.*, 1992a; Lin, 1994; Viswanadham et Narahari, 1992]. Parmi ces méthodes, on trouve aussi les méthodes de l'intelligence artificielle (e.g., la représentation des connaissances et la théorie des décisions [Baroni *et al.*, 1999; Frank, 1990; Hamscher *et al.*, 1992b; Lamperti et Zanella, 2002; Lee *et al.*, 1985; Reiter, 1987]) ou des méthodes inspirées du contrôle des SED sous observation totale ou partielle [Cardoso *et al.*, 1995; Sampath *et al.*, 1996; Su et Wonham, 2000]). Dans cette thèse, on s'intéresse plus précisément à la dernière catégorie, qu'on désignera par approche avec modèle SED.

Diagnostic des SED

Il existe plusieurs études qui se sont intéressées au diagnostic des SED modélisés par les AEF. Il est à noter que le diagnostic est étudié seulement dans le contexte de l'observation partielle (autrement, les défauts seront toujours détectables). Étant donné une faute (ou un défaut) f, $\mathcal{L}(G)$ englobe un comportement *sain*, qu'on note $\mathcal{H}$, et un comportement défaillant, qu'on note $\mathcal{F}$. Ce dernier est l'ensemble des traces caractérisées par l'occurrence de la faute f, c.à.d., $\mathcal{F} = \{s \in \mathcal{L}(G) | \exists u, v \in \Sigma^* : s = ufv\}$. Alors que $\mathcal{H}$ est l'ensemble

des traces non défaillantes, c.à.d., ne contenant pas f. Nous avons $\mathcal{L}(G) = \mathcal{H} \cup \mathcal{F}$ et $\mathcal{H} \cap \mathcal{F} = \emptyset$. Dans le cas où plusieurs fautes distinctes sont à considérer, la même procédure est suivie pour diagnostiquer chacune d'elles. Donc par soucis de clarté, une seule faute est considérée, la généralisation est triviale.

Notons que $\mathcal{H}$ est nécessairement préfixe-clos, parce que si une trace s est saine, alors tout préfixe de s est sain. Alors que $\mathcal{F}$ est nécessairement postfixe-clos par rapport $\mathcal{L}(G)$, dans le sens où $\mathcal{F}.\Sigma^* \cap \mathcal{L}(G) \subseteq \mathcal{F}$. Plus précisément, si une trace s est défaillante, alors toute extension de s est défaillante. Le diagnostic est effectué par un diagnostiqueur DIAG dont la fonction est d'observer le procédé à travers un masque P représenté par une projection naturelle $P : \Sigma \rightarrow \Sigma_o$ et d'émettre un diagnostic. Une détection d'une présence (resp., absence) de f est dite un diagnostic *positif* (resp., *négatif*). Un schéma général du diagnostic des SED est représenté dans la figure 1.4. Notons l'analogie avec le schéma de contrôle de la figure 1.3.

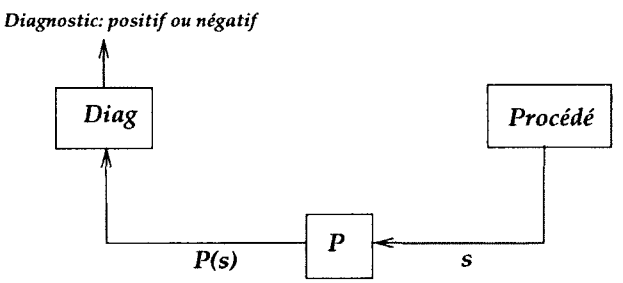

Figure 1.4 Schéma standard du diagnostic des SED

On trouve trois architectures principales pour étudier le diagnostic des SED.

Architecture centralisée Le SED est diagnostiqué par un seul diagnostiqueur centralisé qui observe le SED et émet un diagnostic concernant l'occurrence ou pas d'un défaut. Pour plus de détails sur cette architecture, voir [Sampath *et al.*, 1998, 1995, 1996; Zad *et al.*, 1998]. L'étude de l'architecture centralisée est un cas particulier de l'étude de l'architecture décentralisée. Ainsi toutes les études concernant l'architecture décentralisée (étudiée en détail dans cette thèse) sont valables aussi pour l'architecture centralisée.

Architecture distribuée Dans le cas des systèmes complexes physiquement distribués ou représentés par plusieurs sites, plusieurs facteurs comme les délais de communications, les erreurs de communications et les pertes d'informations pendant les communications peuvent être des sources de défaillances. Comme exemple de tels systèmes, on trouve les

réseaux de communications, les systèmes réseautiques, le trafic routier et les unités de fabrication. La méthode la plus adéquate pour détecter les défaillances des SED distribués est de construire un diagnostiqueur distribué. Plus précisément, pour chaque site, un diagnostiqueur local collecte des informations locales relatives à ce site. Ces informations peuvent être envoyées à d'autres diagnostiqueurs locaux. Vu la structure complexe et distribuée du SED à diagnostiquer, le fait d'envoyer les informations locales à un système central pour être analysées et émettre un diagnostic, peut générer des erreurs à cause des délais de communications et des pertes de données. Ainsi, la meilleure procédure pour diagnostiquer un tel système est de permettre à chaque diagnostiqueur d'engendrer un diagnostic local en se basant sur ses observations locales ainsi que sur les observations des autres diagnostiqueurs locaux. Pour plus de détails sur le diagnostic distribué voir [Aghasaryaiu *et al.*, 1997; Boel et van Schuppen, 2002; Debouk *et al.*, 2000; Fabre *et al.*, 2002; Qiu et Kumar, 2005; Sengupta, 1998; Su *et al.*, 2002].

La troisième architecture est l'architecture décentralisée. Comme elle fait l'objet de cette thèse, elle sera traitée avec plus de détails dans la section qui suit.

1.3 Contrôle et diagnostic décentralisés des SED

1.3.1 Architecture décentralisée de prise de décision

Nous avons vu dans la section 1.2.2 le principe général de la prise de décision pour les SED. Dans cette section, on s'intéresse plus particulièrement à la prise de décision dans le cadre d'une architecture décentralisée. Un schéma général de la prise de décision dans une architecture décentralisée est illustré dans la figure 1.5, où plusieurs *décideurs locaux* $(Dec_i)_{i\in\{1,\ldots,n\}}$ observent un procédé et coopèrent afin de prendre une *décision globale* qui respecte une propriété Pr donnée. Chaque décideur local Dec_i observe localement le procédé en observant un ensemble d'événements $\Sigma_{o,i}$. Soit alors $\Sigma_o = \bigcup_{i=1,\ldots,n} \Sigma_{o,i}$ l'ensemble des événements pouvant être observés par au moins un décideur local. Chaque Dec_i prend aussi une décision locale appartenant à un ensemble de décisions LD. Plus précisément, Dec_i est une fonction $Dec_i : \Sigma_{o,i}^* \rightarrow LD$. Les *décisions locales* des n décideurs locaux $(Dec_i)_{1\leq i\leq n}$ sont fusionnées par un *module de fusion* D afin de calculer une décision globale. Le module D est une fonction $\mathrm{D} : LD^n \rightarrow \{\phi, 0, 1\}$, où ϕ correspond au deux situations suivantes. Dans la première situation, le décideur ne peut pas prendre de décision certaine (0 ou 1), par exemple en raison d'un manque d'informations nécessaires. Dans la seconde situation, la décision (0 ou 1) n'a pas d'importance, c'est-à-dire n'a pas d'influence sur le respect de la propriété désirée. Le système englobant les décideurs locaux $(Dec_i)_{i\in\{1,\ldots,n\}}$ et

le module de fusion D est appelé *décideur décentralisé*. Les décisions locales et le module de fusion doivent donc garantir que la décision globale respecte la propriété Pr. Dans les deux sous-sections suivantes, on présente une introduction aux architectures décentralisées, respectivement pour le contrôle et le diagnostic des SED. Pour le cas du pronostic, la référence [Kumar et Takai, 2008] constitue une bonne introduction.

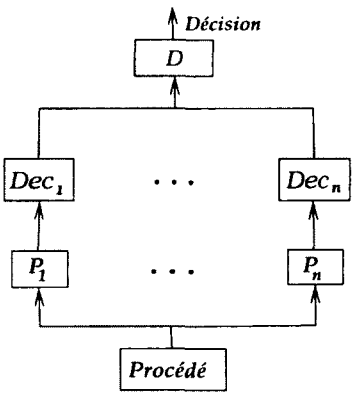

Figure 1.5 Prise de décision décentralisée

1.3.2 Contrôle décentralisé

Le *contrôle décentralisé* a été intensément étudié depuis de nombreuses années [Cieslak *et al.*, 1988; Jiang et Kumar, 2000; Kumar et Takai, 2007; Lin et Wonham, 1988a, 1990; Overkamp et van Schuppen, 2001; Prosser *et al.*, 1997; Ricker et Rudie, 2000, 2003; Rudie et Wonham, 1992; Tripakis, 2004; Yoo et Lafortune, 2002a, 2004]. Le contrôle décentralisé s'avère adéquat dans plusieurs situations, parmi celles-ci on trouve le cas où le procédé à contrôler est représenté par une collection de sites (systèmes en réseaux, télécommunications,...), ou si un superviseur ne peut se connecter au SED à contrôler qu'à travers certains points d'entrées. L'objectif du contrôle décentralisé est de synthétiser des *superviseurs locaux* selon leurs observations afin d'aboutir à un objectif de contrôle souhaité. Le contrôle décentralisé est une généralisation du contrôle à observation partielle dans le sens où plusieurs superviseurs locaux (au lieu d'un seul superviseur) observent et contrôlent le procédé. Il est à noter que les superviseurs n'observent pas et ne contrôlent pas nécessairement les mêmes événements.

Formellement, soit un procédé G et n superviseurs locaux $(Sup_i)_{1\leq i\leq n}$. Chaque superviseur Sup_i contrôle un ensemble d'événements $\Sigma_{c,i}$. Soit alors $\Sigma_c = \cup_{i=1,\ldots,n}\Sigma_{c,i}$ l'ensemble des événements pouvant être contrôlés par au moins un superviseur local. Chaque Sup_i observe un ensemble d'événements $\Sigma_{o,i}$ à travers la projection naturelle $P_i : \Sigma^* \rightarrow \Sigma^*_{o,i}$. Soit alors

$\Sigma_o = \cup_{i=1,\ldots,n}\Sigma_{o,i}$ l'ensemble des événements pouvant être observés par au moins un superviseur local. L'objectif du contrôle décentralisé est de synthétiser des superviseurs locaux $(Sup_i)_{1\leq i\leq n}$ dont les décisions locales sont fusionnées pour synthétiser une décision globale permettant de respecter un objectif de contrôle. La fusion est effectuée via un module de fusion selon une fonction qui peut être avec mémoire (e.g., AEF) ou sans mémoire (e.g., fonctions booléennes). L'ensemble englobant les projections $(P_i)_{i=1,\ldots,n}$, les superviseurs locaux et le module de fusion forment ce qu'on appelle un *superviseur décentralisé*. Selon les références qui ont étudié le contrôle décentralisé, les fonctions qui définissent le module de fusion, les décisions locales et les projections forment ce qu'on appelle *architecture* ou *technique*.

La première architecture proposée dans le cadre du contrôle décentralisé est dite architecture *conjonctive et permissive* (l'explication de ces termes est donnée dans cette section) ou plus simplement architecture C&P [Cieslak *et al.*, 1988; Jiang et Kumar, 2000; Rudie et Wonham, 1992; Yoo et Lafortune, 2002a]. Comme son nom l'indique, cette architecture permet de fusionner les décisions locales *conjonctivement*. Donc le but est de synthétiser n superviseurs locaux $(Sup_i)_{1\leq i\leq n}$ afin d'obtenir $\mathcal{L}_m(\bigwedge_i Sup_i/G) = K$, où $\bigwedge_i Sup_i$ représente le superviseur décentralisé formé par les superviseurs locaux et le module de fusion conjonctive, et K est une spécification donnée. Cieslak et al. [Cieslak *et al.*, 1988] ont étudié ce problème pour les comportements préfixe-clos, puis la généralisation a été faite par Rudie et Wonham [Rudie et Wonham, 1992]. En plus de la contrôlabilité et la $\mathcal{L}_m(G)$-clôture de K, les auteurs de [Rudie et Wonham, 1992] ont introduit la notion de *coobservabilité* que doit respecter K pour s'assurer de l'existence de la solution. La coobservabilité est une caractéristique généralisant l'observabilité dans le sens où il faut plusieurs superviseurs locaux pour la réaliser. En effet, un langage K est dit coobservable si au moins un superviseur local peut prendre la décision de l'inhibition d'un événement sans ambiguïté. Formellement, considérons deux superviseurs locaux Sup_1 et Sup_2 qui contrôlent (respectivement, observent) les alphabets $\Sigma_{c,1}$ et $\Sigma_{c,2}$ (respectivement, $\Sigma_{o,1}$ et $\Sigma_{o,2}$), avec $P_1 : \Sigma^* \to \Sigma_{o,1}^*$ et $P_2 : \Sigma^* \to \Sigma_{o,2}^*$ qui sont des projections naturelles. Si on doit inhiber un événement contrôlable $\sigma \in \Sigma_c$ au niveau d'une séquence $s \in \mathcal{L}(G)$ (c.à.d., $s\sigma \notin \overline{K}$, autrement dit $s \in \mathcal{D}_\sigma$), alors il doit exister un $i \in \{1,2\}$ tel que $(P_i^{-1}P_i(s)).\sigma \cap \overline{K} = \emptyset$ et $\sigma \in \Sigma_{c,i}$. Lorsqu'une telle propriété est respectée, on dit que K est *coobservable* (ou COOBS) par rapport à G et $P_1, \ldots, P_n$. Le même résultat a été trouvé par Jiang et Kumar dans un cadre plus général [Jiang et Kumar, 2000].

En plus de l'objectif de contrôle $\mathcal{L}_m(\bigwedge_i Sup_i/G) = K$, il y a aussi eu d'autres travaux plus généraux ayant un objectif basé sur deux spécifications A (minimale) et E (maximale),

c.à.d., $A \subseteq \mathcal{L}_m(\bigwedge_i Sup_i/G) \subseteq E$. Pour plus de détail, voir [Jiang et Kumar, 2000; Rudie et Wonham, 1992].

Les auteurs de [Prosser *et al.*, 1997] ont présenté la première alternative pour étudier le contrôle décentralisé avec plusieurs règles de fusion. Dans la suite de cette section, on discutera des différentes règles de fusion déjà étudiées dans la littérature. Par exemple dans le cas de l'architecture C&P, les règles qui permettent d'obtenir les décisions locales et la décision globale sont obtenues comme suit :

1. Chaque superviseur local est *permissif* dans le sens où il n'inhibe un événement que lorsqu'il est sûr que cet événement n'est pas accepté par la spécification.
2. Les décisions locales sont fusionnées par *intersection* (ou *conjonctivement*), pour obtenir la décision globale. Plus précisément, un événement est autorisé si et seulement si il est localement autorisé par tous les superviseurs locaux.

Les auteurs dans [Yoo et Lafortune, 2002a] ont proposé l'architecture D&A (pour *Disjonctive et Anti-permissive*) qui est le complémentaire de l'architecture C&P. Plus précisément, les règles qui permettent d'obtenir les décisions locales et la décision globale sont obtenues comme suit :

1. Chaque superviseur local est *anti-permissif* dans le sens où il n'autorise un événement que lorsqu'il est sûr que cet événement est accepté par la spécification.
2. Les décisions locales sont fusionnées par *union* (ou *disjonctivement*) pour obtenir la décision globale. Plus précisément, un événement est inhibé si et seulement si il est localement inhibé par tous les superviseurs locaux.

Les langages réalisables sous les architectures C&P et D&A sont dits respectivement C&P coobservables (ou C&P COOBS) et D&A coobservables (ou D&A COOBS). Les auteurs de [Yoo et Lafortune, 2002a] ont aussi étudié une architecture plus *générale* qui combine et généralise les architectures C&P et D&A. L'idée principale est de partitioner l'ensemble des événements contrôlables Σ_c en deux sous-ensembles disjoints $\Sigma_{c,\wedge}$ et $\Sigma_{c,\vee}$, c.à.d., $\Sigma_c = \Sigma_{c,\wedge} \cup \Sigma_{c,\vee}$ et $\Sigma_{c,\wedge} \cap \Sigma_{c,\vee} = \emptyset$. Les architectures C&P et D&A sont respectivement appliquées aux événements de $\Sigma_{c,\wedge}$ et $\Sigma_{c,\vee}$. Les auteurs de [Yoo et Lafortune, 2002a] ont montré qu'une partition $(\Sigma_{c,\wedge}, \Sigma_{c,\vee})$ peut être trouvée telle que la classe des langages réalisés par cette architecture générale englobe celles des langages réalisés par les architectures C&P et D&A. Un langage réalisable sous cette architecture générale est dit C&P$\vee$D&A *coobservable* (ou C&P$\vee$D&A COOBS).

Il a été montré dans [Rudie et Willems, 1995] que la complexité de la vérification de la C&P coobservabilité est polynômiale en terme du nombre d'états et de transitions du

procédé. En adaptant l'algorithme de [Rudie et Willems, 1995], les auteurs dans [Yoo et Lafortune, 2002a] ont montré que la complexité de la vérification de la D&A coobservabilité est polynômiale en terme du nombre d'états et de transitions. Les auteurs dans [Yoo et Lafortune, 2002a] ont montré que pour une partition $\Sigma_{c,\wedge}$ et $\Sigma_{c,\vee}$ donnée de Σ_c, la complexité de la vérification de C&P∨D&A COOBS est polynômiale en terme du nombre d'états et de transitions. En outre, ils ont montré que si une telle partition existe, alors elle peut être calculée en un temps polynômial.

Les auteurs dans [Yoo et Lafortune, 2004] ont utilisé une architecture *conditionnelle* qui généralise l'architecture C&P∨D&A de [Yoo et Lafortune, 2002a] en autorisant les superviseurs locaux à prendre des décisions conditionnelles telles que :

autoriser $\sigma \in \Sigma_c$ si aucun autre superviseur n'inhibe σ,
inhiber $\sigma \in \Sigma_c$ si aucun autre superviseur n'autorise σ.

Les auteurs [Yoo et Lafortune, 2004] ont montré que la classe des langages réalisés par l'architecture conditionnelle englobe celle des langages réalisés par l'architecture C&P∨D&A de [Yoo et Lafortune, 2002a]. Dans ce cadre, les auteurs dans [Yoo et Lafortune, 2004] ont défini les notions de C&P coobservabilité conditionnelle (ou C&P COND-COOBS) et D&A coobservabilité conditionnelle (ou D&A COND-COOBS) pour décrire les classes de langages qui sont réalisables sous les architectures conditionnelles. Une classe de langages qui englobent les deux précédentes classes, appelée C&P∨D&A coobservabilité conditionnelle (ou C&P∨D&A COND-COOBS) a été définie de telle sorte qu'un langage est C&P∨D&A COND-COOBS s'il existe une partition $\Sigma_{c,Cond-\wedge}$ et $\Sigma_{c,Cond-\vee}$ de Σ_c de telle sorte que le langage est C&P COND-COOBS par rapport à $\Sigma_{c,Cond-\wedge}$ et D&A COND-COOBS par rapport à $\Sigma_{c,Cond-\vee}$. Il a été établi dans le même article les relations qui existent entre les différentes classes de langages coobservables sous les différentes architectures. Le résumé de ces relations est montré dans la figure 1.6. Il a été montré dans [Yoo et Lafortune, 2004] que la complexité de la vérification de la C&P COND-COOBS (ou D&A COND-COOBS) est polynômiale en terme du nombre d'états et de transitions. Il a été aussi montré que pour une partition $\Sigma_{c,Cond-\wedge}$ et $\Sigma_{c,Cond-\vee}$ donnée de Σ_c, la complexité de la vérification de la C&P∨D&A COND-COOBS est polynômiale en terme du nombre d'états et de transitions. En outre, ils ont montré que si une telle partition existe, alors elle peut être calculée en un temps polynômial.

Dans [Ricker et Rudie, 2000, 2003], une approche a été proposée qui permet à chaque superviseur local d'utiliser ses observations ainsi que les observations des autres superviseurs locaux pour prendre une décision locale. Les auteurs [Kumar et Takai, 2005] ont proposé une approche par *inférence* plus générale qui permet d'aménager plus efficace-

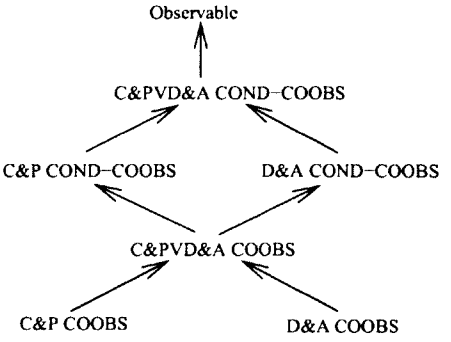

Figure 1.6 Relation entre les différentes classes de langages coobservables

ment les informations de tous les superviseurs locaux. L'idée principale est que chaque superviseur local associe à sa décision un niveau d'*ambiguïté*. En particulier, une décision avec un niveau d'ambiguïté nul veut dire que c'est une décision d'autoriser ou d'inhiber sans ambiguïté. Le principe est que quand le module de fusion reçoit plusieurs décisions à partir des superviseurs locaux, il sélectionne la décision locale "gagnante", c.à.d., la décision avec le niveau d'ambiguïté le plus bas. L'architecture est dite *N-inférence* si le degré d'ambiguïté de toute décision locale gagnante est inférieur ou égal à N, et un langage réalisable sous une architecture N-inférence est dit *N-inférence coobservable*. Il a été montré dans [Kumar et Takai, 2005, 2007] que la technique de décision par inférence généralise les architectures C&P (ou D&A) COOBS et C&P (ou D&A) COND-COOBS de [Yoo et Lafortune, 2002a, 2004], dans le sens où 0-inférence coobservabilité correspond à C&P (ou D&A) COOBS et 1-inférence coobservabilité correspond à C&P (ou D&A) COND-COOBS. En plus de décrire une méthode pour effectuer les calculs entre deux niveaux d'ambiguïtés, les auteurs dans [Kumar et Takai, 2005, 2007] ont montré que la complexité pour passer d'un niveau d'ambiguïté à un autre est polynômiale en terme du nombre d'états et de transitions. Cependant, le problème majeur de l'inférence par ambiguïté est la complexité de sa vérification. Au fur et à mesure que le niveau d'ambiguïté augmente, le nombre d'états augmente exponentiellement.

Notons que pour toutes les architectures et techniques discutées jusqu'ici, la complexité est exponentielle en terme du nombre de superviseurs locaux. Dans [Khoumsi et Chakib, 2007, 2008b], les auteurs ont proposé une technique qui consiste à ce que chaque superviseur local transmet ses observations au module de fusion lorsqu'il ne peut pas prendre de décision certaine (c.à.d., $\neq \phi$). Une telle approche permet une généralisation de toutes les architectures et techniques proposées dans la littérature. Cependant, la vérification de la condition de coobservabilité pour cette approche est indécidable en général dans le cas des langages infinis. En considérant des langages finis, une condition de coobservabilité, qui

définit la classe de langages réalisables sous cette technique, a été étudiée dans [Khoumsi et Chakib, 2008b]. Notons qu'il a été montré dans [Tripakis, 2004] que le problème du contrôle décentralisé est *indécidable* en général.

1.3.3 Diagnostic décentralisé

Dans l'architecture décentralisée, le SED est diagnostiqué par plusieurs diagnostiqueurs locaux où chacun observe localement et partiellement le SED. Chaque diagnostiqueur local émet un diagnostic local concernant l'occurrence d'un défaut. Les diagnostics locaux sont ainsi fusionnés par un module de fusion [Debouk *et al.*, 2000; Kumar et Takai, 2006, 2009; Qiu et Kumar, 2004, 2006; Su et Wonham, 2000; Takai et Kumar, 2006; Wang *et al.*, 2004, 2005, 2007].

Formellement, soit un procédé G et n diagnostiqueurs locaux $(Diag_i)_{1\leq i\leq n}$. Chaque diagnostiqueur $Diag_i$ observe l'ensemble des événements $\Sigma_{o,i}$ à travers la projection naturelle $P_i : \Sigma^* \rightarrow \Sigma_{o,i}^*$. Soit alors $\Sigma_o = \cup_{i=1,\ldots,n}\Sigma_{o,i}$ l'ensemble des événements pouvant être observés par au moins un diagnostiqueur local. Chaque $Diag_i$ émet un diagnostic local et l'ensemble des diagnostics locaux sont fusionnés pour synthétiser un diagnostic global permettant de respecter un objectif de diagnostic. La fusion est effectuée par un module de fusion selon une fonction qui peut être avec mémoire (e.g., AEF) ou sans mémoire (e.g., fonctions booléennes). L'ensemble englobant les projections $(P_i)_{i=1,\ldots,n}$, les diagnostiqueurs locaux et le module de fusion forment ce qu'on appelle un *diagnostiqueur décentralisé*. Il y a trois objectifs de diagnostic qui ont été largement étudiés dans la littérature, à savoir :

O1 : Le diagnostiqueur décentralisé doit détecter l'occurrence d'un défaut après un délai borné.

O2 : À chaque exécution non défaillante, le diagnostiqueur doit émettre un diagnostic négatif après un délai borné.

O3 : Si aucun défaut n'a été exécuté, alors le diagnostiqueur doit diagnostiquer avec certitude que le système était non défaillant dans un passé borné.

Dans [Debouk *et al.*, 2000], les auteurs ont développé quelques méthodes du diagnostic décentralisé, chacune est basée sur un protocole de communication entre les diagnostiqueurs locaux. L'un d'eux est appelé le Protocole 3, pour lequel chaque diagnostiqueur local opère sans communication avec les autres. Ils prennent des diagnostics locaux à propos de l'occurrence d'un défaut, et envoient leurs diagnostics à un module de fusion qui calcule par disjonction un diagnostic global. Les auteurs dans [Debouk *et al.*, 2000] ont considéré l'objectif **O1** où le langage qui peut être diagnostiqué selon cet objectif est

dit "diagnostiquable sous le Protocole 3". Qiu et Kumar [Qiu et Kumar, 2004, 2006] ont étudié le même objectif où le langage qui peut être diagnostiqué sous l'objectif **O1** est appelé *codiagnostiquable* (ou CODIAG). Les auteurs de [Qiu et Kumar, 2006] ont étudié aussi l'objectif **O2**, un langage diagnostiquable sous les deux objectifs **O1** et **O2** est dit *fortement codiagnostiquable.*

Dans [Wang *et al.*, 2004], les auteurs ont étudié les deux objectifs **O1** et **O2**, les langages pouvant être diagnostiqués correctement sous ces deux objectifs ont été respectivement nommés F-codiagnostiquable (ou plus simplement F-CODIAG) et NF-codiagnostiquable (ou plus simplement NF-CODIAG). Les auteurs de [Wang *et al.*, 2004] se sont inspirés du contrôle supervisé des SED [Yoo et Lafortune, 2002a, 2004] pour étudier le diagnostic décentralisé des SED. Ainsi, ils ont établi une correspondance entre les architectures conjonctives et disjonctives de [Yoo et Lafortune, 2002a, 2004] et les notions F-CODIAG et NF-CODIAG, respectivement. Comme pour le contrôle, les auteurs ont défini la notion de codiagnostiquabilité pour définir la classe de langages qui sont diagnostiquables sous une architecture générale qui regroupe les architectures conjonctive et disjonctive. En divisant l'ensemble des fautes Σ_f en deux sous-ensembles, $\Sigma_{f,\wedge}$ et $\Sigma_{f,\vee}$, un langage est dit F∨NF-CODIAG s'il est F-CODIAG par rapport à $\Sigma_{f,\wedge}$ et NF-CODIAG par rapport à $\Sigma_{f,\vee}$. Dans le même article [Wang *et al.*, 2004], les auteurs ont étudié (comme dans le contrôle) une *architecture conditionnelle* où chaque diagnostiqueur local émet un diagnostic qui dépend des autres diagnostics locaux. Ainsi, les décisions comme "positive si aucun autre n'émet négative" et "négative si aucun autre n'émet positive" ont été considérées. Dans ce cadre, les auteurs dans [Wang *et al.*, 2004] ont défini les notions de F-CODIAG conditionnelle (ou F-COND-CODIAG) et NF-CODIAG conditionnelle (ou NF-COND-CODIAG) pour décrire les classes de langages qui sont diagnostiquables sous les architectures conditionnelles. Une classe de langages qui englobent les deux précédentes classes, appelée CODIAG conditionnelle ou plus simplement F∨NF-COND-CODIAG, a été définie de telle sorte qu'un langage est F∨NF-COND-CODIAG s'il existe une partition $\Sigma_{f,Cond-\wedge}$ et $\Sigma_{f,Cond-\vee}$ de Σ_f de telle sorte que le langage est F-COND-CODIAG par rapport à $\Sigma_{f,Cond-\wedge}$ et NF-COND-CODIAG par rapport à $\Sigma_{f,Cond-\vee}$. Il a été établi dans le même article les relations qui existent entre les différentes classes de langages diagnostiquables sous les différentes architectures. le résumé de ces relations est montré dans la figure 1.7.

Cependant, les deux notions de F-codiagnostiquabilité (ou codiagnostiquabilité dans [Qiu et Kumar, 2006]) et NF-codiagnostiquabilité (ou la deuxième condition de la codiagnostiquabilité forte dans [Qiu et Kumar, 2006]) ne sont pas équivalentes dans l'architecture centralisée. Pour résoudre ce problème, les auteurs dans [Wang *et al.*, 2005] ont considéré

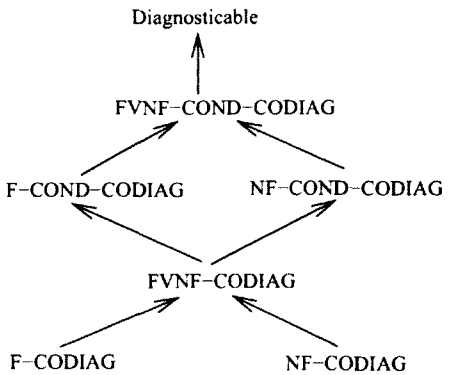

Figure 1.7 Relation entre les différentes classes de langages codiagnostiquables selon les objectifs **O1** et **O2**

les classes de langages qui sont diagnostiquables en considérant l'objectif **O3** au lieu de **O2** pour détecter un comportement non défaillant. L'architecture qui est utilisée pour détecter les comportements non défaillants est une architecture disjonctive (comme pour l'objectif **O2**), c.à.d., les décisions locales sont fusionnées disjonctivement. Les langages diagnostiquables sous cette architecture sont dits NF-CODIAG (n'a pas le même sens que dans [Wang *et al.*, 2004]). La même démarche utilisée dans [Wang *et al.*, 2004] a été suivie pour établir les notions de F[NF]-CODIAG, CODIAG, F[NF]-CODIAG conditionnelle et FVNF-COND-CODIAG. Le diagramme de relations entre les différentes classes de langages est le même que dans [Wang *et al.*, 2004] (voir la figure 1.7 en considérant l'objectif **O3** au lieu de **O2**), à l'exception du fait que les notions F-CODIAG et NF-CODIAG sont équivalentes dans le cas centralisé.

Dans [Wang *et al.*, 2007], les auteurs ont défini un cadre général pour étudier le diagnostic décentralisé. Selon leur approche, n diagnostiqueurs locaux $Diag_i$, $i = 1...n$, sont utilisés pour détecter un défaut f, où chaque $Diag_i$ fait un diagnostic local h_i qui est un élément d'un ensemble LD de décisions locales. Les diagnostics locaux sont fusionnés par un module de fusion selon une fonction globale $\mathbf{D} : LD^n \rightarrow \{positive, négative\}$. Les auteurs ont introduit la notion de ***D**-codiagnostiquabilité* pour décrire les langages pour lesquels il existe des fonctions de diagnostics locaux $(h_i)_{i=1,...,n}$ telles que les objectifs **O1** et **O3** sont respectés. Ils ont étudié les deux cas suivants :

$|LD| = 1$, ce cas regroupe F-CODIAG de [Wang *et al.*, 2004, 2005] et NF-CODIAG de [Wang *et al.*, 2005]. Ces deux notions sont respectivement appelées $\wedge$-CODIAG et $\vee$-CODIAG dans [Wang *et al.*, 2007].

$|LD| = 2$, ce cas regroupe F-COND-CODIAG de [Wang *et al.*, 2004, 2005] et NF-COND-CODIAG de [Wang *et al.*, 2005]. Ces deux notions sont respectivement appelées $\wedge$-COND-CODIAG et $\vee$-COND-CODIAG dans [Wang *et al.*, 2007].

En adoptant la formulation définie dans [Wang *et al.*, 2007], la relation entre les architectures décentralisées élaborées dans le contrôle des SED et le diagnostic des SED est plus explicite. En effet, les architectures conjonctive et disjonctive dans le contrôle peuvent être aisément projetées dans le diagnostic en remplaçant la décision "autoriser" (un événement) par (un diagnostic) "positif" et la décision "inhiber" (un événement) par (un diagnostic) "négatif". La même démarche est vraie pour les architectures conditionnelles.

Sengupta et Tripakis dans [Sengupta et Tripakis, 2002] ont étudié le cas où le module de fusion peut être n'importe quelle fonction arbitraire sans mémoire et les décisions locales comme étant les traces observées. Ils définissent la *diagnostiquabilité conjointe* (en anglais, *joint diagnosability*) comme une caractérisation de la classe de langages diagnostiquables sous ces conditions, et ils ont montré que la diagnostiquabilité conjointe est indécidable.

Un algorithme de complexité polynômiale en terme du nombre d'états et de transitions du procédé a été développé pour vérifier si un langage peut être diagnostiqué sous l'objectif **O1** [Qiu et Kumar, 2006]. Un test de codiagnostiquabilité forte (c.à.d., en considérant les objectifs **O1** et **O2**) de complexité polynômiale en terme du nombre d'états et de transitions du procédé, a été étudié dans [Qiu et Kumar, 2006] (voir aussi [Qiu et Kumar, 2004]). En s'inspirant de l'algorithme de vérification de F-CODIAG de [Qiu et Kumar, 2004], les auteurs de [Wang *et al.*, 2005, 2007] ont élaboré un algorithme, de complexité polynômiale en terme du nombre d'états et de transitions, pour la vérification de NF-CODIAG selon l'objectif **O3**. Dans [Wang *et al.*, 2007], un algorithme, de complexité polynomiale en terme du nombre d'états et de transitions, pour la vérification de F-COND-CODIAG (ou $\wedge$-COND-CODIAG) et NF-COND-CODIAG (ou $\vee$-COND-CODIAG) a été proposé. Cependant, dans toutes les architectures et techniques étudiées, la complexité est exponentielle en terme du nombre de diagnostiqueurs locaux.

En s'inspirant de la technique de contrôle de [Kumar et Takai, 2005, 2007], les auteurs dans [Kumar et Takai, 2006, 2009] ont proposé une technique appelée *inférence par ambiguïté* qui généralise les cas spéciaux $|LD| = 1$ et $|LD| = 2$ de [Wang *et al.*, 2007]. L'idée de leur approche est basée sur le fait que chaque diagnostiqueur local associe un niveau d'ambiguïté à chaque diagnostic local. Le module de fusion sélectionne le diagnostic local "gagnant", c.à.d., ayant le plus petit degré d'ambiguïté. Si deux diagnostiqueurs locaux émettent deux diagnostics différents avec le même degré d'ambiguïté minimal, alors le diagnostic effectif est ϕ qui veut dire que le diagnostiqueur décentralisé est *incertain*.

L'architecture est dite *N-inférence* si le degré d'ambiguïté de toute décision locale gagnante est inférieur ou égal à N. Dans [Kumar et Takai, 2006] les auteurs ont considéré l'objectif **O1**, et un langage diagnostiquable conformément à cet objectif avec l'architecture N-inférence est dit N-inférence F-diagnostiquable. Dans [Kumar et Takai, 2006, 2009], le diagnostic global peut être positif, négatif ou *incertain* (représenté par la décision ϕ). L'approche de l'inférence par ambiguïté permet aussi d'atteindre l'objectif suivant :

O4 : Le diagnostiqueur n'engendre pas de *fausse alarme*, c.à.d., un diagnostic positif n'est pas émis avant l'occurrence d'un défaut et un diagnostic négatif n'est pas émis après l'occurrence d'un défaut.

Il faut noter que dans [Kumar et Takai, 2006, 2009] si le diagnostic global ϕ est remplacé par 0 (négatif), alors l'objectif de [Wang *et al.*, 2005, 2007] devient équivalent à celui de [Kumar et Takai, 2006, 2009]. Les auteurs dans [Kumar et Takai, 2006, 2009] ont comparé l'architecture N-inférence avec les autres architectures dans les cas particuliers $N = 0$ et $N = 1$. Ainsi, ils ont montré que la 0-inférence F-diagnostiquabilité est équivalente à la codiagnostiquabilité de [Qiu et Kumar, 2006] (F-CODIAG de [Wang *et al.*, 2004, 2005, 2007] et $\wedge$-CODIAG de [Wang *et al.*, 2007]). Il a été aussi montré que la 1-inférence F-diagnostiquabilité est *plus forte* que la F-COND-CODIAG (ou $\wedge$-COND-CODIAG) [Wang *et al.*, 2004, 2005, 2007].

Dans [Takai et Kumar, 2006], la technique de l'inférence par ambiguïté a été utilisée dans le but d'obtenir l'objectif suivant :

O5 : Après toute exécution saine de longueur suffisamment longue, un diagnostic global négatif doit être émis.

En outre, par construction du diagnostiqueur décentralisé par inférence, l'objectif **O4** (aucune fausse alarme) est assuré (comme dans [Kumar et Takai, 2006, 2009]). Un langage diagnostiquable conformément à l'objectif **O5** avec l'architecture N-inférence est dit N-inférence NF-diagnostiquable. Pour comparer l'approche par inférence avec les autres études et approches, les auteurs ont considéré les propriétés de l'architecture N-inférence dans les cas particuliers où $N = 0$ et $N = 1$. Ainsi, ils ont montré que la propriété 0-inférence NF-diagnostiquabilité est *plus faible* que NF-codiagnostiquabilité dans [Wang *et al.*, 2004] (ou la deuxième condition de la codiagnostiquabilité forte dans [Qiu et Kumar, 2006]). Il a été aussi montré dans [Takai et Kumar, 2006] que la propriété 0-inférence NF-diagnostiquable est *plus forte* que NF-CODIAG de [Wang *et al.*, 2005] (ou $\vee$-CODIAG de [Wang *et al.*, 2007]). De même, il a été montré dans [Takai et Kumar, 2006] que la propriété 1-inférence NF-diagnostiquable est *plus forte* que NF-COND-CODIAG de [Wang *et al.*, 2005] (ou $\vee$-COND-CODIAG de [Wang *et al.*, 2007]). En considérant les objectifs

O1 dans [Kumar et Takai, 2006, 2009] et **O5** dans [Takai et Kumar, 2006, 2010], les auteurs ont montré que la complexité pour passer d'un niveau d'ambiguïté à un autre est polynômiale en terme du nombre d'états et de transitions.

Notons que pour toutes les architectures et techniques discutées dans le cadre du diagnostic décentralisé, la complexité est exponentielle en terme du nombre de diagnostiqueurs locaux. La technique de l'inférence par ambiguïté permet la diagnostiquabilité d'une plus large classe de langages. Cependant, le problème majeur de l'inférence par ambiguïté est la complexité de sa vérification. Au fur et à mesure que le niveau d'ambiguïté augmente, le nombre d'états augmente exponentiellement.

1.4 Problématique, motivations et objectifs de la thèse

1.4.1 Étude des SED : nécessité et défis

De nos jours, les systèmes technologiques sont devenus très complexes (matériel informatique, logiciel, système de télécommunication, usine manufacturière, etc), et cette complexité croît continuellement de sorte que les anciennes techniques intuitives utilisées pour leur conception, leur étude et leur réalisation deviennent inadaptées. Plus spécialement, les logiciels et les matériaux informatiques connaissent une croissance encore plus marquante dans leurs tailles et fonctionnalités. Par exemple, un véhicule moderne contient typiquement entre une douzaine à environ 100 unités électroniques, et quelques millions de lignes de code [Cook *et al.*, 2007]. En 2007, on estimait qu'en 2010, le nombre de lignes de code estimé devrait atteindre l'ordre de centaines de millions [Sangiovanni-Vincentelli, 2007]. À cause de cette complexité croissante, la probabilité pour qu'une erreur (ou panne) inattendue survienne est de plus en plus grande. Plus encore, quelques erreurs peuvent provoquer des accidents très graves causant des pertes économiques ou humaines. Il devient donc primordial de développer des méthodes formelles d'analyse, de conception et de réalisation des systèmes logiciels et électroniques quelque soit leur complexité. Cela favorise le développement d'outils logiciels supportant ces méthodes formelles pour vérifier ou garantir automatiquement l'exactitude des systèmes très complexes. Dans ce cadre, l'étude des SED a été introduite avec l'objectif de développer des méthodes formelles pour répondre à des besoins pressants, tels que le contrôle, le diagnostic, le pronostic, le test et la vérification des comportements discrets des systèmes technologiques.

Un des problèmes les plus importants (sinon le plus important) auxquels sont confrontés les concepteurs de méthodes formelles pour les SED, c'est la complexité (en temps d'exé-

cution et en mémoire utilisée) qui peut croître d'une manière inacceptable (par exemple exponentielle). Lors du développement d'une méthode formelle, il faut donc toujours s'assurer que sa complexité se trouve dans des limites acceptables.

Dans cette thèse, nous nous intéressons en particulier au contrôle et au diagnostic. Dans ces deux domaines, on est confronté au problème de complexité qui est en partie dû à la composition d'automates. Il est donc important de développer des méthodes évitant le plus possible les compositions d'automates.

Dans la majorité des cas, les systèmes technologiques modélisés en SED se présentent sous une forme distribuée où plusieurs SED fonctionnent ensemble pour réaliser un comportement général. La meilleure approche pour contrôler ou diagnostiquer de tels SED est l'approche décentralisée, car autrement on devrait procéder à une composition des sous-systèmes du SED distribué. Cependant, le contrôle et le diagnostic décentralisés des SED doivent faire face à trois défis majeurs qui mettent en question leur efficacité.

1. Le premier défi est de développer des méthodes de construction de décideurs (superviseurs ou diagnostiqueurs) avec des complexités de calcul acceptables. Bien que les architectures décentralisées permettent de réduire la complexité due à la composition d'automates, il reste d'autres opérations engendrant la complexité, telles que la projection d'automates sur des sous-alphabets.
2. Le second défi est d'obtenir des conditions de coobservabilité et de codiagnostiquabilité les moins contraignantes possibles. Il s'agit de développer des architectures de décideurs (superviseurs ou diagnostiqueurs) pouvant traiter des cas qui ne pouvaient pas être traités par des architectures existantes.
3. Le troisième défi est de développer des méthodes non complexes de vérification des conditions de coobservabilité et de codiagnostiquabilité.

Dans cette thèse, nous nous intéressons particulièrement aux deuxième et troisième défis.

1.4.2 Limitations des architectures décentralisées existantes

Un problème important relatif au contrôle et au diagnostic est que chaque architecture développée peut traiter un ensemble limité de langages défini par une propriété de coobservabilité ou codiagnostiquabilité associée à l'architecture en question. Afin de repousser cette limitation, des architectures et techniques de plus en plus générales ont été développées. Le fait qu'une architecture A est plus générale qu'une architecture B, se traduit par le fait que la condition de coobservabilité ou codiagnostiquabilité relative à A est moins

contraignante que celle relative à B. Par conséquent, il existera des cas où la coobservabilité ou diagnostiquabilité relative à A est respectée alors que celle relative à B ne l'est pas. Dans la section 1.3, nous avons présenté les différentes versions de coobservabilités et de codiagnostiquabilités qui ont été étudiées dans la littérature.

Un autre problème majeur relatif au contrôle et au diagnostic des architectures décentralisées est la complexité de la vérification de la coobservabilité et de la codiagnostiquabilité. En effet, que se soit pour le contrôle ou le diagnostic décentralisés, dans toutes les architectures et techniques étudiées, la complexité est exponentielle en terme du nombre de superviseurs ou diagnostiqueurs locaux. La technique de l'inférence par ambiguïté permet la réalisation et la diagnostiquabilité d'une plus large classe de langages. Cependant, le problème majeur de l'inférence par ambiguïté est la complexité de sa vérification. Au fur et à mesure que le niveau d'ambiguïté augmente, le nombre d'états augmente exponentiellement. Ainsi, on voit que le prix à payer pour avoir des classes de langages plus larges est une complexité plus élevée.

1.4.3 Objectifs poursuivis

Comme on l'a expliqué dans la sous-section précédente, l'étude du contrôle et du diagnostic décentralisés fait face à un dilemme : trouver des conditions de coobservabilité et codiagnostiquabilité les plus générales possibles, mais dont la vérification n'est pas trop complexe. L'objectif de la présente thèse est de résoudre ce dilemme le mieux que nous pouvons. Cela consiste en deux objectifs qui correspondent respectivement aux second et troisième défis exprimés dans la section 1.4.1. Voyons ces deux objectifs un peu plus précisément :

1. Depuis l'apparition de la technique de l'inférence par ambiguïté en 2005 [Kumar et Takai, 2005], aucune architecture plus générale n'a été proposée. Notre premier objectif dans cette thèse est justement de proposer une nouvelle approche qui nous a permis de développer des architectures dite multi-décisionnelles qui sont plus générales que toutes celles existantes, incluant l'architecture par inférence.

2. Notre second objectif est de développer une méthode *efficace* pour vérifier les coobservabilité et codiagnostiquabilité associées aux architectures multi-décisionnelles que nous avons développées. Par "méthode efficace", nous voulons dire une méthode de vérification de coobservabilité et codiagnostiquabilité dont la complexité (en temps de calcul et en mémoire utilisée) n'est pas plus élevée que les méthodes de vérification développées pour des architectures décentralisées moins générales.

1.5 Contributions, résultats et organisation de la thèse

1.5.1 Contributions

Principe de l'approche multi-décisionnelle dans la prise de décision

Nous avons discuté dans la section 1.3 du principe de la prise de décision dans les architectures décentralisés (la figure 1.5 présente un schéma général d'un décideur décentralisé). L'objectif de chaque décideur décentralisé est de prendre des décisions globales qui respectent un objectif (ou propriété) Pr donné. Il y a différents types d'objectifs, voir la section 1.2 pour les différents types d'objectifs liés au contrôle, au diagnostic et au pronostic des SED.

La principale contribution de cette thèse est de proposer une nouvelle approche, appelée *multi-décision* ou *multi-décisionnelle*. Le principe de la multi-décision est basé sur l'utilisation de plusieurs (disons p) décideurs décentralisés $(DD^j)_{j=1,\ldots,p}$ qui fonctionnent simultanément et en parallèle. Les décisions globales des décideurs décentralisés sont fusionnées afin d'obtenir une *décision effective*, comme illustré dans la figure 1.8. Chaque DD^j a la structure de la figure 1.5, c.à.d., il regroupe un ensemble de décideurs locaux $(Dec_i^j)_{i\in I}$ dont les décisions locales sont fusionnées par un module de fusion D^j afin d'obtenir une décision globale respectant une propriété Pr^j. Chaque décideur décentralisé DD^j peut donc être noté $((Dec_i^j)_{i\in I}, \mathrm{D}^j)$. Les décisions globales des p $(DD^j)_{j=1,\ldots,p}$ sont fusionnées par un module $\mathbf{D}$ afin que la décision effective respecte la propriété désirée Pr. Le décideur obtenu est dit multi-décisionnel et peut être noté $DD = ((DD^j)_{j=1,\ldots,p}, \mathbf{D}) = (((Dec_i^j)_{i\in I}, \mathrm{D}^j)_{j=1,\ldots,p}, \mathbf{D})$. L'architecture obtenue est dite multi-décisionnelle et notée $\mathbf{D}$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$.

Le fait d'utiliser plusieurs architectures décentralisées revient à considérer une *seule* architecture décentralisée dont chaque décideur local engendre plusieurs décisions locales au lieu d'une seule. D'où l'utilisation du terme "multi-décision".

Principe de l'approche multi-décisionnelle dans la prise de décision dans le contrôle des SED

Le calcul des décisions locales et globales des superviseurs décentralisés, dépendamment de l'architecture ou technique appliquée, fait appel aux ensembles $\mathcal{E}_\sigma$ et $\mathcal{D}_\sigma$ (voir la section 1.2.2). Pour plus de détails sur les prises des décisions locales et globales dans le contrôle décentralisé, voir [Kumar et Takai, 2005, 2007], L'utilisation de $\mathcal{E}_\sigma$ et $\mathcal{D}_\sigma$ est implicite dans [Yoo et Lafortune, 2002a, 2004].

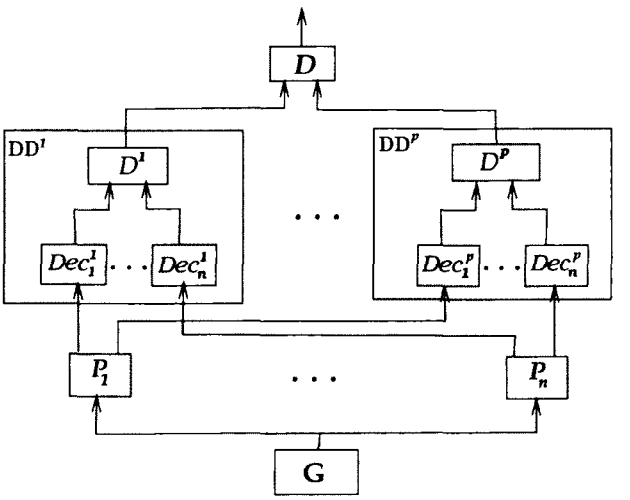

Figure 1 8 Schéma général d'un décideur multi-décisionnel

Dans le cas du contrôle supervisé, l'approche multi-décisionnelle est appelée *contrôle multi-décisionnel*. Les décideurs décentralisés DD^j de la figure 1.8 sont appelés superviseurs décentralisés et notés Sup^j. Les décideurs locaux Dec_i^j sont appelés superviseurs locaux et notés Sup_i^j. Ainsi, le contrôle multi-décisionnel consiste à utiliser plusieurs (disons p) superviseurs décentralisés $(Sup^j)_{j=1,\ldots,p}$ fonctionnant en parallèle Chaque superviseur décentralisé Sup^j est donc défini par un ensemble de superviseurs locaux $(Sup_i^j)_{i=1,\ldots,n}$ et un module de fusion D^j, et il est ainsi noté $Sup^j = ((Sup_i^j)_{i=1,\ldots,n}, \mathrm{D}^j)$. Les décisions globales de tous les superviseurs décentralisés $(Sup^j)_{j=1,\ldots,p}$ sont ensuite fusionnées selon une fonction de fusion $\mathbf{D}$ afin d'obtenir une décision *effective* qui sera appliquée au procédé. Le système composé des superviseurs décentralisés $(Sup^j)_{j=1,\ldots,p}$ et de la fonction de fusion $\mathbf{D}$ est appelé *superviseur multi-décisionnel*, qu'on note $Sup = ((Sup^j)_{j=1,\ldots,p}, \mathbf{D})$, et qui a pour but de contrôler le procédé afin de réaliser une spécification donnée. L'architecture obtenue est notée $\mathbf{D}$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$.

Nous avons étudié les cas où la décision effective est obtenue en fusionnant les décisions globales des différents p superviseurs décentralisés $(Sup^j)_{j=1,\ldots,p}$, disjonctivement ($\mathbf{D} = \vee$) ou conjonctivement ($\mathbf{D} = \wedge$). Dans le cas $\mathbf{D} = \vee$, pour chaque $\sigma \in \Sigma_c$, on utilise une décomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ de $\mathcal{E}_\sigma$ telle que chaque superviseur décentralisé Sup^j prend sa décision en se basant sur le couple $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ au lieu de $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$. Dans le cas $\mathbf{D} = \wedge$, pour chaque $\sigma \in \Sigma_c$, on utilise une décomposition $\{\mathcal{D}_\sigma^1, \ldots, \mathcal{D}_\sigma^p\}$ de $\mathcal{D}_\sigma$ telle que chaque superviseur décentralisé Sup^j prend sa décision en se basant sur le couple $(\mathcal{E}_\sigma, \mathcal{D}_\sigma^j)$ au lieu de $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$.

Principe de l'approche multi-décisionnelle dans la prise de décision dans le diagnostic des SED

Concernant le diagnostic décentralisé, le calcul des diagnostics locaux est basé sur les ensembles $\mathcal{F}$ et $\mathcal{H}$ (voir la section 1.2.3). Le calcul des diagnostics locaux et globaux dépend de l'architecture ou technique appliquée. Pour plus de détails sur les prises des décisions locales dans le diagnostic décentralisé, voir [Kumar et Takai, 2006, 2009; Takai et Kumar, 2006; Wang *et al.*, 2004, 2005, 2007].

Dans le cas du diagnostic, l'approche multi-décisionnelle est appelée *diagnostic multi-décisionnel.* Les décideurs décentralisés DD^j de la figure 1.8 sont alors appelés diagnostiqueurs décentralisés et notés $Diag^j$. Les décideurs locaux Dec_i^j sont appelés diagnostiqueurs locaux et notés $Diag_i^j$. Ainsi, le diagnostic multi-décisionnel est basé sur l'utilisation de plusieurs (disons p) diagnostiqueurs décentralisés $(Diag^j)_{j=1,\ldots,p}$ fonctionnant en parallèle. Chaque diagnostiqueur décentralisé $Diag^j$ est donc défini par un ensemble de diagnostiqueurs locaux $(Diag_i^j)_{i=1,\ldots,n}$ et un module de fusion D^j, et il est ainsi noté $Diag^j = ((Diag_i^j)_{i=1,\ldots,n}, \mathrm{D}^j)$. Les diagnostics globaux de tous les diagnostiqueurs décentralisés $(Diag^j)_{j=1,\ldots,p}$ sont ensuite fusionnés selon une fonction de fusion $\mathbf{D}$ afin d'obtenir un diagnostic *effectif.* Le système composé des diagnostiqueurs décentralisés $(Diag^j)_{j=1,\ldots,p}$ et de la fonction de fusion $\mathbf{D}$ est appelé *diagnostiqueur multi-décisionnel*, qu'on note $Diag = ((Diag^j)_{j=1,\ldots,p}, \mathbf{D})$, et qui a pour but de diagnostiquer le procédé dans le but de respecter un objectif de diagnostic. L'architecture obtenue est notée $\mathbf{D}$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$.

Nous avons étudié les cas où le diagnostic effectif est obtenu en fusionnant les diagnostics globaux des différents diagnostiqueurs décentralisés $(Diag^j)_{j=1,\ldots,p}$, disjonctivement ($\mathbf{D} = \vee$) ou conjonctivement ($\mathbf{D} = \wedge$). Dans le cas $\mathbf{D} = \vee$, on utilise une décomposition $(\mathcal{F}^1, \ldots, \mathcal{F}^p)$ de $\mathcal{F}$ telle que chaque diagnostiqueur décentralisé $Diag^j$ fait un diagnostic en se basant sur le couple $(\mathcal{F}^j, \mathcal{H})$ au lieu de $(\mathcal{F}, \mathcal{H})$. Dans le cas $\mathbf{D} = \wedge$, on utilise une décomposition $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ de $\mathcal{H}$ telle que chaque diagnostiqueur décentralisé $Diag^j$ fait un diagnostic en se basant sur le couple $(\mathcal{F}, \mathcal{H}^j)$ au lieu de $(\mathcal{F}, \mathcal{H})$.

1.5.2 Résultats

Contrôle multi-décisionnel

Le contrôle multi-décisionnel a été étudié en détail dans le cas où les architectures des superviseurs décentralisés sont bien définies. Plus précisément, le cas où l'architecture est une C&P, D&A , C&P conditionnelle, D&A conditionnelle ou inférence par ambiguïté.

La référence [Chakib et Khoumsi, 2008b] contient la première tentative pour l'utilisation du principe multi-décisionnel afin de généraliser l'architecture C&P. La nouvelle architecture est dite C&P multi-décisionnelle. On utilise une décomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ de $\mathcal{E}_\sigma$ et l'opérateur de fusion $\mathbf{D} = \vee$. La notion de C&P p-coobservabilité a été définie pour caractériser la classe de langages réalisables sous l'architecture C&P multi-décisionnelle. Une spécification K est dite C&P p-coobservable si, pour chaque $\sigma \in \Sigma_c$, il existe une décomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ de $\mathcal{E}_\sigma$ telle que, pour chaque $\mathcal{E}_\sigma^j$, $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ est C&P COOBS.

Le principe du contrôle multi-décisionnel est basé sur des décompositions des langages $\mathcal{E}_\sigma$ et $\mathcal{D}_\sigma$. Comme la décomposition d'un langage infini pose, en général, un problème de décidabilité, on a proposé dans [Chakib et Khoumsi, 2008a,b], dans le cas des langages réguliers, une méthode qui transforme le problème de la décomposition d'un langage régulier infini (en l'occurrence $\mathcal{E}_\sigma$ ou $\mathcal{D}_\sigma$) en un problème de décomposition de l'ensemble d'états marqués d'un AEF $\mathcal{A}$ acceptant $\mathcal{E}_\sigma$ ou $\mathcal{D}_\sigma$. On a ainsi défini une notion de coobservabilité multi-décisionnelle *forte* par rapport à $\mathcal{A}$, pour caractériser la classe de langages réalisables sous les architectures multi-décisionnelles, en considérant des décompositions $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ de $\mathcal{E}_\sigma$ ou $\{\mathcal{D}_\sigma^1, \ldots, \mathcal{D}_\sigma^p\}$ de $\mathcal{D}_\sigma$, qui respectent la condition suivante : $\mathcal{E}_\sigma^j$ (ou $\mathcal{D}_\sigma^j$) contient toutes les traces menant à un ou plusieurs états marqués de l'AEF acceptant $\mathcal{E}_\sigma$ (ou $\mathcal{D}_\sigma$). Dans [Chakib et Khoumsi, 2008a], on a proposé une méthode qui permet, à partir d'un AEF $\mathcal{A}_1$ acceptant $\mathcal{E}_\sigma$, pour lequel $\mathcal{E}_\sigma$ n'est pas C&P p-coobservable par rapport à $\mathcal{A}_1$, de calculer un autre AEF $\mathcal{A}_2$ pour lequel $\mathcal{E}_\sigma$ est C&P p-coobservable par rapport à $\mathcal{A}_2$.

Dans la référence [Chakib et Khoumsi, 2008a] (présentée au chapitre 2), en plus de l'architecture C&P multi-décisionnelle, on a étudié le contrôle multi-décisionnel dans le cas des architectures D&A et C&P$\vee$D&A. La notion de D&A p-coobservabilité a été définie pour caractériser la classe de langages réalisables sous l'architecture D&A multi-décisionnelle. Une spécification K est dite D&A p-coobservable si, pour chaque $\sigma \in \Sigma_c$, il existe une décomposition $\{\mathcal{D}_\sigma^1, \ldots, \mathcal{D}_\sigma^p\}$ de $\mathcal{D}_\sigma$ telle que, pour chaque $\mathcal{D}_\sigma^j$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma^j)$ est D&A COOBS. L'architecture C&P$\vee$D&A multi-décisionnelle a été définie pour généraliser les deux architectures C&P multi-décisionnelle et D&A multi-décisionnelle. L'idée principale est de partitionner l'ensemble des événements contrôlables Σ_c en deux sous-ensembles disjoints $\Sigma_{c,\wedge}$ et $\Sigma_{c,\vee}$, c.à.d., $\Sigma_c = \Sigma_{c,\wedge} \cup \Sigma_{c,\vee}$ et $\Sigma_{c,\wedge} \cap \Sigma_{c,\vee} = \emptyset$. Les architectures C&P multi-décisionnelle et D&A multi-décisionnelle sont respectivement appliquées aux événements de $\Sigma_{c,\wedge}$ et $\Sigma_{c,\vee}$. Ainsi, la classe de langages réalisables par l'architecture C&P$\vee$D&A multi-décisionnelle englobe les classes de langages réalisables par les architectures C&P multi-décisionnelle et D&A multi-décisionnelle. Nous proposons aussi dans ce chapitre, une méthode qui permet de transformer un AEF, pour qui la condition de coobservabi-

lité multi-décisionnelle n'est pas satisfaite, en un autre AEF pour qui cette condition est satisfaite.

Dans la référence [Chakib et Khoumsi, 2011b] (présentée au chapitre 3) on a proposé la forme *générique* du contrôle multi-décisionnel, qui consiste à utiliser des superviseurs décentralisés qui fonctionnent en parallèle et dont les décisions sont fusionnées par un module de fusion. Nous avons identifié des conditions suffisantes qui rendent une architecture décentralisée éligible pour être utilisée dans une architecture multi-décisionnelle. Nous avons effectué une étude générique de l'approche multi-décisionnelle en considérant plusieurs architectures éligibles fonctionnant en parallèle. Ainsi, nous avons étudié en détail les cas où la décision finale est obtenue en combinant les décisions globales de tous les superviseurs décentralisés Sup^j ($j \in \{1, \ldots, p\}$) soit disjonctivement ($\mathbf{D} = \vee$) soit conjonctivement ($\mathbf{D} = \wedge$). Dans le cas $\mathbf{D} = \vee$ (resp., $\mathbf{D} = \wedge$), pour chaque $\sigma \in \Sigma_c$, nous utilisons une décomposition $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p)$ de $\mathcal{E}_\sigma$ (resp., $(\mathcal{D}_\sigma^1, \ldots, \mathcal{D}_\sigma^p)$ de $\mathcal{D}_\sigma$) telle que chaque superviseur décentralisé Sup^j prend sa décision en se basant sur $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ (resp., $(\mathcal{E}_\sigma, \mathcal{D}_\sigma^j)$). Nous avons étudié en détail le cas $\mathbf{D} = \vee$ (c.à.d., décomposition de $\mathcal{E}_\sigma$), mais nous avons expliqué comment les résultats peuvent être adaptés au cas $\mathbf{D} = \wedge$ (c.à.d., décomposition de $\mathcal{D}_\sigma$). Dans le cas $\mathbf{D} = \vee$, nous avons défini la notion de $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS, qui est utile pour caractériser la classe de langages réalisables sous l'architecture $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$. En plus, nous avons montré que la classe de langages réalisables sous l'architecture $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$ englobe la classe de langages réalisables sous l'architecture D^j, pour chaque $j \in \{1, \ldots, p\}$. Dans le but de montrer l'efficacité de notre approche, nous avons appliqué l'approche multi-décisionnelle au cas spécial où plusieurs (disons p) Inf_{N_j}-superviseurs ($j = 1 \cdots p$) à inférence par ambiguïté fonctionnent en parallèle et dont les décisions globales sont fusionnées disjonctivement.

Dans la référence [Chakib et Khoumsi, 2011c] (présentée au chapitre 4), on a proposé une méthode (et l'algorithme correspondant) qui vérifie si $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ est $\vee$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-COOBS par rapport à un AEF $\mathcal{A}_{\mathcal{E}_\sigma}$ donné acceptant $\mathcal{E}_\sigma$, où p est un entier (inférieur ou égal au nombre des états marqués de l'AEF $\mathcal{A}_{\mathcal{E}_\sigma}$) calculé par l'algorithme. La même méthode nous permet d'obtenir un algorithme qui vérifie si $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ est $\wedge$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-COOBS par rapport à un AEF $\mathcal{A}_{\mathcal{D}_\sigma}$ donné acceptant $\mathcal{D}_\sigma$, où p est un entier (inférieur ou égal au nombre des états marqués de l'AEF $\mathcal{A}_{\mathcal{D}_\sigma}$) calculé par l'algorithme. Nous avons montré que la complexité de l'algorithme, dans le pire des cas, n'augmente pas avec le nombre des superviseurs décentralisés fonctionnant en parallèle. Plus précisément, la complexité de vérification de la coobservabité selon l'architecture multi-décisionnelle est, dans le pire cas,

du même ordre de grandeur que celle de la vérification de la coobservabité selon une des architectures décentralisées en parallèle qui constituent l'architecture multi-décisionnelle.

Diagnostic multi-décisionnel

Dans la référence [Khoumsi et Chakib, 2008a], le principe multi-décisionnel a été élaboré pour le diagnostic des SED selon les objectifs **O1**, **O4** et **O5** (voir la section 1.3.3). Ainsi, la notion de F-p-CODIAG a été définie pour caractériser la classe de langages diagnostiquables selon les objectifs **O1** et **O4**. Le couple $(\mathcal{F}, \mathcal{H})$ est dit F-p-CODIAG s'il existe une décomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ de $\mathcal{H}$ telle que, pour tout $\mathcal{H}^j$, $(\mathcal{F}, \mathcal{H}^j)$ est F-CODIAG. La notion de NF-p-CODIAG a été définie pour caractériser la classe de langages diagnostiquables selon les objectifs **O4** et **O5**. Le couple $(\mathcal{F}, \mathcal{H})$ est dit NF-p-CODIAG s'il existe une décomposition $\{\mathcal{F}^1, \ldots, \mathcal{F}^p\}$ de $\mathcal{F}$ telle que, pour tout $\mathcal{F}^j$, $(\mathcal{F}^j, \mathcal{H})$ est NF-CODIAG. La propriété F$\wedge$NF-p-CODIAG a été introduite pour caractériser la classe de langages diagnostiquables selon les objectifs **O1**, **O4** et **O5**. Le couple $(\mathcal{F}, \mathcal{H})$ est dit F$\wedge$NF-p-CODIAG s'il est à la fois F-p-CODIAG et NF-p-CODIAG. Comme les conditions de F-p-CODIAG et NF-p-CODIAG et F$\wedge$NF-p-CODIAG sont basées sur des décompositions de langages, généralement infinis, des versions plus fortes de ces conditions ont été définies pour résoudre ce problème. Ainsi, une condition a été introduite pour transformer les décompositions des langages $\mathcal{F}$ et $\mathcal{H}$ en des décompositions des ensembles d'états marqués d'AEF acceptant respectivement $\mathcal{F}$ et $\mathcal{H}$.

Dans la référence [Chakib et Khoumsi, 2009], l'approche multi-décisionnelle est étudiée dans un cadre générique dans le but de généraliser toutes les architectures fonctionnant en parallèle. Plus spécifiquement, on a étudié le cas où tous les diagnostiqueurs décentralisés $(Diag^j)_{j=1,\ldots,p}$ diagnostiquent le procédé selon l'architecture D&A, et leurs diagnostics sont fusionnés conjonctivement, c.à.d., $\mathbf{D} = \wedge$, afin que le diagnostic effectif respecte les objectifs **O1** et **O4**. L'architecture obtenue est notée $\wedge$-$\vee^p$.

Dans la référence [Chakib et Khoumsi, 2011a] (présentée au chapitre 5), on a étudié le diagnostic multi-décisionnel dans un cas plus général. Plus précisément, les architectures $\wedge$-$(Inf_{N_1}, \cdots, Inf_{N_p})$ et $\vee$-$(Inf_{N_1}, \cdots, Inf_{N_p})$ ont été définies pour décrire le cas où p diagnostiqueurs décentralisés $(Diag^j)_{j=1,\ldots,p}$ prennent leurs décisions selon la technique de l'inférence par ambiguïté, c.à.d., chaque $Diag^j$ prend sa décision selon l'architecture N_j-inférence, où N_j est un entier donné. Ces diagnostiqueurs décentralisés fonctionnent en parallèle et leurs diagnostics globaux sont fusionnés conjonctivement (c.à.d., $\mathbf{D} = \wedge$) dans le cas de l'architecture $\wedge$-$(Inf_{N_1}, \cdots, Inf_{N_p})$ ou disjonctivement (c.à.d., $\mathbf{D} = \vee$) dans le cas l'architecture $\vee$-$(Inf_{N_1}, \cdots, Inf_{N_p})$. Nous avons défini et étudié la notion de $\wedge$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-

CODIAG (resp., $\vee$-($Inf_{N_1}, \cdots, Inf_{N_p}$)-CODIAG) pour caractériser la classe de langages diagnostiquables selon l'architecture $\wedge$-($Inf_{N_1}, \cdots, Inf_{N_p}$) (resp., $\vee$-($Inf_{N_1}, \cdots, Inf_{N_p}$)). Dans le cas $\mathbf{D} = \wedge$, on a montré que l'existence d'une décomposition $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ de $\mathcal{H}$ telle que chaque $Diag^j$ respecte les objectifs **O1** et **O4** par rapport à $(\mathcal{F}, \mathcal{H}^j)$, est une condition nécessaire et suffisante pour que le diagnostiqueur multi-décisionnel $Diag = ((Diag^j)_{j \in J}, \wedge)$ respecte les objectifs **O1** et **O4** par rapport à $(\mathcal{F}, \mathcal{H})$. Le même résultat a été obtenu dans le cas $\mathbf{D} = \vee$ en considérant une décomposition $(\mathcal{F}^1, \ldots, \mathcal{F}^p)$ de $\mathcal{F}$.

On a proposé dans [Chakib et Khoumsi, 2009, 2011a; Khoumsi et Chakib, 2008a], dans le cas des langages réguliers, une méthode qui transforme le problème de la décomposition d'un langage régulier infini (en l'occurrence $\mathcal{F}$ ou $\mathcal{H}$) en un problème de décomposition de l'ensemble d'états marqués d'un AEF $\mathcal{A}$ acceptant $\mathcal{F}$ ou $\mathcal{H}$. On a ainsi défini une notion de coobservabilité multi-décisionnelle *forte* par rapport à $\mathcal{A}$, pour caractériser la classe de langages diagnostiquables sous les architectures multi-décisionnelles, en considérant des décompositions $\{\mathcal{F}^1, \ldots, \mathcal{F}^p\}$ de $\mathcal{F}$ ou $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ de $\mathcal{H}$, qui respectent la condition suivante : $\mathcal{F}^j$ (ou $\mathcal{H}^j$) contient toutes les traces menant à un ou plusieurs états marqués de l'AEF acceptant $\mathcal{F}$ (ou $\mathcal{H}$).

Un algorithme a été proposé dans la référence [Chakib et Khoumsi, 2011a] (présentée au chapitre 5), pour vérifier si $(\mathcal{F}, \mathcal{H})$ est $\vee$-($Inf_{N_1}, \cdots, Inf_{N_p}$)-CODIAG par rapport à un AEF $\mathcal{A}_{\mathcal{F}}$ donné acceptant $\mathcal{F}$, où p est un entier (inférieur ou égal au nombre des états marqués de l'AEF $\mathcal{A}_{\mathcal{F}}$) calculé par l'algorithme. La même procédure nous permet d'obtenir un algorithme qui vérifie si $(\mathcal{F}, \mathcal{H})$ est $\wedge$-($Inf_{N_1}, \cdots, Inf_{N_p}$)-CODIAG par rapport à un AEF $\mathcal{A}_{\mathcal{H}}$ donné acceptant $\mathcal{H}$, où p est un entier (inférieur ou égal au nombre des états marqués de l'AEF $\mathcal{A}_{\mathcal{H}}$) calculé par l'algorithme. Nous avons montré que la complexité de l'algorithme, dans le pire des cas, n'augmente pas avec le nombre des diagnostiqueurs décentralisés fonctionnant en parallèle. Plus précisément, la complexité de vérification de la diagnostiquabilité selon l'architecture multi-décisionnelle est, dans le pire des cas, du même ordre de grandeur que celle de la vérification de la diagnostiquabilité selon une des architectures décentralisées en parallèle qui constituent l'architecture multi-décisionnelle.

1.5.3 Organisation

Nous présentons une thèse par articles. Sur les quatre articles inclus dans la thèse, un est publié dans une conférence internationale avec comité de lecture, un article accepté dans un journal international avec comité de lecture, et deux articles sont soumis à des journaux internationaux avec comité de lecture. La thèse est structurée en deux parties. La première partie est constituée des chapitres 2 à 4 et contient nos contributions au

contrôle multi-décisionnel. La seconde partie est constituée du chapitre 5 et contient nos contributions au diagnostic multi-décisionnel. Considérons chaque chapitre.

1. Le chapitre 2 présente l'application du contrôle multi-décisionnel aux architectures D&A et C&P∨D&A. Ce chapitre est un article présenté à la conférence CASE'08 [Chakib et Khoumsi, 2008a].
2. Le chapitre 3 étudie en détail le cas général et générique du contrôle multi-décisionnel. Dans ce chapitre, on étudie aussi plus spécifiquement le contrôle multi-décisionnel pour mettre en parallèle des architectures par inférence. Ce chapitre est un article accepté au journal "IEEE Transactions on Automatic Control" [Chakib et Khoumsi, 2011b].
3. Le chapitre 4 présente un algorithme de vérification de la coobservabilité multi-décisionnelle dans le cas d'architectures par inférence en parallèle. Ce chapitre est un article soumis au journal "IEEE Transactions on Automatic Control" [Chakib et Khoumsi, 2011c].
4. Le chapitre 5 étudie le diagnostic multi-décisionnel pour mettre en parallèle des architectures par inférence. Un algorithme de vérification de la codiagnostiquabilité multi-décisionnelle dans le cas d'architectures par inférence en parallèle est présenté dans ce chapitre. Ce dernier est un article soumis au journal "Journal of Discrete Event Dynamic Systems : Theory & Applications" [Chakib et Khoumsi, 2011a].
5. La conclusion est présentée au chapitre 6 avec les contributions et les perspectives de nos travaux.

CHAPITRE 2

Architecture C&P∨D&A multi-décisionnelle pour le contrôle décentralisé de SED

Article : Hicham Chakib and Ahmed Khoumsi, *Multi-Decision C&P∨D&A Architecture for the Decentralized Control of Discrete Event Systems*, IEEE Conference on Automation Science and Engineering (CASE), Washington, États-Unis, 23-26 août, 2008.

Avant-propos

Auteurs et affiliation :

Hicham Chakib : étudiant au doctorat, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Ahmed Khoumsi : professeur, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Date d'acception : 25 mai 2008

État de l'acceptation : version finale publiée

Revue : Conference on Automation Science and Engineering (CASE)

Référence : IEEE Conference on Automation Science and Engineering (CASE), Washington, États-Unis, 23-26 août, p. 187-193, 2008

Titre français : Architecture C&P∨D&A multi-décisionnelle pour le contrôle décentralisé de SED

Contribution au document : Le principe du contrôle multi-décisionnel est basé sur le fait que chaque superviseur local émet un ensemble de décisions (que nous avons appelées micro-décisions) au lieu d'une seule décision. Dans la référence [Chakib et Khoumsi, 2008b], nous avions développé une première version de contrôle multi-décisionnel pour généraliser l'architecture de contrôle Conjonctive et Permissive (C&P). Maintenant, nous présentons une version améliorée pour généraliser les architectures C&P, D&A et C&P∨D&A. Nous proposons aussi dans ce chapitre, une méthode qui permet de transformer un AEF, pour qui la condition de coobservabilité multi-décisionnelle n'est pas satisfaite, en un autre AEF pour qui cette condition est satisfaite.

Résumé français : Dans cet article, nous étudions le contrôle multi-décisionnel, présenté dans la référence [Chakib et Khoumsi, 2008b], appliqué aux architectures C&P, D&A et C&P∨D&A. Le nouveau principe est basé sur le fait que chaque superviseur local émet un ensemble de micro-décisions, au lieu d'une seule décision. Pour chaque architecture, on définit la notion de m-coobservabilité, qui est utilisée pour caractériser la classe de langages réalisables par l'une des architectures multi-décisionnelles. Des variétés fortes et décidables de la m-coobservabilité ont été définies afin de résoudre le problème de la décomposition des langages infinis. La décidabilité de ces nouvelles variétés est assurée dans le cas des langages réguliers.

Note : À la suite des corrections demandées par les membres du jury, le contenu de cet article diffère de celui qui a été accepté.

Abstract : In this paper, we study the multi-decision decentralized control framework, introduced in [Chakib et Khoumsi, 2008b], within the context of C&P, D&A and C&P∨D&A architectures. The new framework is based on the fact that each supervisor issues a tuple of so-called micro-decisions instead of a single decision. For each architecture, we define the notion of m-coobservability, which is useful to characterize the class of achievable languages. Stronger and decidable varieties of m-coobservability are defined in order to cope with the problem of decomposing infinite languages encountered in verifying the m-coobservability. The decidability of the new varieties is ensured in the case of regular languages.

2.1 Introduction

This paper studies decentralized supervisory control (or more briefly, decentralized control) of discrete event systems (DES), where several supervisors cooperate according to their observations in order to determine the adequate enabling/disabling decisions to be applied to a *plant* so that it respects a given global *specification*.

The decentralized control of DES has been studied intensively [Cieslak *et al.*, 1988; Jiang et Kumar, 2000; Lin et Wonham, 1988a, 1990; Overkamp et van Schuppen, 2001; Prosser *et al.*, 1997; Ricker et Rudie, 2000; Rudie et Willems, 1995; Rudie et Wonham, 1992]. The first decentralized control architecture that has been proposed is referred to as the C&P (for *Conjunctive and Permissive*) architecture [Rudie et Wonham, 1992; Yoo et Lafortune, 2002a]. The paper [Prosser *et al.*, 1997] presents the first alternative in the study of decentralized control with different fusion rules.

In decentralized control, each supervisor takes local decisions, based on its local observation of the plant, consisting of disabling or enabling events. The local decisions taken by all the supervisors are fused in order to generate the actual decision that will be applied to the plant. With the C&P architecture : 1) each supervisor is *permissive*, since it locally *disables* an event iff it is certain that the event is rejected by the specification ; and 2) the local decisions of the supervisors are fused by *intersection* (or *conjunctively*), in order to generate the actual decision.

The authors of [Yoo et Lafortune, 2002a] have proposed the D&A (for *Disjunctive and Anti-permissive*) architecture which is complementary with the C&P one. In fact, the above two points 1 and 2 become : 1) each supervisor is *anti-permissive*, since it locally *enables* an event iff it is certain that the event is accepted by the specification ; and 2) the local decisions of the supervisors are fused by *union* (or *disjunctively*), in order to generate the actual decision.

The authors in [Yoo et Lafortune, 2002a] also propose a *general* architecture where the set Σ_c of controllable events is partitioned into two disjoint sets $\Sigma_{c,\wedge}$ and $\Sigma_{c,\vee}$, to which are applied the C&P and D&A architectures, respectively. The authors of [Yoo et Lafortune, 2002a] show that the class of languages achievable by the general architecture strictly includes those of the C&P and D&A architectures.

In a more recent work, the authors of [Yoo et Lafortune, 2004] propose a conditional architecture which generalizes the general architecture of [Yoo et Lafortune, 2002a]. In [Takai et Ushio, 2005], the authors propose a new decentralized supervisory control architecture using dynamic default control instead static default control. A knowledge-based concept has been introduced in [Ricker et Rudie, 2000, 2003] where the local decision of each supervisor is based on the evaluation of supervisors ambiguities. And the authors of [Kumar et Takai, 2005] propose an inference-based framework which generalizes the architectures of [Yoo et Lafortune, 2004] and [Ricker et Rudie, 2000, 2003].

The main purpose of the multi-decision control framework is to minimize the information lost when local supervisors transform an observed sequence into a local decision. Under the multi-decision control framework, each supervisor generates a multi-decision $(dec_1, dec_2, \ldots, dec_m)$ for each controllable event. The multi-decisions of all supervisors all fused in order to generate the actual decision which is applied to the plant. The multi-decision framework is intended to be applied to any of the decentralized control architectures cited above. In [Chakib et Khoumsi, 2008b], we studied the multi-decision control when applied to the C&P architecture of [Rudie et Wonham, 1992; Yoo et Lafor-

tune, 2002a]. In [Khoumsi et Chakib, 2008a], the framework has been applied to the fault diagnosis.

In this article, we generalize our multi-decision framework to the D&A architecture as well as the general (also called C&P∨D&A) architecture. Moreover, the problems related to the decomposition of infinite languages are solved in a more optimal way than in [Chakib et Khoumsi, 2008b].

The organization of the present paper is as follows. Notation and preliminaries are presented in Section 2.2. Section 2.3 summarizes [Chakib et Khoumsi, 2008b], that is, it presents the multi-decision control when applied to the C&P architecture. Specific varieties of C&P m-coobservable languages are defined in Section 2.4 in order to solve in a more optimal than in [Chakib et Khoumsi, 2008b] the problem of decomposing infinite languages encountered in the multi-decision control. In Section 2.5, we study the multi-decision control when applied to D&A architecture as well as the C&P∨D&A architecture of [Yoo et Lafortune, 2002a]. Finally the conclusion is presented in Section 2.6.

2.2 Notation and Preliminaries

We consider a DES plant modeled by an automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the set of states, Σ is the finite set of events, a partial function $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the set of marked states. Let Σ^* be the set of all finite traces of elements of Σ, including the empty trace ε. The transition function δ can be generalized to $\delta : Q \times \Sigma^* \to Q$ in the usual way. The generated and marked languages of G, are denoted by $\mathcal{L}$ and $\mathcal{L}_m$, respectively.

We say that a sequence $t \in \Sigma^*$ is a prefix of a sequence $s \in \Sigma^*$, denoted by $t \leq s$, if there exists a sequence $\lambda \in \Sigma^*$ such that $s = t\lambda$. We denote by $\overline{K}$ the set of all prefixes of sequences of a language $K \subseteq \Sigma^*$. K is said to be prefix-closed if $K = \overline{K}$. Given $K \subseteq \mathcal{L}_m$ and $\sigma \in \Sigma$, we denote by $\mathcal{E}_\sigma(K) = \{s \in \overline{K} \mid s\sigma \in \overline{K}\}$ the set of sequences of $\overline{K}$ after which σ is permitted by $\overline{K}$. Similarly, we define by $\mathcal{D}_\sigma(K) = \{s \in \overline{K} \mid s\sigma \in \mathcal{L} \backslash \overline{K}\}$ the set of sequences of $\overline{K}$ after which σ is accepted by $\mathcal{L}$ and forbidden by $\overline{K}$.

The problem of decentralized control treated in this paper is performed by n supervisors $(Sup_i)_{1 \leq n}$. Each Sup_i has its own set of observable events $\Sigma_{o,i}$ and own set of controllable events $\Sigma_{c,i}$. The supervisors together can then observe $\Sigma_o = \Sigma_{o,1} \cup \cdots \cup \Sigma_{o,n}$ and control $\Sigma_c = \Sigma_{c,1} \cup \cdots \cup \Sigma_{c,n}$. The sets of unobservable and uncontrollable events are denoted, respectively, by $\Sigma_{uo} = \Sigma \setminus \Sigma_o$ and $\Sigma_{uc} = \Sigma \backslash \Sigma_c$. We denote by $I = \{1, \ldots, n\}$ the indexing

set of all supervisors. For any $\sigma \in \Sigma_c$, we define by $I_\sigma = \{i \in I | \sigma \in \Sigma_{c,i}\}$ the indexing set of supervisors controlling σ. We denote by $P_i : \Sigma^* \to \Sigma^*_{o,i}$ the natural projection that hides the events of $\Sigma \setminus \Sigma_{o,i}$ from any sequence $s \in \Sigma^*$.

A language $K \subseteq \mathcal{L}_m$ is said $\mathcal{L}_m$-closed if $K = \overline{K} \cap \mathcal{L}_m$, and is said controllable w.r.t. $\mathcal{L}$ and Σ_{uc} if $\overline{K}\Sigma_{uc} \cap \mathcal{L} \subseteq \overline{K}$.

The notion of C&P (resp., D&A) co-observability has been defined formally for the C&P (resp., D&A) architecture in [Yoo et Lafortune, 2002a]. For the purpose of our study, let us define them equivalently by using $\mathcal{E}_\sigma(K)$ and $\mathcal{D}_\sigma(K)$ as follows :

Definition 2.2.1 *A language $K \subseteq \mathcal{L}_m$ is said to be C&P co-observable w.r.t. $\mathcal{L}$, $\Sigma_{o,1}$, $\Sigma_{c,1}$, $\cdots$, $\Sigma_{o,n}$, $\Sigma_{c,n}$ iff $\forall s \in \mathcal{D}_\sigma(K)$ and $\forall \sigma \in \Sigma_c$, $\exists i \in I_\sigma$ such that :*

$$[(P_i^{-1}P_i(s) \cap \mathcal{E}_\sigma(K) = \emptyset] \wedge [\sigma \in \Sigma_{c,i}]. \tag{2.1}$$

Definition 2.2.2 *A language $K \subseteq \mathcal{L}_m$ is said to be D&A co-observable w.r.t. $\mathcal{L}$, $\Sigma_{o,1}$, $\Sigma_{c,1}$, $\cdots$, $\Sigma_{o,n}$, $\Sigma_{c,n}$ iff $\forall s \in \mathcal{E}_\sigma(K)$ and $\forall \sigma \in \Sigma_c$, $\exists i \in I_\sigma$ such that :*

$$[(P_i^{-1}P_i(s) \cap \mathcal{D}_\sigma(K) = \emptyset] \wedge [\sigma \in \Sigma_{c,i}]. \tag{2.2}$$

2.3 C&P Multi-Decision Control and C&P Multi-Coobservability

In this section, we summarize [Chakib et Khoumsi, 2008b] by introducing the multi-decision control framework when applied to the C&P architecture of [Rudie et Wonham, 1992; Yoo et Lafortune, 2002a]. We consider a plant modeled by $\mathcal{L}$ and $\mathcal{L}_m$, and a specification modeled by K, and we assume that for each $\sigma \in \Sigma_c$, we are given a partition $(\mathcal{E}^1_\sigma, \mathcal{E}^2_\sigma, \ldots, \mathcal{E}^{m_\sigma}_\sigma)$ of $\mathcal{E}_\sigma(K)$. That is, $\mathcal{E}_\sigma(K) = \mathcal{E}^1_\sigma \cup \mathcal{E}^2_\sigma \cup \cdots \cup \mathcal{E}^{m_\sigma}_\sigma$ and the subsets $\mathcal{E}^j_\sigma$ are non-empty and disjoint with each other. Note that these subsets $\mathcal{E}^j_\sigma$ depend on K and σ. For simplicity of notation, K is not indicated in $\mathcal{E}^j_\sigma$.

2.3.1 Multi-Decision Control

Each supervisor Sup_i $(i = 1, \ldots, n)$ generates tuples of so-called micro-decisions instead of single decisions. Since we study the multi-decision control when it is applied to the C&P architecture, we obtain what is called : C&P multi-decision architecture. In the C&P multi-decision architecture, each supervisor Sup_i generates a m_σ-tuple $(Sup^1_i(P_i(s), \sigma),$

$\cdots, Sup_i^{m_\sigma}(P_i(s),\sigma))$, where each $Sup_i^j(P_i(s),\sigma)$ is an enable/disable decision related to $\mathcal{E}_\sigma^j$ and is called the jth micro-decision of Sup_i. The objective is that for each $j \in \{1,\ldots,m_\sigma\}$, the jth micro-decisions $Sup_i^j(P_i(s),\sigma)$ $(i \in \{1,\ldots,n\})$ issued by the n supervisors will be combined such that the resulting decision $S^j(s,\sigma)$ enables σ *if*[1] it is permitted in $\mathcal{E}_\sigma^j$. Formally, according to the conjunctive and permissive rules, we have :

$$\begin{aligned}&\forall j \in \{1,\ldots,m_\sigma\}, \forall i \in \{1,\ldots,n\} :\\ &Sup_i^j(P_i(s),\sigma)=1 \Leftrightarrow [P_i^{-1}P_i(s) \cap \mathcal{E}_\sigma^j \neq \emptyset] \vee [\sigma \in \Sigma \backslash \Sigma_{c,i}]\end{aligned} \tag{2.3}$$

$$\forall j \in \{1,\ldots,m_\sigma\} : S^j(s,\sigma) = \bigwedge_{1\leq i\leq n} Sup_i^j(P_i(s),\sigma) \tag{2.4}$$

The global decision $Sup(s,\sigma)$ issued by the fusion system that will be applied to the plant is computed by :

$$Sup(s,\sigma) = \bigvee_{1\leq j\leq m_\sigma} S^j(s,\sigma) \tag{2.5}$$

Note that for each $\sigma \in \Sigma_c$, we have used a value m_σ. Let us denote by m the biggest of these values m_σ considering all the events of Σ_c. The obtained supervisor will be denoted *Sup* and called C&P m-decision supervisor. From Eqs. 2.4 and 2.5, the C&P multi-decision architecture can be schematized as shown in Fig. 2.1.

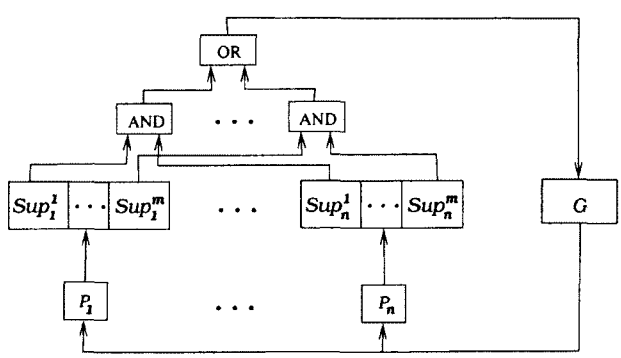

Figure 2.1 The C&P multi-decision architecture

The prefix-closed language $\mathcal{L}(Sup/G)$ generated by the system G under the control of the C&P m-decision supervisor *Sup* can be defined as follows :

1. $\epsilon \in \mathcal{L}(Sup/G)$,
2. $[s \in \mathcal{L}(Sup/G)] \wedge [s\sigma \in \mathcal{L}] \wedge [Sup(s,\sigma)=1] \Leftrightarrow s\sigma \in \mathcal{L}(Sup/G)$.

[1] The *only if* does not necessarily hold.

The corresponding marked language is defined by $\mathcal{L}_m(Sup/G) = \mathcal{L}(Sup/G) \cap \mathcal{L}_m$. A C&P m-decision supervisor Sup is called nonblocking if $\overline{\mathcal{L}_m(Sup/G)} = \mathcal{L}(Sup/G)$.

2.3.2 Class of C&P m-Coobservable Languages

In this subsection, we present necessary and sufficient conditions for the existence of supervisors that achieve a given desired specification in the context of C&P multi-decision control. For this purpose, we need to introduce the notion of C&P *m-coobservability*

Definition 2.3.1 *Given an integer $m \geq 1$, a language $K \subseteq \mathcal{L}_m$ is said to be C&P m-coobservable w.r.t. $\mathcal{L}$, $\Sigma_{o,1}$, $\Sigma_{c,1}$, $\cdots$, $\Sigma_{o,n}$, $\Sigma_{c,n}$, if for each $\sigma \in \Sigma_c$, there exists an integer m_σ (s.t., $1 \leq m_\sigma \leq m$), and a partition $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^{m_\sigma})$ of $\mathcal{E}_\sigma(K)$ such that, $\forall j \in \{1, \ldots, m_\sigma\}$:*

$$\bigcap_{i \in I_\sigma} P_i^{-1} P_i(\mathcal{E}_\sigma^j) \cap \mathcal{D}_\sigma(K) = \emptyset \tag{2.6}$$

For brevity, in the sequel we will omit the expression "w.r.t. $\mathcal{L}$, $\Sigma_{o,1}$, $\Sigma_{c,1}$, $\cdots$, $\Sigma_{o,n}$, $\Sigma_{c,n}$".

Let us now define the C&P *multi*-coobservabilty, a weaker notion of C&P m-coobservability.

Definition 2.3.2 *A language $K \subseteq \mathcal{L}_m$ is said to be C&P multi-coobservable if there exists an integer $m \geq 1$ such that K is C&P m-coobservable.*

The next Theorem relates C&P m-coobservability of a specification K with existence of C&P m-decision supervisor Sup that can control the plant so that it respects K.

Theorem 2.3.1 *Consider $K \subseteq \mathcal{L}_m$ where $K \neq \emptyset$ and an integer $m \geq 1$. The following two points are equivalent :*

1. *There exists a partition $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^{m_\sigma})$ of every $\mathcal{E}_\sigma(K)$ (where $\sigma \in \Sigma_c$ and $m_\sigma \leq m$) s.t. the C&P m-decision supervisor Sup defined by Eqs. (2.3)-(2.5) is nonblocking and satisfying $\mathcal{L}_m(Sup/G) = K$ and $\mathcal{L}(Sup/G) = \overline{K}$,*

2. *K is controllable w.r.t $\mathcal{L}$ and Σ_{uc}, C&P m-coobservable and $\mathcal{L}_m$-closed.*

The gain obtained with the multi-decision is demonstrated in [Chakib et Khoumsi, 2008b]. In fact, it is proved that given G, $\Sigma_{o,i}$ and $\Sigma_{c,i}$, $i = 1, \ldots, n$, the class of languages obtained under the C&P multi-decision architecture strictly includes the class obtained with the C&P architecture of [Rudie et Wonham, 1992; Yoo et Lafortune, 2002a].

2.4 Decidable and Stronger Varieties of C&P Multi-Coobservability

Checking the C&P multi-coobservability of $K \subseteq \mathcal{L}_m$ requires to determine (for every $\sigma \in \Sigma_c$) whether there exists a partition $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^{m_\sigma})$ of $\mathcal{E}_\sigma(K)$ that respects Eq. 2.6. In general, the number of partitions of $\mathcal{E}_\sigma(K)$ is infinite, and thus, it may be necessary to check an infinite number of partitions before to determine whether K is C&P m-coobservable. A solution to tackle this problem is to consider stronger varieties of the C&P m-coobservability where the number of partitions to be checked is finite. For this purpose, we use for every $\sigma \in \Sigma_c$:

Requirement 1 : define an equivalence relation between the sequences of $\mathcal{E}_\sigma(K)$ such that the number of equivalence classes is finite ;

Requirement 2 : require that every $\mathcal{E}_\sigma^j$ contains one or several equivalence classes.

An example of equivalence relation is the Nerode relation $\mathcal{N}_\mathcal{E}^\sigma \subseteq \mathcal{E}_\sigma(K) \times \mathcal{E}_\sigma(K)$ defined as follows.

$$(s,t) \in \mathcal{N}_\mathcal{E}^\sigma \Leftrightarrow \forall \lambda \in \Sigma^* : s\lambda \in \mathcal{E}_\sigma(K) \Leftrightarrow t\lambda \in \mathcal{E}_\sigma(K).$$

We denote by $\mathcal{N}_\mathcal{E} = (\mathcal{N}_\mathcal{E}^\sigma)_{\sigma \in \Sigma_c}$ the tuple of Nerode relations related to the controllable events. In particular, when $\Sigma_c = \{\sigma\}$, we have $\mathcal{N}_\mathcal{E} = \mathcal{N}_\mathcal{E}^\sigma$. It is worth noting that K and $\mathcal{E}_\sigma(K)$ are regular languages, and thus, can be defined by finite state automata. Let $\mathcal{A}_{\mathcal{N}_\mathcal{E}^\sigma}$ be the minimal automaton recognizing $\mathcal{E}_\sigma(K)$. This automaton is noted with the index $\mathcal{N}_\mathcal{E}^\sigma$ because each of its marked states represents an equivalence class of $\mathcal{N}_\mathcal{E}^\sigma$. If $\mathcal{N}_\mathcal{E}^\sigma$ is the selected equivalence relation for Requirement 1, we deduce from Requirement 2 that each $\mathcal{E}_\sigma^j$ is defined by one or several marked states of $\mathcal{A}_{\mathcal{N}_\mathcal{E}^\sigma}$. Therefore, the finite number of states implies that we have to check a finite number of partitions.

Notation 2.4.1 *Given two equivalence relations $\mathcal{R}$ and $\mathcal{V}$, $\mathcal{R}$ is said to be stronger than $\mathcal{V}$, denoted $\mathcal{R} \leq \mathcal{V}$, if :*

$$(s,t) \in \mathcal{R} \Rightarrow (s,t) \in \mathcal{V}.$$

In order to increase the number of possible partitions, we need to define, for every $\sigma \in \Sigma_c$, an equivalence relation $\mathcal{R}_\mathcal{E}^\sigma \subseteq \mathcal{E}_\sigma(K) \times \mathcal{E}_\sigma(K)$ which is stronger than $\mathcal{N}_\mathcal{E}^\sigma$. Hence, each equivalence class of $\mathcal{R}_\mathcal{E}^\sigma$ is a subset of an equivalence of $\mathcal{N}_\mathcal{E}^\sigma$. We thus define $\mathcal{A}_{\mathcal{R}_\mathcal{E}^\sigma}$ as the (non-minimal) automaton which recognizes $\mathcal{E}_\sigma(K)$ and whose each marked state represents an equivalence class of $\mathcal{R}_\mathcal{E}^\sigma$. The interest of using $\mathcal{R}_\mathcal{E}^\sigma \leq \mathcal{N}_\mathcal{E}^\sigma$ is that Requirements 1 and 2 permit more partitions of $\mathcal{E}_\sigma(K)$ with $\mathcal{R}_\mathcal{E}^\sigma$ than with $\mathcal{N}_\mathcal{E}^\sigma$.

Hereafter, we define by $\mathcal{R}_\mathcal{E} = (\mathcal{R}_\mathcal{E}^\sigma)_{\sigma\in\Sigma_c}$ a tuple of equivalence relations related to the controllable events such that $\mathcal{R}_\mathcal{E}^\sigma \leq \mathcal{N}_\mathcal{E}^\sigma$. In particular, when $\Sigma_c = \{\sigma\}$, we have $\mathcal{R}_\mathcal{E} = \mathcal{R}_\mathcal{E}^\sigma$. Let us now define the C&P m-coobservability w.r.t. $\mathcal{R}_\mathcal{E}$, a stronger variety of the C&P m-coobservability :

Definition 2.4.1 *Consider $K \subseteq \mathcal{L}_m$ and a tuple of finite equivalence relations $\mathcal{R}_\mathcal{E} = (\mathcal{R}_\mathcal{E}^\sigma)_{\sigma\in\Sigma_c}$. K is said to be C&P m-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$ if, for each $\sigma \in \Sigma_c$, there exists an integer m_σ (s.t., $1 \leq m_\sigma \leq m$) and a partition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^{m_\sigma}\}$ of $\mathcal{E}_\sigma(K)$ such that : $\forall j \in \{1, \ldots, m_\sigma\}$, $\mathcal{E}_\sigma^j$ satisfies Eq. 2.6 and :*

$$\forall (s,t) \in \mathcal{R}_\mathcal{E}^\sigma \quad s \in \mathcal{E}_\sigma^j \Leftrightarrow t \in \mathcal{E}_\sigma^j. \tag{2.7}$$

Definition 2.4.2 *Given a tuple of finite equivalence relations $\mathcal{R}_\mathcal{E} = (\mathcal{R}_\mathcal{E}^\sigma)_{\sigma\in\Sigma_c}$, a language $K \subseteq \mathcal{L}_m$ is said to be C&P multi-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$ if there exists an integer $m \geq 1$ such that K is C&P m-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$.*

Let $E_{\mathcal{R}_\mathcal{E}^\sigma}$ denote the set of equivalence classes of $\mathcal{R}_\mathcal{E}^\sigma$. The next Proposition sets a necessary and sufficient condition for a language to be C&P multi-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$.

Proposition 2.4.1 *Given a tuple of finite equivalence relations $\mathcal{R}_\mathcal{E} = (\mathcal{R}_\mathcal{E}^\sigma)_{\sigma\in\Sigma_c}$, a language $K \subseteq \mathcal{L}_m$ is C&P multi-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$ iff, $\forall \sigma \in \Sigma_c$, $\forall A \in E_{\mathcal{R}_\mathcal{E}^\sigma}$:*

$$\bigcap_{i\in I_\sigma} P_i^{-1}P_i(A) \cap \mathcal{D}_\sigma(K) = \emptyset. \tag{2.8}$$

From Proposition 2.4.1, we deduce the following corollary.

Corollary 2.4.1 *Let $\mathcal{V}_\mathcal{E} = (\mathcal{V}_\mathcal{E}^\sigma)_{\sigma\in\Sigma_c}$ be a tuple of finite equivalence relations such that, for every $\sigma \in \Sigma_c$, $\mathcal{R}_\mathcal{E}^\sigma \leq \mathcal{V}_\mathcal{E}^\sigma \leq \mathcal{N}_\mathcal{E}^\sigma$. If $K \subseteq \mathcal{L}_m$ is C&P multi-coobservable w.r.t. $\mathcal{V}_\mathcal{E}$, then K is C&P multi-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$.*

The following example proves that the converse of Corollary 2.4.1 is not true.

Example 2.4.1 *We consider the prefix-closed plant G of Fig 2.2, the prefix-closed specification $K \subseteq \mathcal{L}$ is obtained by forbidding σ at states 6 and 7 (represented by dashed transitions). We take $\Sigma_{o,1} = \{a_1, b_1, c_1, d_1\}$, $\Sigma_{o,2} = \{a_2, b_2, c_2, d_2\}$ and $\Sigma_{c,1} = \Sigma_{c,2} = \{\sigma\}$ The (minimal) automaton $\mathcal{A}_{\mathcal{N}_\mathcal{E}^\sigma}$ is obtained by marking only State 8. Therefore, $X = \{b_1a_2, c_2a_1a_2, c_2a_1b_2d_1, b_1b_2d_1, a_1a_2d_1, d_2b_1a_2d_1\} \in E_{\mathcal{N}_\mathcal{E}^\sigma}$ is an equivalence class of $\mathcal{N}_\mathcal{E}^\sigma$ corresponding to State 8, and we have $b_1b_2 \in \bigcap_{i=1,2} P_i^{-1}P_i(X) \cap \mathcal{D}_\sigma(K) \neq \emptyset$. We deduce from Proposition 2.4.1 that K is not C&P multi-coobservable w.r.t. $\mathcal{N}_\mathcal{E}$. And from Theorem*

2.3.1, we deduce that the C&P m-decision supervisor Sup defined by Eqs. (2.3)-(2.5) is blocking or such that $\mathcal{L}_m(Sup/G) \neq K$ *or* $\mathcal{L}(Sup/G) \neq \overline{K}$.

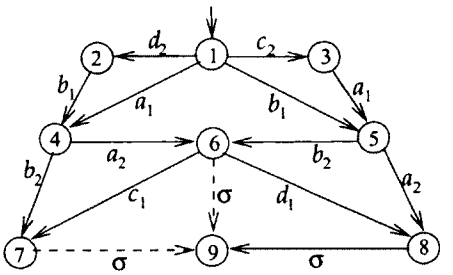

Figure 2.2 Plant and specification. The latter is obtained by forbidding the dashed transitions

Let us now represent the same specification by the non-minimal automaton of Fig. 2.3. Note that the dashed transitions (from states F, I, K and L) are not contained in $\overline{K}$. $\mathcal{E}_\sigma(K)$ *contains the sequences leading to states M, G and N. Let us consider the relation* $\mathcal{R}_\mathcal{E}^\sigma \leq \mathcal{N}_\mathcal{E}^\sigma$ *having three equivalence classes* Y_1, Y_2 *and* Y_3 *corresponding, respectively, to the three states M, G and N. These three equivalence classes are defined as follows :* $Y_1 = \{d_2b_1a_2d_1, a_1a_2d_1, b_1b_2d_1, c_2a_1b_2d_1\}$, $Y_2 = \{b_1a_2\}$ *and* $Y_3 = \{c_2a_1a_2\}$. *It is easy to check that* $\bigcap_{i=1,2} P_i^{-1}P_i(Y_j) \cap \mathcal{D}_\sigma(K) = \emptyset$ *for all* $j \in \{1,2,3\}$. *Therefore, by Proposition 2.4.1, K is C&P 3-coobservable w.r.t.* $\mathcal{R}_\mathcal{E}$, *and thus, K is C&P multi-coobservable w.r.t* $\mathcal{R}_\mathcal{E}$.

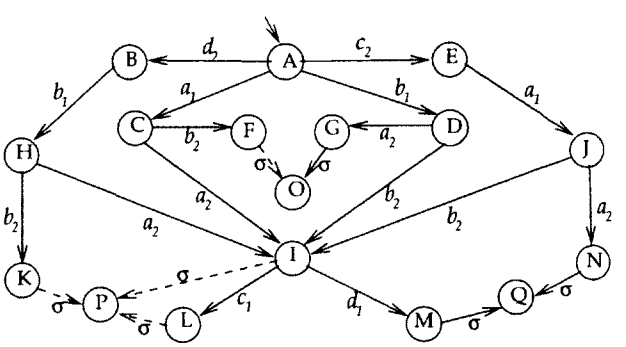

Figure 2.3 Example of a specification having the same language as the specification of Fig. 2.2

C&P m-coobservability w.r.t. $\mathcal{R}_\mathcal{E}$ is *decidable* (because the number of partitions to be checked is *finite*) and permits to define the following sufficient (and not necessary in general) condition for the existence of supervisors.

Proposition 2.4.2 *Consider* $K \subseteq \mathcal{L}_m$ *where* $K \neq \emptyset$ *and a tuple of finite equivalence relations* $\mathcal{R}_\mathcal{E} = (\mathcal{R}_\mathcal{E}^\sigma)_{\sigma \in \Sigma_c}$. *If K is : 1) controllable w.r.t.* $\mathcal{L}$ *and* Σ_{uc}, *2) C&P multi-*

coobservable w.r.t. $\mathcal{R}_{\mathcal{E}}$, and 3) $\mathcal{L}_m$-closed, then there exists a partition $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^{m_\sigma})$ of every $\mathcal{E}_\sigma(K)$ (where $\sigma \in \Sigma_c$ and $m_\sigma \leq m$) s.t. the C&P m-decision supervisor Sup defined by Eqs. (2.3)-(2.5) is nonblocking and satisfying $\mathcal{L}_m(Sup/G) = K$ and $\mathcal{L}(Sup/G) = \overline{K}$.

Now, for any $\sigma \in \Sigma_c$ and any $\mathcal{R}_{\mathcal{E}}^\sigma \leq \mathcal{N}_{\mathcal{E}}^\sigma$, we define an equivalence relation $\mathcal{P}_{\mathcal{R}_{\mathcal{E}}^\sigma}$ which will be useful further to define a stronger relation than $\mathcal{R}_{\mathcal{E}}^\sigma$. We define the equivalence relation $\mathcal{P}_{\mathcal{R}_{\mathcal{E}}^\sigma} \subseteq \mathcal{E}_\sigma(K) \times \mathcal{E}_\sigma(K)$ as follows :

$$(s,t) \in \mathcal{P}_{\mathcal{R}_{\mathcal{E}}^\sigma} \Leftrightarrow \forall A \in E_{\mathcal{R}_{\mathcal{E}}^\sigma}, \forall i \in I_\sigma, \text{ we have :}$$
$$P_i^{-1}P_i(s) \cap A \neq \emptyset \Leftrightarrow P_i^{-1}P_i(t) \cap A \neq \emptyset.$$

It is easy to show that $\mathcal{P}_{\mathcal{R}_{\mathcal{E}}^\sigma}$ is an equivalence relation from the fact that $\mathcal{R}_{\mathcal{E}}^\sigma$ is an equivalence relation. Intuitively, if two sequences s and t are in relation under $\mathcal{P}_{\mathcal{R}_{\mathcal{E}}^\sigma}$, then for any $i \in I_\sigma$, the subset of states $y \subseteq Q_{\mathcal{R}_{\mathcal{E}}^\sigma}$ (where $Q_{\mathcal{R}_{\mathcal{E}}^\sigma}$ is the set of states of the automaton $\mathcal{E}_{\mathcal{R}_{\mathcal{E}}^\sigma}$) not distinguishable by the supervisor Sup_i is the same after the execution of s or t. Hence, by considering Sup_i as an observer of G, $P_i(s)$ and $P_i(t)$ lead to the same state in Sup_i for all $i \in I_\sigma$.

In the following proposition, we give a necessary (and not sufficient) condition for a language to be C&P multi-coobservable w.r.t. $\mathcal{R}_{\mathcal{E}}$.

Proposition 2.4.3 *If a language $K \subseteq \mathcal{L}_m$ is C&P multi-coobservable w.r.t. $\mathcal{R}_{\mathcal{E}}$, then, $\forall X \in E_{\mathcal{P}_{\mathcal{R}_{\mathcal{E}}^\sigma}}$, we have :*

$$X \cap \mathcal{E}_\sigma(K) \neq \emptyset \Rightarrow X \cap \mathcal{D}_\sigma(K) = \emptyset.$$

Now, for any equivalence relation $\mathcal{R}_{\mathcal{E}}^\sigma \leq \mathcal{N}_{\mathcal{E}}^\sigma$, we define an equivalence relation $\mathcal{M}_{\mathcal{R}_{\mathcal{E}}^\sigma}$ such that $\mathcal{M}_{\mathcal{R}_{\mathcal{E}}^\sigma} \leq \mathcal{R}_{\mathcal{E}}^\sigma$. The motivation is to obtain more possibilities for partitioning $\mathcal{E}_\sigma(K)$ than with $\mathcal{R}_{\mathcal{E}}^\sigma$, and thus, we can achieve more languages (see Corollary 2.4.1). Given a language $K \subseteq \mathcal{L}_m$, for any $\mathcal{R}_{\mathcal{E}}^\sigma \leq \mathcal{N}_{\mathcal{E}}^\sigma$, we define $\mathcal{M}_{\mathcal{R}_{\mathcal{E}}^\sigma}$ as follows : $\forall \sigma \in \Sigma_c$,

$$(s,t) \in \mathcal{M}_{\mathcal{R}_{\mathcal{E}}^\sigma} \Leftrightarrow (s,t) \in \mathcal{R}_{\mathcal{E}}^\sigma \text{ and } (s,t) \in \mathcal{P}_{\mathcal{R}_{\mathcal{E}}^\sigma}.$$

Hereafter, a tuple of relations $(\mathcal{M}_{\mathcal{R}_{\mathcal{E}}^\sigma})_{\sigma \in \Sigma_c}$ is denoted by $\mathcal{M}_{\mathcal{R}_{\mathcal{E}}}$. From Corollary 2.4.1 and the fact that $\mathcal{M}_{\mathcal{R}_{\mathcal{E}}^\sigma} \leq \mathcal{R}_{\mathcal{E}}^\sigma$, we state the next corollary, which justifies the interest of defining $\mathcal{M}_{\mathcal{R}_{\mathcal{E}}^\sigma}$.

Corollary 2.4.2 *Consider a tuple of finite equivalence relations $\mathcal{R}_{\mathcal{E}} = (\mathcal{R}^{\sigma}_{\mathcal{E}})_{\sigma\in\Sigma_c}$. If a language $K \subseteq \mathcal{L}$ is C&P multi-coobservable w.r.t. $\mathcal{R}_{\mathcal{E}}$, then it is C&P multi-coobservable w.r.t. $\mathcal{M}_{\mathcal{R}_{\mathcal{E}}}$.*

Example 2.4.1 proves that the converse of Corollary 2.4.2 is not true, since we can easily check that the relation $\mathcal{R}^{\sigma}_{\mathcal{E}}$ used is exactly $\mathcal{M}_{\mathcal{N}^{\sigma}_{\mathcal{E}}}$.

Given any equivalence relation $\mathcal{R}^{\sigma}_{\mathcal{E}} \leq \mathcal{N}^{\sigma}_{\mathcal{E}}$ defined in $\mathcal{E}_{\sigma}(K)$, we have developed an automata-based algorithm which checks C&P multi-coobservabilities w.r.t. $\mathcal{R}_{\mathcal{E}}$ and $\mathcal{M}_{\mathcal{R}_{\mathcal{E}}}$. This algorithm is not presented here for space limit.

Note that in [Chakib et Khoumsi, 2008b] we have tackled the problem of decomposing infinite languages by using a single equivalence relation $\mathcal{R}$ which is defined in $\overline{K}$ and does not depend on σ. Consequently, the approach proposed here is better than [Chakib et Khoumsi, 2008b] in the following sense : The existence of a single relation $\mathcal{R}$ on $\overline{K}$ for which we have C&P multi-coobservability w.r.t. $\mathcal{R}$, guarantees the existence of a tuple $\mathcal{R}_{\mathcal{E}} = (\mathcal{R}^{\sigma}_{\mathcal{E}})_{\sigma\in\Sigma_c}$ for which we have C&P multi-coobservability w.r.t. $\mathcal{R}_{\mathcal{E}}$. But the converse is not true.

2.5 C&P∨D&A Multi-Decision Control Framework

In this section, we show how the multi-decision is applied to the D&A architecture, and then we study its application to the C&P∨D&A (also called *general*) architecture, which comprises the C&P and D&A architectures.

2.5.1 Class of D&A multi-Coobservable Languages

The D&A architecture is dual to the C&P one, in the sense that we obtain one from the other by essentially switching between *enable* and *disable* and between $\mathcal{E}_{\sigma}(K)$ and $\mathcal{D}_{\sigma}(K)$. And the D&A multi-decision architecture will be obtained by partitioning $\mathcal{D}_{\sigma}(K)$ instead of $\mathcal{E}_{\sigma}(K)$. Hence, we assume that for every $\sigma \in \Sigma_c$, we are given a partition $(\mathcal{D}^1_{\sigma}, \mathcal{D}^2_{\sigma}, \ldots, \mathcal{D}^{m_{\sigma}}_{\sigma})$ of $\mathcal{D}_{\sigma}(K)$.

In the D&A multi-decision architecture, for every $\sigma \in \Sigma_c$, each Sup_i generates m_{σ} micro-decisions $(Sup^1_i(P_i(s), \sigma), \ldots, Sup^{m_{\sigma}}_i(P_i(s), \sigma))$. The anti-permissive decision strategy of the D&A architecture implies that the jth micro-decision of Sup_i disables $\sigma \in \Sigma_c$ iff : the observation of Sup_i may correspond to an execution of $\mathcal{D}^j_{\sigma}$. Formally : $\forall j \in \{1, \ldots, m_{\sigma}\}, \forall i \in$

$\{1, \ldots, n\}$,

$$Sup_i^j(P_i(s), \sigma) = 0 \Leftrightarrow [[P_i^{-1}P_i(s) \cap \mathcal{D}_\sigma^j \neq \emptyset] \wedge [\sigma \in \Sigma_{c,i}]] \vee [\sigma \in \Sigma_c \backslash \Sigma_{c,i}] \tag{2.9}$$

The disjunctive fusion rule of the D&A architecture implies that the decision $S^j(s, \sigma)$, resulting from the combination of the n j^{th} micro-decisions $Sup_i^j(P_i(s), \sigma)$ ($i \in \{1, \ldots, n\}$), is computed as follows :

$$\forall j \in \{1, \ldots, m_\sigma\} : S^j(s, \sigma) = \bigvee_{1 \leq i \leq n} Sup_i^j(P_i(s), \sigma) \tag{2.10}$$

The global decision $Sup_\vee(s, \sigma)$ issued by the fusion system that will be applied to the plant is computed by :

$$Sup_\vee(s, \sigma) = \bigwedge_{1 \leq j \leq m_\sigma} S^j(s, \sigma) \tag{2.11}$$

Equation 2.11 expresses the fact that an event $\sigma \in \Sigma_c$ is disabled, i.e. $Sup_\vee(s, \sigma) = 0$, if at least one of the micro-decisions $S^j(s, \sigma)$ disables it. While σ is enabled if all the micro-decisions enable it. Note that, the D&A multi-decision architecture can be schematized as the C&P multi-decision architecture shown in Fig. 2.1 by switching the modules OR and AND

Next, we will directly define D&A m-coobservability w.r.t $\mathcal{R}_\mathcal{D} = (\mathcal{R}_\mathcal{D}^\sigma)_{\sigma \in \Sigma_c}$ such that $\mathcal{R}_\mathcal{D}^\sigma \leq \mathcal{N}_\mathcal{D}^\sigma$. Note that $\mathcal{N}_\mathcal{D}^\sigma$ and $\mathcal{R}_\mathcal{D}^\sigma$ are defined like $\mathcal{N}_\mathcal{E}^\sigma$ and $\mathcal{R}_\mathcal{E}^\sigma$, but over $\mathcal{D}_\sigma(K)$ instead of $\mathcal{E}_\sigma(K)$. We define $\mathcal{A}_{\mathcal{R}_\mathcal{D}^\sigma}$ as the (non-minimal) automaton which recognizes $\mathcal{D}_\sigma(K)$ and whose each marked state represents an equivalence class of $\mathcal{R}_\mathcal{D}^\sigma$.

Definition 2.5.1 *Consider $K \subseteq \mathcal{L}_m$, a tuple of finite equivalence relations $\mathcal{R}_\mathcal{D} = (\mathcal{R}_\mathcal{D}^\sigma)_{\sigma \in \Sigma_c}$ and an integer $m \geq 1$. K is said to be D&A m-coobservable w.r.t. $\mathcal{R}_\mathcal{D}$, $\mathcal{L}$, $\Sigma_{o,1}$, $\Sigma_{c,1}$, $\cdots$, $\Sigma_{o,n}$, $\Sigma_{c,n}$, if, for each $\sigma \in \Sigma_c$, there exists an integer m_σ (s.t., $1 \leq m_\sigma \leq m$) and a partition $\{\mathcal{D}_\sigma^1, \ldots, \mathcal{D}_\sigma^{m_\sigma}\}$ of $\mathcal{D}_\sigma(K)$ such that, $\forall j \in \{1, \ldots, m_\sigma\}$, we have :*

$$\bigcap_{i \in I_\sigma} P_i^{-1}P_i(\mathcal{D}_\sigma^j) \cap \mathcal{E}_\sigma(K) = \emptyset, \tag{2.12}$$

$$\forall (s, t) \in \mathcal{R}_\mathcal{D}^\sigma : s \in \mathcal{D}_\sigma^j \Leftrightarrow t \in \mathcal{D}_\sigma^j. \tag{2.13}$$

For simplicity, the term "w.r.t $\mathcal{L}$, $\Sigma_{o,1}$, $\Sigma_{c,1}$, $\cdots$, $\Sigma_{o,n}$, $\Sigma_{c,n}$" can be omitted.

Definition 2.5.2 *Consider a language $K \subseteq \mathcal{L}_m$. K is said to be D&A multi-coobservable w.r.t $\mathcal{R}_\mathcal{D}$ if there exists an integer $m \geq 1$ such that K is D&A m-coobservable w.r.t. $\mathcal{R}_\mathcal{D}$.*

In the next theorem, we relate the D&A m-coobservability w.r.t. $\mathcal{R}_\mathcal{D}$ with the existence of D&A m-decision supervisor $Sup_\vee$ that can control the plant so that it respects K.

Theorem 2.5.1 *Consider $K \subseteq \mathcal{L}_m$ where $K \neq \emptyset$ and an integer $m \geq 1$. If K is controllable w.r.t. $\mathcal{L}$ and Σ_{uc}, D&A m-coobservable w.r.t $\mathcal{R}_\mathcal{D}$ and $\mathcal{L}_m$-closed, then there exists a partition $\{\mathcal{D}_\sigma^1, \ldots, \mathcal{D}_\sigma^{m_\sigma}\}$ of every $\mathcal{D}_\sigma(K)$ (where $\sigma \in \Sigma_c$ and $m_\sigma \leq m$) s.t. the D&A m-decision supervisor $Sup_\vee$ defined by Eqs. (2.9)-(2.11) is nonblocking and satisfying $\mathcal{L}_m(Sup_\vee/G) = K$ and $\mathcal{L}(Sup_\vee/G) = \overline{K}$.*

The next Proposition sets a necessary and sufficient condition for a language to be D&A multi-coobservable w.r.t. $\mathcal{R}_\mathcal{D}$.

Proposition 2.5.1 *$K \subseteq \mathcal{L}_m$ is D&A multi-coobservable w.r.t. $\mathcal{R}_\mathcal{D}$ iff, $\forall \sigma \in \Sigma_c$, $\forall A \in E_{\mathcal{R}_\mathcal{D}^\sigma}$, we have :*

$$\bigcap_{i \in I_\sigma} P_i^{-1} P_i(A) \cap \mathcal{E}_\sigma(K) = \emptyset. \tag{2.14}$$

In the next example, we show a situation when a specification is D&A multi-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$ and not C&P multi-coobservable w.r.t $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$.

Example 2.5.1 *Consider the specification of Fig. 2.4, which is complementary with the one of Fig. 2.2 (used in Section 2.4), in the sense that one is obtained from the other by just switching between $\mathcal{E}_\sigma(K)$ and $\mathcal{D}_\sigma(K)$. In Example 2.4.1, the specification of Fig. 2.2 was shown to be C&P 2-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$ ($\mathcal{N}_\mathcal{E} = \mathcal{N}_\mathcal{E}^g$). We can then easily deduce that the new specification is D&A 2-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$. More precisely, the equivalence classes of $\mathcal{M}_{\mathcal{N}_\mathcal{D}^g}$ are given by : $A_{d,1} = \{b_1a_2\}$, $A_{d,2} = \{a_1a_2d_1, b_1b_2d_1, d_2b_1a_2d_1, c_2a_1b_2d_1\}$ and $A_{d,3} = \{c_2a_1a_2\}$ which correspond to the sets of sequences reaching the states G, M and N (Fig 2.3), respectively. We have for all $A_{d,j}$ ($j \in \{1,2,3\}$), $\bigcap_{i=1,2} P_i^{-1}P_i(A_{d,j}) \cap \mathcal{E}_\sigma(K) = \emptyset$, then K is D&A multi-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$.*

Let us now show that the new specification is not *C&P 2-coobservable w.r t. $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$. The equivalence classes of $\mathcal{M}_{\mathcal{N}_\mathcal{E}^g}$ are given by : $A_{e,1} = \{a_1b_2\}$, $A_{e,2} = \{a_1a_2, b_1b_2, d_2b_1a_2, c_2a_1b_2\}$, $A_{e,3} = \{d_2b_1b_2\}$ and $A_{e,4} = \{a_1a_2c_1, b_1b_2c_1, d_2b_1a_2c_1, c_2a_1b_2c_1\}$ which correspond to the sets of sequences reaching the states F, I, K and L (Fig. 2.3), respectively. We have $b_1a_2 \in \bigcap_{i=1,2} P_i^{-1}P_i(A_{e,2}) \cap \mathcal{D}_\sigma(K) \neq \emptyset$, which means that K is not C&P multi-coobservable w.r.t $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$.*

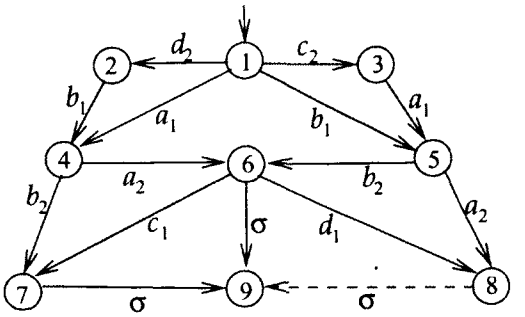

Figure 2.4 Plant and specification. The latter is obtained by forbidding the dashed transition

From the fact that the new specification is not C&P 2-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$, we can easily deduce that the specification of Fig. 2.2 is not D&A 2-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$. To recapitulate, we have found an example (Fig. 2.2) which is C&P 2-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$ and not D&A 2-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$, and an example (Fig. 2.4) which is D&A 2-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$ and not C&P 2-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$.

In the general case where we have more than one controllable event, we can have the situation where the specification is C&P multi-coobservable for some controllable events and D&A multi-coobservable for others. This fact has motivated to study the so-called C&P∨D&A multi-decision architecture defined in the following subsection.

2.5.2 Class of C&P∨D&A multi-Coobservable Languages

The C&P∨D&A multi-decision architecture is based on partitioning the set of controllable events in two subsets $\Sigma_{c,\wedge}$ and $\Sigma_{c,\vee}$ such that $\Sigma_c = \Sigma_{c,\wedge} \cup \Sigma_{c,\vee}$ and $\Sigma_{c,\wedge} \cap \Sigma_{c,\vee} = \emptyset$. We obtain a so-called C&P∨D&A multi-decision architecture by applying the rules of C&P (resp. D&A) multi-decision to the events of $\Sigma_{c,\wedge}$ (resp. $\Sigma_{c,\vee}$). Hence, we assume that for every $\sigma \in \Sigma_{c,\wedge}$, we are given a partition $(\mathcal{E}_\sigma^1, \mathcal{E}_\sigma^2, \ldots, \mathcal{E}_\sigma^{m_\sigma})$ of $\mathcal{E}_\sigma(K)$, and for every $\sigma \in \Sigma_{c,\vee}$, we are given a partition $(\mathcal{D}_\sigma^1, \mathcal{D}_\sigma^2, \ldots, \mathcal{D}_\sigma^{m_\sigma})$ of $\mathcal{D}_\sigma(K)$. Therefore, for every $\sigma \in \Sigma_{c,\wedge}$, the decisions are computed using Eqs. 2.3-2.5, and for every $\sigma \in \Sigma_{c,\vee}$, the decisions are computed using Eqs. 2.9-2.11. For clarity, we give again all these equations in the following. Equations 2.15-2.16 correspond to Equations 2.3 and 2.9 and compute each jth micro-decision of each Sup_i :

$\sigma \in \Sigma_{c,\wedge} : \forall j \in \{1, \ldots, m_\sigma\}, \forall i \in \{1, \ldots, n\},$

$$Sup_i^j(P_i(s), \sigma) = 1 \Leftrightarrow [P_i^{-1}P_i(s) \cap \mathcal{E}_\sigma^j \neq \emptyset] \vee [\sigma \in \Sigma_{c,\wedge} \backslash \Sigma_{c,i} \cup \Sigma_{uc}]. \tag{2.15}$$

$\sigma \in \Sigma_{c,\vee} : \forall j \in \{1, \ldots, m_\sigma\}, \forall i \in \{1, \ldots, n\}$,

$$Sup_i^j(P_i(s), \sigma) = 0 \Leftrightarrow [P_i^{-1}P_i(s) \cap \mathcal{D}_\sigma^j \neq \emptyset \wedge \sigma \in \Sigma_{c,i}] \vee [\sigma \in \Sigma_{c,\vee} \backslash \Sigma_{c,i}]. \tag{2.16}$$

Equation 2.17 puts together Equations 2.4 and 2.10 and combines the n j^{th} micro-decisions :

$$S^j(s, \sigma) = \begin{cases} \bigwedge_{1 \leq i \leq n} Sup_i^j(P_i(s), \sigma) & \text{if } \sigma \in \Sigma_{c,\wedge}\,; \\ \bigvee_{1 \leq i \leq n} Sup_i^j(P_i(s), \sigma) & \text{if } \sigma \in \Sigma_{c,\vee}. \end{cases} \tag{2.17}$$

Equation 2.18 puts together Equations 2.5 and 2.11 and computes the actual decision that is applied to the plant :

$$Sup_G(s, \sigma) = \begin{cases} \bigvee_{1 \leq j \leq m_\sigma} S^j(s, \sigma) & \text{if } \sigma \in \Sigma_{c,\wedge}\,; \\ \bigwedge_{1 \leq j \leq m_\sigma} S^j(s, \sigma) & \text{if } \sigma \in \Sigma_{c,\vee}. \end{cases} \tag{2.18}$$

Hence, the C&P∨D&A multi-decision architecture can be schematized by incorporating two modules, one performing the C&P multi-decision architecture for all $\sigma \in \Sigma_{c,\wedge}$ and the other performing the D&A multi-decision architecture for all $\sigma \in \Sigma_{c,\vee}$.

Note that for each $\sigma \in \Sigma_c$, we have used a value m_σ. Let us denote by m the biggest of these values m_σ considering all the events of Σ_c. The obtained supervisor will be denoted Sup_G and called C&P∨D&A m-decision supervisor.

For a given partition $\{\Sigma_{c,\wedge}, \Sigma_{c,\vee}\}$ of Σ_c, for all $i \in I_\sigma$, we define $\Sigma_{c,\wedge,i} = \Sigma_{c,i} \cap \Sigma_{c,\wedge}$ the set of locally controllable events to which we apply the rules of C&P multi-decision, and $\Sigma_{c,\vee,i} = \Sigma_{c,i} \cap \Sigma_{c,\vee}$ the set of locally controllable events to which we apply the rules of D&A multi-decision.

Since we are interested by decidable notions (as already explained for the D&A multi-decision), we directly define C&P∨D&A m-coobservability w.r.t. $\mathcal{R}_\mathcal{E}$ and $\mathcal{R}_\mathcal{D}$ (the term C&P∨D&A can be omitted) that will be a condition for the existence of C&P∨D&A m-decision supervisor, where $\mathcal{R}_\mathcal{E} = (\mathcal{R}_\mathcal{E}^\sigma)_{\sigma \in \Sigma_{c\wedge}}$ and $\mathcal{R}_\mathcal{D} = (\mathcal{R}_\mathcal{D}^\sigma)_{\sigma \in \Sigma_{c\vee}}$.

Definition 2.5.3 *Consider $K \subseteq \mathcal{L}_m$ and an integer $m \geq 1$. K is said to be m-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$, $\mathcal{R}_\mathcal{D}$, $\mathcal{L}$, $\Sigma_{o,1}$, $\Sigma_{c,1}$, $\cdots$, $\Sigma_{o,n}$, $\Sigma_{c,n}$, if, there exists a partition $\{\Sigma_{c,\wedge}, \Sigma_{c,\vee}\}$ of Σ_c such that :*

1. *K is C&P m-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$, $\mathcal{L}$, $\Sigma_{o,1}$, $\Sigma_{c,\wedge,1}$, $\cdots$, $\Sigma_{o,n}$, $\Sigma_{c,\wedge,n}$,*
2. *K is D&A m-coobservable w.r.t. $\mathcal{R}_\mathcal{D}$, $\mathcal{L}$, $\Sigma_{o,1}$, $\Sigma_{c,\vee,1}$, $\cdots$, $\Sigma_{o,n}$, $\Sigma_{c,\vee,n}$.*

For simplicity, the term "w.r.t. $\mathcal{L}$, $\Sigma_{o,1}$, $\Sigma_{c,1}$, $\cdots$, $\Sigma_{o,n}$, $\Sigma_{c,n}$" can be omitted.

A partition $\{\Sigma_{c\wedge}, \Sigma_{c,\vee}\}$ of Σ_c can be found as follows. $\Sigma_{c,\wedge}$ (resp., $\Sigma_{c,\vee}$) may contain only events $\sigma \in \Sigma_c$ for which there exists a partition of $\mathcal{E}_\sigma(K)$ (resp., $\mathcal{D}_\sigma(K)$) that satisfies Eqs 2 6-2.7 (resp., Eqs. 2.12-2.13). And thus, for every $\sigma \in \Sigma_c$ that respects both requirements (related to $\Sigma_{c,\wedge}$ and $\Sigma_{c,\vee}$), we have the choice to put it in either $\Sigma_{c,\wedge}$ and $\Sigma_{c,\vee}$. Let us now define the multi-coobservabily w.r.t. $\mathcal{R}_\mathcal{E}$ and $\mathcal{R}_\mathcal{D}$ a weaker notion of m-coobservability w.r.t. $\mathcal{R}_\mathcal{E}$ and $\mathcal{R}_\mathcal{D}$.

Definition 2.5.4 *Consider $K \subseteq \mathcal{L}_m$. K is said to be multi-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$ and $\mathcal{R}_\mathcal{D}$ if, there exists an integer $m \geq 1$ such that K is m-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$ and $\mathcal{R}_\mathcal{D}$*

Theorem 2.5.2 *Consider $K \subseteq \mathcal{L}_m$ where $K \neq \emptyset$, tuples of finite equivalence relations $\mathcal{R}_\mathcal{E} = (\mathcal{R}^\sigma_\mathcal{E})_{\sigma\in\Sigma_{c\wedge}}$ and $\mathcal{R}_\mathcal{D} = (\mathcal{R}^\sigma_\mathcal{D})_{\sigma\in\Sigma_{c\vee}}$, and an integer $m \geq 1$. If K is controllable w.r.t. $\mathcal{L}$ and Σ_{uc}, m-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$ and $\mathcal{R}_\mathcal{D}$, and $\mathcal{L}_m$-closed, then there exist a partition $\{\mathcal{E}^1_\sigma, \ldots, \mathcal{E}^{m_\sigma}_\sigma\}$ of every $\mathcal{E}_\sigma(K)$ ($\sigma \in \Sigma_{c,e}, m_\sigma \leq m$) and a partition $\{\mathcal{D}^1_\sigma, \ldots, \mathcal{D}^{m_\sigma}_\sigma\}$ of every $\mathcal{D}_\sigma(K)$ ($\sigma \in \Sigma_{c,d}, m_\sigma \leq m$) s.t. the C&P∨D&A m-decision supervisor Sup_G defined by Eqs. (2.15-2.18) is nonblocking and satisfying $\mathcal{L}_m(Sup_G/G) = K$ and $\mathcal{L}(Sup_G/G) = \overline{K}$.*

Corollary 2.5.1 *Consider $K \subseteq \mathcal{L}_m$ where $K \neq \emptyset$. If K is controllable w.r.t. $\mathcal{L}$ and Σ_{uc}, multi-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$ and $\mathcal{R}_\mathcal{D}$, and $\mathcal{L}_m$-closed, then there exist an integer $m \geq 1$, a partition $\{\mathcal{E}^1_\sigma, \ldots, \mathcal{E}^{m_\sigma}_\sigma\}$ of every $\mathcal{E}_\sigma(K)$ ($\sigma \in \Sigma_{c,e}, m_\sigma \leq m$) and a partition $\{\mathcal{D}^1_\sigma, \ldots, \mathcal{D}^{m_\sigma}_\sigma\}$ of every $\mathcal{D}_\sigma(K)$ ($\sigma \in \Sigma_{c,d}, m_\sigma \leq m$), such that the C&P∨D&A m-decision supervisor Sup_G defined by Eqs. (2.15)-(2.18) is nonblocking and satisfying $\mathcal{L}_m(Sup_G/G) = K$ and $\mathcal{L}(Sup_G/G) = \overline{K}$.*

The next corollary is a consequence of the definitions of C&P∨D&A m-coobservability and multi-coobservability w.r.t. $\mathcal{R}_\mathcal{E}$ and $\mathcal{R}_\mathcal{D}$.

Corollary 2.5.2 *Consider $K \subseteq \mathcal{L}_m$. If K is C&P multi-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$ or D&A multi-coobservable w.r.t. $\mathcal{R}_\mathcal{D}$ then it is C&P∨D&A multi-coobservable w.r.t. $\mathcal{R}_\mathcal{E}$ and $\mathcal{R}_\mathcal{D}$.*

Let us consider an example which proves that the converse of Corollary 2.5.2 is not true, and thus, the C&P∨D&A architecture is more general than the C&P and D&A architectures. More precisely, we will give a specification which is C&P∨D&A multi-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$ and $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$ but neither C&P multi-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$ nor D&A multi-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$.

Example 2.5.2 *We consider the specification K of Fig. 2.5 that combines two previous examples : 1) the specification of Fig. 2.2 (where σ is renamed τ) which is C&P 2-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$; and 2) the specification of Fig. 2.4 which is D&A 2-coobservable w.r t. $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$. We can deduce that the new specification is C&P∨D&A 2-coobservable w.r.t.*

$\mathcal{M}_{\mathcal{N}_\mathcal{E}}$ and $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$, with $\Sigma_{c,\wedge} = \{\tau\}$ and $\Sigma_{c,\vee} = \{\sigma\}$. It can be easily deduced that the new specification is neither C&P multi-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$ nor D&A multi-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$, from the fact that the specification of Fig. 2.2 (resp. Fig. 2.4) has been shown to be not D&A (resp., not C&P) multi-coobservable w.r.t. $\mathcal{M}_{\mathcal{N}_\mathcal{D}}$ (resp. $\mathcal{M}_{\mathcal{N}_\mathcal{E}}$) (see Example 2.5.1).

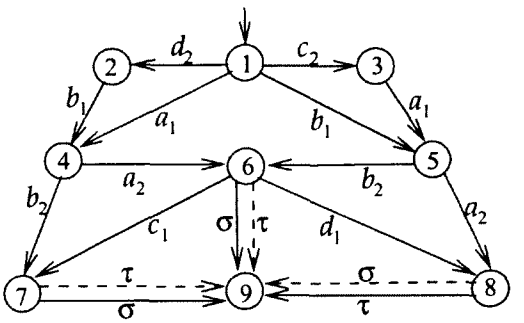

Figure 2.5 Plant and specification. The latter is obtained by forbidding the dashed transitions

2.6 Conclusion

We have studied the multi-decision control framework dealing with decentralized supervisory control. In this framework, each supervisor issues a tuple of micro-decisions instead of a single decision. We have opted to apply the multi-decision control in the case of C&P, D&A and C&P∨D&A architectures. For every architecture, we have defined and studied the notion of m-coobservability, which is useful to characterize the class of languages achievable by the multi-decision control. Depending on the considered architecture, the m-coobservability is based on partitioning $\mathcal{E}_\sigma(K)$ or $\mathcal{D}_\sigma(K)$ or both. Since the verification of the multi-coobservability is potentially undecidable (due to infinite number of partitions), we have defined stronger and decidable varieties of multi-coobservability for each architecture, by using specific equivalence relations.

As a future work, we will investigate more efficient methods for obtaining decidable versions of multi-coobservability. Another interesting issue is to apply the multi-decision control to the inference-based architecture. And last but not least, we will study the complexity aspect of all our developed methods.

CHAPITRE 3

Contrôle supervisé multi-décisionnel : architectures décentralisées fonctionnant en parallèle pour contrôler des SED

Article : H. Chakib and A. Khoumsi, *Multi-decision Supervisory Control : Parallel Decentralized Architectures Cooperating for Controlling Discrete Event Systems*, Accepted "as a full paper" in IEEE Transactions on Automatic Control, 2011.

Avant-propos

Auteurs et affiliation :

Hicham Chakib : étudiant au doctorat, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Ahmed Khoumsi : professeur, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Date d'acception : 18 février 2011

État de l'acceptation : Accepté pour être publié

Revue : IEEE Transactions on Automatic Control

Référence : L'éditeur nous a informé que la publication est prévue pour octobre 2011.

Titre français : Contrôle supervisé multi-décisionnel : architectures décentralisées fonctionnant en parallèle pour contrôler des SED.

Contribution au document : Cet article effectue une étude générale de l'approche multi-décisionnelle qui peut généraliser une classe générique d'architectures de contrôle dites éligibles. Nous étudions ensuite plus spécifiquement le contrôle multi-décisionnel pour généraliser une catégorie particulière d'architectures éligibles : les architectures par inférence. Notons que celles-ci englobent les architectures C&P, D&A et C&P∨D&A que nous avons généralisées dans notre précédente contribution (chapitre 2).

Résumé français : Cet article étudie le contrôle décentralisé de SED, où un ensemble de superviseurs locaux coopèrent dans le but de réaliser une spécification globale donnée en contrôlant un SED. On propose une nouvelle approche, appelé *contrôle multi-décisionnel*,

qui consiste à utiliser des superviseurs décentralisés fonctionnant en parallèle et dont les décisions sont fusionnées disjonctivement ou conjonctivement. Nous avons identifié des conditions suffisantes qui rendent une architecture décentralisée éligible pour être utilisée dans une architecture multi-décisionnelle. Nous avons effectué une étude générique de l'approche multi-décisionnelle en considérant plusieurs architectures éligibles fonctionnant en parallèle, et nous avons aussi considéré en detail le cas particulier où plusieurs Inf_{N_j}-superviseurs à inférence par ambiguïté fonctionnent en parallèle.

Note : À la suite des corrections demandées par les membres du jury, le contenu de cet article diffère de celui qui a été accepté.

Abstract : This article deals with decentralized supervisory control, where a set of local supervisors cooperate in order to achieve a given global specification by controlling a discrete event system. We propose a new framework, called *multi-decision control*, whose basic principle consists in using several decentralized supervisory control architectures working in parallel and whose decisions are combined disjunctively or conjunctively. We have identified sufficient conditions that make a decentralized architecture eligible to be used in the multi-decision framework. We have studied the generic framework consisting of several eligible architectures running in parallel, and we have also considered in detail the particular case of several inference-based Inf_{N_j}-supervisors running in parallel.

3.1 Introduction

This paper deals with decentralized control of DES, where a set of local supervisors cooperate according to their observations in order to restrict the behaviour of a *plant* so that it respects a given global *specification*. Each local supervisor has a local observation of the plant and takes local decisions without communicating with the other local supervisors. And the local decisions taken by the various supervisors are merged in order to issue an effective decision.

We propose a multi-decision framework whose basic principle consists in using several existing decentralized architectures working in parallel. For example, we can have a *conjunctive* and a *conditionally disjunctive* architectures [Rudie et Wonham, 1992; Yoo et Lafortune, 2002a, 2004] running in parallel. The global decisions of all these decentralized architectures are fused disjunctively or conjunctively in order to obtain an effective decision that is applied to the plant. The motivation of multi-decision framework is to obtain an architecture that generalizes all the decentralized architectures that compose it.

The paper is organized as follows. Discussion about related work is given in Section 3.2. Section 3.3 introduces the decentralized control and presents pertinent definitions and results. Section 3.4 presents the motivation and principle of *multi-decision control*, where several decentralized control architectures are running in parallel and their global decisions are combined disjunctively or conjunctively. In Subsection 3.4.1, we identify sufficient conditions that make a decentralized architecture eligible to be used in the multi-decision framework. The case of *disjunctive* combination is considered in detail in Subsection 3.4.2. Then, we explain in Subsection 3.4.3 how the results of Subsection 3.4.2 can be adapted to the case of *conjunctive* combination. In Section 3.5, we illustrate the multi-decision by studying the particular case of several inference-based architectures running in parallel and combined disjunctively. Section 3.6 presents several interesting properties of the multi-decision framework. In multi-decision, we are confronted with the problem of decomposing infinite languages. In Section 3.7, we propose an approach of how to tackle that decomposition problem. Section 3.8 discusses the verification of coobservability for the inference-based architecture of Section 3.5, using the decomposition approach of Section 3.7. The conclusion is presented in Section 3.9. The proofs are presented in an appendix.

3.2 Related work

There are many prior articles studying decentralized control of DES [Cieslak *et al.*, 1988; Jiang et Kumar, 2000; Kumar et Takai, 2007; Lin et Wonham, 1988a; Overkamp et van Schuppen, 2001; Prosser *et al.*, 1997; Ricker et Rudie, 2000, 2003; Rudie et Willems, 1995; Rudie et Wonham, 1992; Takai et Ushio, 2005; Tripakis, 2004; Yoo et Lafortune, 2002a, 2004]. It has been shown in [Tripakis, 2004] that the decentralized control problem is in general undecidable. The first architecture that has been proposed in decentralized control is referred to as the C&P architecture [Rudie et Wonham, 1992; Yoo et Lafortune, 2002a]. The paper [Prosser *et al.*, 1997] presents the first alternative in the study of decentralized control with different fusion rules.

In all existing decentralized control architectures, each local supervisor continuously observes the plant and takes local decisions. The local decisions taken by the local supervisors are fused in order to generate the actual decision that is applied to the plant. The authors of [Yoo et Lafortune, 2002a] have proposed the D&A architecture which is complementary with the C&P one. The authors in [Yoo et Lafortune, 2002a] also propose a *general* architecture that combines and generalizes the C&P and D&A ones. The authors of [Yoo et Lafortune, 2004] use a *conditional* architecture, which generalizes the general architecture of [Yoo et Lafortune, 2002a].

The authors of [Takai et Ushio, 2005] propose a new decentralized supervisory control architecture using dynamic default control instead of static one, where the default decisions are updated dynamically.

In [Ricker et Rudie, 2000, 2003], an approach is proposed where each local supervisor can take decisions based on its ambiguities together with the ambiguities of the other local supervisors. The authors of [Kumar et Takai, 2005, 2007] propose a general *inference*-based framework for managing ambiguities. Note that the disjunctive and conjunctive architectures of [Yoo et Lafortune, 2002a] are specific cases of the inference-based framework, when the ambiguity is restricted to be 0. And the conditional architecture of [Yoo et Lafortune, 2004] is a specific case of the inference-based framework, when the ambiguity is restricted to be ≤ 1.

In [Chakib et Khoumsi, 2008a,b], the multi-decision control has been defined in the special case where several disjunctive or several conjunctive architectures are running in parallel, without any mention and study of its generic form. Actually, it was defined in an operational way which does not show that several architectures are working in parallel. For this reason, [Chakib et Khoumsi, 2008a,b] does not permit to understand the fundamental aspects of multi-decision. Note that the multi-decision approach has also been studied in the decentralized diagnosis of DES [Chakib et Khoumsi, 2009; Khoumsi et Chakib, 2008a] and prognosis of DES [Khoumsi et Chakib, 2009].

3.3 Decentralized supervisory control of DES

3.3.1 Supervisory control of DES

We consider a plant described over an alphabet Σ by a prefix-closed language $\mathcal{L} \subseteq \Sigma^*$ and a marked language $\mathcal{L}_m \subseteq \mathcal{L}$, and a specification described by a language $K \subseteq \mathcal{L}_m$. We consider that the plant is also modeled by a finite state automaton (FSA) G accepting $\mathcal{L}_m$.

We say that a trace $t \in \Sigma^*$ is a prefix of a trace $s \in \Sigma^*$, if there exists a trace $\lambda \in \Sigma^*$ such that $s = t\lambda$. We denote by $\overline{K}$ the set of all prefixes of traces of a language $K \subseteq \Sigma^*$, i.e., $\overline{K} = \{s \in \Sigma^* \mid \exists t \in K,\ s \text{ is a prefix of } t\}$. K is said prefix-closed if $K = \overline{K}$.

The alphabet Σ is partitioned into Σ_c and Σ_{uc}, the sets of *controllable* and *uncontrollable* events, respectively. For every controllable event $\sigma \in \Sigma_c$, we denote by $\mathcal{E}_\sigma(K) = \{s \in \overline{K} \mid s\sigma \in \overline{K}\}$ the set of traces of $\overline{K}$ after which σ is accepted by $\overline{K}$. Similarly, we denote by

$\mathcal{D}_\sigma(K) = \{s \in \overline{K} | \, s\sigma \in \mathcal{L}\backslash\overline{K}\}$ the set of traces of $\overline{K}$ after which σ is accepted by $\mathcal{L}$ and forbidden by $\overline{K}$. When a single specification K is used, K can be omitted in $\mathcal{E}_\sigma(K)$ and $\mathcal{D}_\sigma(K)$.

A supervisory system (supervisor for short) *Sup* restricts the behavior of the plant so that it conforms to K, by taking effective enabling/disabling decisions. By *effective decision*, we mean the decision that is actually applied to the plant. Let then $Sup(s,\sigma) \in \{\phi, 0, 1\}$ denote the effective enabling/disabling decision taken on an event $\sigma \in \Sigma$ after the execution of a trace $s \in \mathcal{L}$. $Sup(s,\sigma) = 1$ (resp. 0) means that σ is enabled (resp. disabled) after the execution of s. A decision $Sup(s,\sigma) = \phi$ means a "don't care" or "unsure" decision. A fundamental property that is respected by the effective decision $Sup(s,\sigma)$ of any supervisor is :

$$\forall s \in \mathcal{L}, \forall \sigma \in \Sigma_{uc} : Sup(s,\sigma) = 1.$$

Definition 3.3.1 *Given a language $K \subseteq \mathcal{L}_m$, a supervisor Sup is said* admissible *w.r.t. K if for every $\sigma \in \Sigma_c$, $\forall s \in \mathcal{E}_\sigma \cup \mathcal{D}_\sigma$, $Sup(s,\sigma) \neq \phi$.*

In the sequel, the terms "w.r.t. (E, D)" and "w.r.t. K" can be omitted if it is clear from the context. The prefix-closed language $\mathcal{L}(Sup/G)$ generated by the plant under the control of an admissible supervisor *Sup* is defined as follows :

- $\epsilon \in \mathcal{L}(Sup/G)$,
  $[s \in \mathcal{L}(Sup/G) \wedge s\sigma \in \mathcal{L} \wedge Sup(s,\sigma) = 1] \Leftrightarrow [s\sigma \in \mathcal{L}(Sup/G)]$.

The corresponding marked language is defined by $\mathcal{L}_m(Sup/G) = \mathcal{L}(Sup/G) \cap \mathcal{L}_m$. A supervisor *Sup* is called nonblocking if $\overline{\mathcal{L}_m(Sup/G)} = \mathcal{L}(Sup/G)$.

The ultimate objective of supervisory control is to satisfy the following conditions :

$$\mathcal{L}(Sup/G) = \overline{K}, \tag{3.1}$$

$$\mathcal{L}_m(Sup/G) = K. \tag{3.2}$$

To satisfy Conds. (3.1) and (3.2), *Sup* acts upon the plant by enabling every event which is authorized by the specification, and disabling every event which is authorized by the plant and forbidden by the specification. Formally, $\forall \sigma \in \Sigma_c$

$$s \in \mathcal{E}_\sigma \Rightarrow Sup(s,\sigma) = 1, \tag{3.3}$$

$$s \in \mathcal{D}_\sigma \Rightarrow Sup(s,\sigma) = 0. \tag{3.4}$$

Definition 3.3.2 *Given a language $K \subseteq \mathcal{L}_m$, a supervisor Sup is said :*

consistent *w.r.t.* (E, D), *where* $E \subseteq \mathcal{E}_\sigma$ *and* $D \subseteq \mathcal{D}_\sigma$ *for some* $\sigma \in \Sigma_c$, *if Sup satisfies Conds. (3.3) and (3.4) w.r.t.* (E, D), *i.e.,* $\forall s \in E$, $Sup(s, \sigma) = 1$, *and* $\forall s \in D$, $Sup(s, \sigma) = 0$,
consistent *w.r.t.* K *if* , $\forall \sigma \in \Sigma_c$, *Sup is consistent w.r.t.* $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$.

Recall the usual notions of $\mathcal{L}_m$-closure and controllability.

Definition 3.3.3 *A language* $K \subseteq \mathcal{L}_m$ *is said* $\mathcal{L}_m$*-closed if* $K = \overline{K} \cap \mathcal{L}_m$.

Definition 3.3.4 *A language* $K \subseteq \mathcal{L}_m$ *is said controllable w.r.t.* $\mathcal{L}$ *and* Σ_{uc} *if* $\overline{K}\Sigma_{uc} \cap \mathcal{L} \subseteq \overline{K}$.

In the sequel, the term "w.r.t. $\mathcal{L}$ and Σ_{uc}" will be omitted when referring to controllability.

3.3.2 Decentralized supervisory control principle

Decentralized control [Kumar et Takai, 2007; Rudie et Wonham, 1992; Yoo et Lafortune, 2002a, 2004] is performed by n local supervisors $(Sup_i)_{1 \leq i \leq n}$. Each Sup_i has its own set of observable events $\Sigma_{o,i}$ and own set of controllable events $\Sigma_{c,i}$. We define $\Sigma_o = \Sigma_{o,1} \cup \cdots \cup \Sigma_{o,n}$, $\Sigma_c = \Sigma_{c,1} \cup \cdots \cup \Sigma_{c,n}$, $\Sigma_{uo} = \Sigma \setminus \Sigma_o$ and $\Sigma_{uc} = \Sigma \setminus \Sigma_c$. We denote by $I = \{1, \ldots, n\}$ the indexing set of all supervisors. For any controllable event $\sigma \in \Sigma_c$, we define by $I_\sigma = \{i \in I \mid \sigma \in \Sigma_{c,i}\}$ the indexing set of local supervisors controlling σ, and $n_\sigma = |I_\sigma|$ denotes the number of local supervisors controlling σ. We denote by $P_i : \Sigma^* \to \Sigma_{o,i}^*$ the natural projection that hides the events of $\Sigma \setminus \Sigma_{o,i}$ from any trace $s \in \Sigma^*$. The inverse projection is defined as $P_i^{-1}(s) = \{t \in \Sigma^* : P_i(t) = s\}$.

In decentralized control, each local supervisor $Sup_i, i \in I$, is a function $Sup_i : \Sigma_{o,i}^* \times \Sigma_c \to LD$, that associates a local decision $Sup_i(P_i(s), \sigma) \in LD$ to every controllable event σ and every observed trace $P_i(s)$. LD is a finite arbitrary set of local decisions. A particular decision, denoted ϕ, accounts for silent decision, meaning that the local supervisor is unsure or doesn't care. The effective decision that is applied to the plant is obtained by combining the local decisions of the n local supervisors $(Sup_i)_{i \in I}$ using a so-called *coordinating operator* $\mathrm{D} : LD^n \to \{\phi, 0, 1\}$, we say that D is defined over LD. The system enclosing the tuple of local supervisors $(Sup_i)_{i \in I}$ and the coordinating module D is called a *decentralized supervisor*, and denoted by $((Sup_i)_{i \in I}, \mathrm{D})$. The effective decision $Sup(s, \sigma) \in \{\phi, 0, 1\}$ of a decentralized supervisor $Sup = ((Sup_i)_{i \in I}, \mathrm{D})$ is computed as follows :

$$Sup(s, \sigma) = \mathrm{D}((Sup_i(P_i(s), \sigma))_{i \in I_\sigma}) \tag{3.5}$$

(3.5) means that $Sup(s, \sigma)$ is computed by applying the operator D to all $Sup_i(P_i(s), \sigma)$.

Definition 3.3.5 *A* D*-supervisor is a decentralized supervisor defined by (3.5) for a given* D *and any local supervisors* $(Sup_i)_{i\in I}$*. The set of all* D*-supervisors is called a* D*-architecture.*

For example, we have the conjunctive architecture ($\wedge$-architecture) and the disjunctive architecture ($\vee$-architecture) of [Rudie et Wonham, 1992; Yoo et Lafortune, 2002a], the conditional conjunctive architecture (COND-$\wedge$-architecture) and conditional disjunctive architecture (COND-$\vee$-architecture) of [Yoo et Lafortune, 2004], and the N-inference architecture (Inf_N-architecture) of [Kumar et Takai, 2007].

3.4 Architectures running in Parallel

Increasing the set of languages that are achievable under control has often been an important criterion when it comes to the development of control architectures, and more specifically decentralized architectures. With this idea, the objective of multi-decision is to be applied to a decentralized architecture A in order to obtain a new architecture B that generalizes A, in the sense that the set of languages achievable with B contains the set of languages achievable with A. The basic idea is to increase the information transmitted from the local supervisors $(Sup_i)_{i\in I}$ to the fusion operator D. An extreme solution is that each Sup_i takes, and transmits to D, a decision on enabling/disabling an event $\sigma \in \Sigma_c$ only when it is *certain* of its decision. When Sup_i is unsure of the decision to take, it informs D of what it has observed. When all $(Sup_i)_{i\in I}$ are unsure on the decision to take on an event $\sigma \in \Sigma_c$, then D combines their observations in order to obtain a richer information and take a decision on σ. This solution has been developed in [Khoumsi et Chakib, 2007, 2008b]. Its limitation is that it turns out *undecidable* when it is question of checking a notion of n-observability and a more restrictive notion of n-normality [Khoumsi et Chakib, 2008b]. The decidability is obtained when the plant or specification language is *finite*, but in general this is an unrealistic restriction.

The objective of multi-decision is to provide a less extreme but decidable solution. The basic principle is that in each site i, we will use several, say p, local supervisors $(Sup_i^j)_{j\in J}$, where each Sup_i^j generates a local decision. Therefore, in each site i, we have p different local decisions. That is why we use the term "multi-decision". This is equivalent to say that we have p decentralized supervisors $(Sup^j)_{j\in\{1,..,p\}}$ running in parallel and whose global decisions are fused into an *effective decision*. Hereafter, the number of decentralized supervisors is denoted by p, and we denote by $J = \{1, \ldots, p\}$ the indexing set of the decentralized supervisors running in parallel. Therefore, each decentralized supervisor Sup^j

achieves its control according to a given decentralized architecture as shown in Subsection 3.3.2 and (3.5), but with a superscript j. That is, for every $j \in J$, $Sup^j = ((Sup_i^j)_{i\in I}, \mathrm{D}^j)$ contains n local supervisors $(Sup_i^j)_{i\in I}$ and a coordinating module D^j. The global decision issued by Sup^j, following the execution of a trace $s \in \mathcal{L}$, is computed by the following equation, which corresponds to (3.5) with a superscript j added to Sup, Sup_i and D :

$$Sup^j(s,\sigma) = \mathrm{D}^j((Sup_i^j(P_i(s),\sigma))_{\{i\in I_\sigma\}}) \tag{3.6}$$

The global decisions of the decentralized supervisors $(Sup^j)_{j\in J}$ are combined using a coordinating module $\mathbf{D}$ in order to obtain an effective decision that satisfy Conds. (3.1) and (3.2). $\mathbf{D}$ is defined as a function $\mathbf{D} : \{\phi, 0, 1\}^p \rightarrow \{\phi, 0, 1\}$. The system enclosing the tuple of local supervisors $(Sup_i^j)_{i\in I, j\in J}$, the fusion operators $(\mathrm{D}^j)_{j\in J}$ and the global fusion operator $\mathbf{D}$ is called a *multi-decision supervisor*, and denoted by $((Sup^j)_{j\in J}, \mathbf{D}) = (((Sup_i^j)_{i\in I}, \mathrm{D}^j)_{j\in J}, \mathbf{D})$. The effective decision of $Sup = ((Sup^j)_{j\in J}, \mathbf{D})$, $Sup(s,\sigma) \in \{\phi, 0, 1\}$, is computed as follows :

$$Sup(s,\sigma) = \mathbf{D}((Sup^j(s,\sigma))_{\{j\in J\}}) \tag{3.7}$$

Eq. (3.7) means that the effective decision $Sup(s,\sigma)$ is computed by applying the operator $\mathbf{D}$ to all the global decisions $Sup^j(s,\sigma)$.

Definition 3.4.1 *A* $\boldsymbol{D}$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$*-supervisor is a multi-decision supervisor defined by (3.6) and (3.7) for given* $\boldsymbol{D}$ *and* $(Sup^j)_{j\in J}$*. The set of all* $\boldsymbol{D}$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$*-supervisors is called a* $\boldsymbol{D}$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$*-architecture.*

When all the D^j-supervisors Sup^j (for $j \in J$) correspond to the same decentralized architecture ψ, we obtain an architecture which is denoted $\mathbf{D}$-ψ^p, while Sup is called $\mathbf{D}$-ψ^p-supervisor.

Recall that the global decisions $Sup^j(s,\sigma)$ ($= 0$, 1 or ϕ), $j \in J$, are just intermediate results used to compute $Sup(s,\sigma)$, the latter being the effective decision actually applied to the plant.

Let us for example take $p = 2$, $\mathrm{D}^1 = \vee$, $\mathrm{D}^2 =$ COND-$\wedge$, and $\mathbf{D} = \wedge$. This means that we have a disjunctive (D^1) and a conditionally-conjunctive (D^2) architectures running in parallel and whose global decisions are fused conjunctively ($\mathbf{D}$) for obtaining the effective decision. And thus we have the $\wedge$-($\vee$,COND-$\wedge$)-architecture.

The multi-decision control consists therefore in using p D^j-supervisors Sup^j (for $j \in J$) running in parallel and whose global decisions are fused by an operator $\mathbf{D}$ in order to

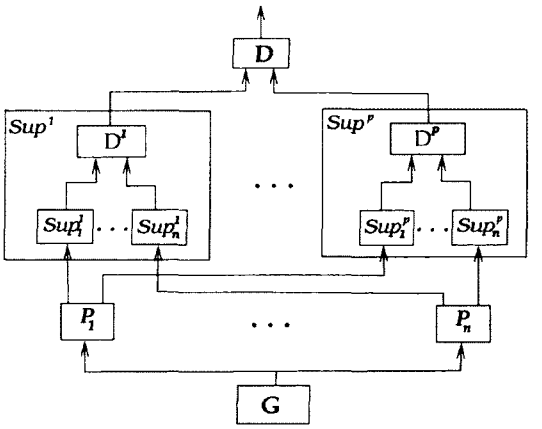

Figure 3.1 Multi-decision control framework

obtain an effective decision (see Fig. 3.1). We state in the following lemma the relation of the consistency of a nonblocking and admissible supervisor *Sup* with a controllable and $\mathcal{L}_m$-closed language K and the achievement of K by *Sup*.

Lemma 3.4.1 *Consider $K \subseteq \mathcal{L}_m$ and a supervisor Sup. The following assertions are equivalent :*

1. *Sup is nonblocking and admissible and such that $\mathcal{L}_m(Sup/G) = K$ and $\mathcal{L}(Sup/G) = \overline{K}$,*
2. *K is controllable and $\mathcal{L}_m$-closed, and Sup is consistent w.r.t. K.*

Consider $\mathcal{L}$ and K as in Section 3.3.1, and $(\Sigma_{o,i})_{i=1,\ldots,n}$ as in Section 3.3.2. For the purpose of our study, we use the following definition of *coobservability*.

Definition 3.4.2 *We call* coobservability *any property defined on pairs (E, D), for any $\sigma \in \Sigma_c$, $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$. Given a coobservability denoted* COOBS, *we say that (E, D) is* COOBS *w.r.t. $\mathcal{L}$, $\Sigma_{o,1}, \ldots, \Sigma_{o,n}$ to mean that* COOBS *is satisfied by (E, D).*

In the sequel, the term "w.r.t. $\mathcal{L}$, $\Sigma_{o,1}, \ldots, \Sigma_{o,n}$" will be omitted, except for some special cases. Note that Def. 3.4.2 also holds for a centralized architecture, by taking Σ_o instead of $(\Sigma_{o,i})_{i=1,\ldots,n}$.

3.4.1 Eligible architectures

In this section, we define the notion of *eligibility* that specifies sufficient conditions that an architecture should respect for being acceptable to be used as one of the architectures in parallel that compose any multi-decision system. Shortly, if a decentralized architecture is

eligible, then it can be used in the multi-decision framework. Eligibility is formally defined as follows.

Definition 3.4.3 *Given a fusion operator* D, *the corresponding* D*-architecture is said* eligible *(implicitly, for multi-decision), if there exists a coobservability property associated with* D, *denoted* D-coobservability *(or* D-COOBS*), such that the following* eligibility condition *ELC is satisfied :*

ELC-a · *For every* D*-supervisor Sup,* $\forall\sigma \in \Sigma_c$, $\forall E \subseteq \mathcal{E}_\sigma$, $\forall D \subseteq \mathcal{D}_\sigma$, *we have : if Sup is consistent w.r.t.* (E, D) *then* (E, D) *is* D-COOBS.

ELC-b *: There exists a* D*-supervisor SUP such that,* $\forall\sigma \in \Sigma_c$, $\forall E \subseteq \mathcal{E}_\sigma$, $\forall D \subseteq \mathcal{D}_\sigma$, *we have : if* (E, D) *is* D-COOBS *then SUP is consistent w.r.t.* (E, D).

Note that by ELC, we mean the two parts ELC-a and ELC-b. Based on ELC, we define the notion of D-coobservability of a language under eligible architectures.

Definition 3.4.4 *Given a fusion operator* D *and the corresponding eligible* D*-architecture, a language* $K \subseteq \mathcal{L}_m$ *is said* D*-coobservable (or* D-COOBS*) if,* $\forall\sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ *is* D-COOBS.

At the present state of our study, we consider two cases of **D** when $p > 1$: **D** is disjunctive ($\vee$) or conjunctive ($\wedge$), which will be considered in Sections 3.4.2 and 3.4.3, respectively. Actually, the terms "disjunctive" and "conjunctive" are used with an abuse of the language, in the following sense : By disjunctive, we mean that $Sup(s, \sigma) = 0$ if all $Sup^j(s, \sigma) = 0$, $Sup(s, \sigma) = 1$ if at least one $Sup^j(s, \sigma) = 1$, and $Sup(s, \sigma) = \phi$ otherwise. By conjunctive, we mean that $Sup(s, \sigma) = 1$ if all $Sup^j(s, \sigma) = 1$, $Sup(s, \sigma) = 0$ if at least one $Sup^j(s, \sigma) = 0$, and $Sup(s, \sigma) = \phi$ otherwise.

Remark 3.4.1 *As we will see, the two cases* $\boldsymbol{D} = \wedge$ *and* $\boldsymbol{D} = \vee$ *are based on decomposing the sets* $\mathcal{D}_\sigma$ *and* $\mathcal{E}_\sigma$, *respectively. So far, we have identified and thoroughly studied these two fusion operators* $\boldsymbol{D} = \wedge$ *and* $\boldsymbol{D} = \vee$, *but we see no reason why other values of* $\boldsymbol{D}$ *could not be used. The main question is how to realize a multi-decision architecture for other operators* $\boldsymbol{D}$. *This is an open problem. At the end of the conclusion, we propose to investigate this point in a near future.*

3.4.2 D is disjunctive

When **D** is disjunctive, we are in the presence of a so-called *disjunctive* multi-decision architecture, denoted $\vee$-($D^1, \ldots, D^p$), and any possible supervisor under this architecture is

called disjunctive multi-decision supervisor denoted $\vee$-($D^1, \ldots, D^p$)-supervisor. The latter consists of a set of p D^j-supervisors Sup^j (for $j \in J$) running in parallel and whose global decisions are fused *disjunctively* The effective decision $Sup(s, \sigma)$ is obtained as follows : $\forall \sigma \in \Sigma, \forall s \in \mathcal{L}$,

$$Sup(s,\sigma) = \begin{cases} 0 & \text{If, } \forall j \in J, Sup^j(s,\sigma) = 0, \\ 1 & \text{If, } \exists j \in J, Sup^j(s,\sigma) = 1, \\ \phi & \text{otherwise.} \end{cases} \tag{3.8}$$

The principle of disjunctive multi-decision architecture is based on considering, for every $\sigma \in \Sigma_c$, a decomposition $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p)$ of $\mathcal{E}_\sigma$ such that $\mathcal{E}_\sigma = \mathcal{E}_\sigma^1 \cup \ldots \cup \mathcal{E}_\sigma^p$. Note that we use the word *decomposition* instead of *partition* because the various $\mathcal{E}_\sigma^j$ are not necessarily disjoint with each other. For the sake of simplicity and without loss of generality, we consider that all $\mathcal{E}_\sigma$ are decomposed into the same number of sublanguages, that is, we use the same number p for all $\sigma \in \Sigma_c$, instead of a distinct p_σ for each $\sigma \in \Sigma_c$. This may necessitate that some $\mathcal{E}_\sigma^j$ are empty. In the latter case ($\mathcal{E}_\sigma^j = \emptyset$), we can take any supervisor Sup^j such that $Sup^j(s, \sigma) = 0$ for every $s \in \mathcal{L}$.

Definition 3.4.5 *For a given $p \geq 1$, let $\vee$-ELA^p-architecture mean a (disjunctive) multi-decision architecture consisting of p eligible architectures (hence the acronym ELA) fused disjunctively. And by $\vee$-ELA$^{\geq 1}$-architecture, we mean any (disjunctive) multi-decision architecture consisting of one or more eligible architectures fused disjunctively.*

Considering $\vee$-ELA^p-architecture, for which are defined $(D^j\text{-COOBS})_{j \in J}$, we define coobservability related to the $\vee$-ELA^p-architecture as follows :

Definition 3.4.6 *Consider a $\vee$-ELA^p-architecture consisting of p eligible architectures $(D^j)_{j \in J}$ and $K \subseteq \mathcal{L}_m$. For every $\sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is said $\vee$-($D^1, \ldots, D^p$)-COOBS if there exists a decomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ such that, $\forall j \in J$ $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is D^j-COOBS. K is said $\vee$-($D^1, \ldots, D^p$)-COOBS if, $\forall \sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-($D^1, \ldots, D^p$)-COOBS.*

In the case where the architectures and/or their number are unspecified, we obtain the following weaker notions :

Definition 3.4.7 *Given $\sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is said $\vee$-ELA^p-COOBS if it is $\vee$-($D^1, \ldots, D^p$)-COOBS for some eligible architectures $(D^j)_{1 \leq j \leq p}$. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is said $\vee$-ELA$^{\geq 1}$-COOBS if it is $\vee$-ELA^p-COOBS for some $p \geq 1$. K is said $\vee$-ELA^p-COOBS (resp., $\vee$-ELA$^{\geq 1}$-COOBS) if, $\forall \sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-ELA^p-COOBS (resp., $\vee$-ELA$^{\geq 1}$-COOBS).*

We have the following theorem that relates the decomposition of $\mathcal{E}_\sigma$ and the consistency of a $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-supervisor w.r.t. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$.

Theorem 3.4.1 *Consider $K \subseteq \mathcal{L}_m$, D^j-supervisors $(Sup^j)_{j\in J}$, and $\sigma \in \Sigma_c$. The $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-supervisor $Sup = ((Sup^j)_{j\in J}, \vee)$ is consistent w.r.t. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ if and only if there exists a decomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ such that, $\forall j \in J$, the D^j-supervisor Sup^j is consistent w.r.t. $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$.*

By using Lemma 3.4.1 and Theorem 3.4.1, we obtain the following theorem that states necessary and sufficient conditions for a language to be achievable under a disjunctive multi-decision architecture.

Theorem 3.4.2 *Consider a $\vee$-ELA^p-architecture consisting of p eligible architectures $(\mathrm{D}^j)_{j\in J}$, and a language $K \subseteq \mathcal{L}_m$. There exists a nonblocking and admissible $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-supervisor $Sup = ((Sup^j)_{j\in J}, \vee)$ such that $\mathcal{L}(Sup/G) = \overline{K}$ and $\mathcal{L}_m(Sup/G) = K$ if and only if K is $\mathcal{L}_m$-closed, controllable and $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-*Coobs.

By considering the case where we have a single eligible D-architecture (instead of several $(\mathrm{D}^j)_{j\in J}$ architectures in parallel), Theorem 3.4.2 becomes the following Corollary 3.4.1, which states that D-Coobs (in addition to the controllability and the $\mathcal{L}_m$-closure) characterizes the class of languages that are achievable under the D-architecture. The proof of Corollary 3.4.1 is omitted since it is a particular case of Theorem 3.4.2.

Corollary 3.4.1 *Consider an eligible architecture* D *and a language $K \subseteq \mathcal{L}_m$. There exists a nonblocking and admissible* D*-supervisor Sup such that $\mathcal{L}_m(Sup/G) = K$ and $\mathcal{L}(Sup/G) = \overline{K}$ if and only if K is controllable, $\mathcal{L}_m$-closed and* D-Coobs.

3.4.3 D is conjunctive

When **D** is conjunctive, we are in the presence of a so-called *conjunctive* multi-decision architecture, denoted $\wedge$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$; and any possible supervisor under this architecture is called conjunctive multi-decision supervisor, which is denoted $\wedge$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-supervisor. The conjunctive multi-decision architecture can be developed by adapting the disjunctive multi-decision architecture of Subsection 3.4.2 by the following essential modifications, for each $\sigma \in \Sigma_c$: 1) we decompose $\mathcal{D}_\sigma$ instead of $\mathcal{E}_\sigma$; and 2) we compute the effective decisions by using (3.9) instead of (3.8), that is, we replace the $\vee$ operator by the $\wedge$ operator. $\forall \sigma \in \Sigma$,

$\forall s \in \mathcal{L}$,

$$Sup(s,\sigma) = \begin{cases} 1 & \text{If, } \forall j \in J, Sup^j(s,\sigma) = 1, \\ 0 & \text{If, } \exists j \in J, Sup^j(s,\sigma) = 0, \\ \phi & \text{otherwise.} \end{cases} \tag{3.9}$$

Remark 3.4.2 *By using the above points 1) and 2), all the notions and results that can be obtained for $\boldsymbol{D} = \vee$ (i.e., for the disjunctive multi-decision architecture) can be adapted quite easily for $\boldsymbol{D} = \wedge$ (i.e., for the conjunctive multi-decision architecture). That is why we have not developed the latter case.*

Due to remark 3.4.2, in the following sections 3.5-3.8, we will consider uniquely the disjunctive multi-decision architecture (for simplicity, the term "disjunctive" will be omitted). More precisely, we will study our multi-decision framework in the case where $\mathbf{D} = \vee$. In order to clarify our idea, we will apply in Section 3.5 the multi-decision framework in the case of the inference-based architecture studied in [Kumar et Takai, 2007]. The reason why we have decided to apply the multi-decision to the inference-based architecture, is because, to the best of our knowledge, the inference-based architecture is the most general decentralized architecture before the development of our multi-decision framework. For example, it has been proved in [Kumar et Takai, 2007] that the disjunctive, conjunctive, conditionally disjunctive, and conditionally conjunctive architectures [Rudie et Wonham, 1992; Yoo et Lafortune, 2002a, 2004] are special cases of the inference-based architecture. So, the main objective of Section 3.5 is to prove the effectiveness of our multi-decision framework by showing that it permits to generalize the inference-based architecture. Furthermore, the inference-based ambiguity principle is more suitable when heterogenous decentralized supervisors are considered, it suffices to consider decentralized supervisors Sup^j with ambiguity levels N_j, $j \in J$, that are not necessarily the same.

3.5 Inference-based multi-decision architecture

In this section, we study the inference-based multi-decision architecture. An inference-based multi-decision supervisor $Sup = ((Sup^j)_{j\in J}, \vee)$ consists on p inference-based decentralized supervisors $(Sup^j)_{j\in J}$ running in parallel and whose global decisions are fused disjunctively ($\mathbf{D} = \vee$) for obtaining an effective decision that is actually applied to the plant. Each inference-based decentralized supervisor Sup^j consists of n *inference-based* local supervisors $(Sup_i^j)_{i\in I}$, and its decisions are computed using the inference-based architecture introduced in [Kumar et Takai, 2007]. First of all, we will present in the following subsection the inference-based architecture, and we prove its eligibility.

3.5.1 Inference-based architecture

Inference-based supervisor

Each inference-based decentralized supervisor *Sup* consists of n *inference-based* local supervisors $(Sup_i)_{i\in I}$, and its decisions are computed using the results of [Kumar et Takai, 2007] as follows. After the execution of $s \in \overline{K}$, each Sup_i has observed $P_i(s)$ and issues a local decision $c_i(P_i(s), \sigma)$ for each $\sigma \in \Sigma_c$. An ambiguity level $n_i(P_i(s), \sigma)$ is associated to each local decision. Formally :

$$Sup_i(P_i(s), \sigma) = (c_i(P_i(s), \sigma), n_i(P_i(s), \sigma)), \tag{3.10}$$

The generic global decision of each *Sup* was defined by (3.6). In the present case of inference-based decentralized supervisor *Sup*, its global decision is computed as follows [Kumar et Takai, 2007] : $\forall \sigma \in \Sigma_c, \forall s \in \overline{K}$,

$$Sup(s, \sigma) = \begin{cases} 1, & \text{if } \forall i \in I_\sigma; [n(s,\sigma) = n_i(P_i(s), \sigma) \Rightarrow c_i(P_i(s), \sigma) = 1], \\ 0, & \text{if } \forall i \in I_\sigma; [n(s,\sigma) = n_i(P_i(s), \sigma) \Rightarrow c_i(P_i(s), \sigma) = 0], \\ \phi, & otherwise. \end{cases} \tag{3.11}$$

Where $n(s, \sigma)$ denotes the minimal ambiguity level of local decisions of *Sup*, i.e.,

$$n(s, \sigma) = \min_{i \in I_\sigma} n_i(P_i(s), \sigma). \tag{3.12}$$

Definition 3.5.1 *Given an integer $N \geq 0$, an inference-based decentralized supervisor (i.e., defined by (3.10)-(3.12)) is said* N-inferring *(w.r.t. K), and denoted Inf_N-supervisor, if for any $\sigma \in \Sigma_c$, $\min\{n^d_\sigma, n^e_\sigma\} \leq N$, where $n^d_\sigma = \max_{\{s \in \overline{K} | s\sigma \in \mathcal{L} \wedge Sup(s,\sigma)=0\}} n(s, \sigma)$ and $n^e_\sigma = \max_{\{s \in \overline{K} | s\sigma \in \mathcal{L} \wedge Sup(s,\sigma)=1\}} n(s, \sigma)$.*

Note that N is the maximal ambiguity level that is used in a Inf_N-supervisor.

Computing the local decisions $c_i(P_i(s), \sigma)$ and $n_i(P_i(s), \sigma)$

We proceed as in [Kumar et Takai, 2007], but for a convenient presentation of the inference-based multi-decision (in the following Section 3.5.2), we use languages (E, D) such that $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$, instead of $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$. We define inductively a monotonically decreasing sequence of language pairs $(E[k], D[k])$ as follows.

Basis : $E[0] = E$ and $D[0] = D$,

Inductive step : for $k \geq 1$,

$$E[k+1] = E[k] \cap \bigcap_{i \in I_\sigma} P_i^{-1} P_i(D[k]),$$
$$D[k+1] = D[k] \cap \bigcap_{i \in I_\sigma} P_i^{-1} P_i(E[k]).$$

The decisions of every *Sup* are defined by (3.10)-(3.12) from $(c_i(P_i(s), \sigma), n_i(P_i(s), \sigma))_{i \in I_\sigma}$. Now we propose a method of [Kumar et Takai, 2007] to compute $c_i(P_i(s), \sigma)$ and $n_i(P_i(s), \sigma)$, for $i \in I_\sigma$. More precisely, using the sequence $(E[k], D[k])$ of language pairs, every local supervisor Sup_i ($i \in I_\sigma$) computes, for every $s \in \overline{K}$ and $\sigma \in \Sigma_c$, $n_i(P_i(s), \sigma)$ and $c_i(P_i(s), \sigma)$ as follows.

$$\begin{aligned} n_i^e(P_i(s), \sigma) &= \min\{k \in \mathbb{Z}^+ \mid P_i^{-1} P_i(s) \cap D[k] = \emptyset\}, \\ n_i^d(P_i(s), \sigma) &= \min\{k \in \mathbb{Z}^+ \mid P_i^{-1} P_i(s) \cap E[k] = \emptyset\}. \end{aligned} \tag{3.13}$$

A local decision is issued by comparing the two ambiguity levels, $n_i^e(P_i(s), \sigma)$ and $n_i^d(P_i(s), \sigma)$, giving preference to the smallest one. This is formalized as follows. For every $\sigma \in \Sigma_c$, the decision and ambiguity level of a local supervisor Sup_i following an observation $P_i(s)$, i.e., $Sup_i(P_i(s), \sigma) = (c_i(P_i(s), \sigma), n_i(P_i(s), \sigma))$, is computed as follows :

$$c_i(P_i(s), \sigma) = \begin{cases} 0, & \text{if } n_i^d(P_i(s), \sigma) < n_i^e(P_i(s), \sigma), \\ 1, & \text{if } n_i^e(P_i(s), \sigma) < n_i^d(P_i(s), \sigma), \\ \phi, & \text{otherwise}, \end{cases} \tag{3.14}$$

and

$$n_i(P_i(s), \sigma) = \min\{n_i^d(P_i(s), \sigma), n_i^e(P_i(s), \sigma)\}. \tag{3.15}$$

We have not explained in detail the (3.10)-(3.15) because they have been taken from the inference-based framework of [Kumar et Takai, 2007].

Eligibility of the inference-based architecture

In order to show that the inference-based architecture is eligible, we define the notion of Inf_N-COOBS (or N-inference-coobservability) as introduced in [Kumar et Takai, 2007]. For a convenient presentation of the inference-based multi-decision (in Section 3.5.2), we do not use exactly the definition of [Kumar et Takai, 2007]. We rather generalize it by the use of subsets $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$ instead of $\mathcal{E}_\sigma$ and $\mathcal{D}_\sigma$, respectively.

Definition 3.5.2 *Consider $K \subseteq \mathcal{L}_m$, $\sigma \in \Sigma_c$, $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$. (E, D) is said Inf_N-*coobservable *(or Inf_N-*COOBS*) if $E[N+1] = \emptyset$ or $D[N+1] = \emptyset$. K is said Inf_N-*COOBS *if, $\forall \sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is Inf_N-*COOBS.

We have the following lemmas 3.5.1 and 3.5.2 that state respectively that the inference-based architecture satisfy the conditions ELC-a and ELC-b. From these two lemmas, we then deduce lemma 3.5.3 by definition of the eligibility.

Lemma 3.5.1 *Consider $K \subseteq \mathcal{L}_m$ and a Inf_N-supervisor Sup. For every event $\sigma \in \Sigma_c$, $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$, if Sup is consistent w.r.t. (E, D) then (E, D) is Inf_N-*COOBS.

Lemma 3.5.2 *Consider $K \subseteq \mathcal{L}_m$ and the Inf_N-supervisor SUP given by (3.13)-(3.15). For every event $\sigma \in \Sigma_c$, $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$, if (E, D) is Inf_N-*COOBS *then SUP is consistent w.r.t. (E, D).*

Lemma 3.5.3 *The inference-based architecture is eligible.*

3.5.2 Parallel inference-based architectures running in parallel

Recall that a $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$ architecture consists of p D^j-architectures running in parallel and whose global decisions are fused disjunctively. When the p architectures are inference-based with respective ambiguities $N_1, \ldots, N_p$, we are in the presence of an inference-based multi-decision architecture, denoted $\vee$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-architecture. Any corresponding supervisor $Sup = ((Sup)_{j \in J}, \vee)$ consists on p Inf_{N_j}-supervisors $(Sup^j)_{j \in J}$, and is denoted $\vee$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-supervisor. For each Inf_{N_j}-architecture, the local decisions are computed by using (3.10) but with a superscript j, i.e., $Sup_i^j(P_i(s), \sigma) = (c_i^j(P_i(s), \sigma), n_i^j(P_i(s), \sigma))$, and the global decision $Sup^j(s, \sigma)$ is computed by using (3.11) but with a superscript j. The effective decision $Sup(s, \sigma)$ of the $\vee$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-supervisor is computed by combining disjunctively the global decisions $(Sup^j(s, \sigma))_{j \in J}$ using (3.8).

As already explained in Subsection 3.4.2, we will use a decomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ for every $\sigma \in \Sigma_c$. This decomposition is a practical rule for computing the inference-based local decisions as described in (3.13)-(3.15) for every Inf_{N_j}-architecture, by taking $E = \mathcal{E}_\sigma^j$ and $D = \mathcal{D}_\sigma$. Hence, the sequence of language pairs $(\mathcal{E}_\sigma^j[k], \mathcal{D}_\sigma^j[k])$ are computed as follows. For $k = 0$, $\mathcal{E}_\sigma^j[0] = \mathcal{E}_\sigma^j$ and $\mathcal{D}_\sigma^j[0] = \mathcal{D}_\sigma$. For $k \geq 1$, $\mathcal{E}_\sigma^j[k+1] = \mathcal{E}_\sigma^j[k] \cap \bigcap_{i \in I_\sigma} P_i^{-1} P_i(\mathcal{D}_\sigma^j[k])$ and $\mathcal{D}_\sigma^j[k+1] = \mathcal{D}_\sigma^j[k] \cap \bigcap_{i \in I_\sigma} P_i^{-1} P_i(\mathcal{E}_\sigma^j[k])$. For simplicity, j is omitted when $p = 1$ (i.e., no decomposition). Before studying the existence of solutions for the inference-based multi-decision architecture, we illustrate the multi-decision framework, applied to the inference-based architecture, in the following example.

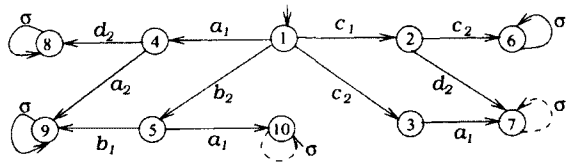

Figure 3.2 Plant and specification. The latter is obtained by forbidding the dashed transitions

Example 3.5.1 *Figure 3.2 presents a plant G controlled by two local supervisors, the first one observes $\Sigma_{o,1} = \{a_1, b_1, c_1\}$, while the second observes $\Sigma_{o,2} = \{a_2, b_2, c_2, d_2\}$, and $\Sigma_{c,1} = \Sigma_{c,2} = \{\sigma\}$. The specification is obtained by erasing the two dashed selfloops. We consider that all states of the plant and the specification are marked, which implies that (3.1) and (3.2) are equivalent. $\mathcal{D}_\sigma = \{c_1d_2, c_2a_1, b_2a_1\}$ is accepted by the FSA obtained from the specification by removing states 4, 6, 8 and 9, and marking states 7 and 10. $\mathcal{E}_\sigma = \{a_1d_2\sigma^*, a_1a_2\sigma^*, b_2b_1\sigma^*, c_1c_2\sigma^*\}$ is accepted by the FSA obtained from the specification by removing states 3, 7 and 10, and marking states 6, 8 and 9. Let us consider the cases $p = 1$ and $p = 2$.*

Case *$p = 1$: Since j takes the single value 1, it will be implicit in all equations. We have $\mathcal{E}_\sigma[0] = \mathcal{E}_\sigma$ and $\mathcal{D}_\sigma[0] = \mathcal{D}_\sigma$, and there is no Inf_N-supervisor satisfying Conds. (3.1) and (3.2), $\forall N \in \mathbb{Z}^+$. Indeed, after the execution of trace c_1d_2, the language pairs $(\mathcal{E}_\sigma[k], \mathcal{D}_\sigma[k])_{k\geq 0}$ are then computed as follows : First step ($k = 1$), we compute $\mathcal{D}_\sigma[1] = \{b_2a_1, c_2a_1, c_1d_2\}$ and $\mathcal{E}_\sigma[1] = \{a_1d_2\sigma^*, c_1c_2\sigma^*\}$. Second step ($k = 2$), we compute $\mathcal{D}_\sigma[2] = \{c_2a_1, c_1d_2\}$ and $\mathcal{E}_\sigma[2] = \mathcal{E}_\sigma[1]$. Third step ($k = 3$), we compute $\mathcal{D}_\sigma[3] = \mathcal{D}_\sigma[2]$ and $\mathcal{E}_\sigma[3] = \mathcal{E}_\sigma[2] = \mathcal{E}_\sigma[1]$.*

We obtain $\mathcal{D}_\sigma[k] = \mathcal{D}_\sigma[2] = \{c_2a_1, c_1d_2\}, \forall k \geq 2$, and $\mathcal{E}_\sigma[k] = \mathcal{E}_\sigma[1] = \{a_1d_2\sigma^*, c_1c_2\sigma^*\}$, $\forall k \geq 1$. Since, $\forall k \geq 2$, $c_1d_2 \in \mathcal{D}_\sigma[k]$, we have, $\forall i = 1, 2$, $\forall k \geq 1$, $P_i^{-1}P_i(c_1d_2) \cap \mathcal{D}_\sigma[k] \neq \emptyset$. And from (3.13), $n_i^e(P_i(c_1d_2), \sigma)$ is infinite. Moreover, $\forall k \geq 1$, $P_1^{-1}P_1(c_1d_2) \cap \mathcal{E}_\sigma[k] = \{c_1c_2\sigma^*\}$, $P_2^{-1}P_2(c_1d_2) \cap \mathcal{E}_\sigma[k] = \{a_1d_2\sigma^*\}$. Therefore, $\forall i = 1, 2$, $\forall k \geq 1$, $P_i^{-1}P_i(c_1d_2) \cap \mathcal{E}_\sigma[k] \neq \emptyset$. And from (3.13), $n_i^d(P_i(c_1d_2), \sigma)$ is infinite Since $n_i^d(P_i(c_1d_2), \sigma)$ and $n_i^e(P_i(c_1d_2), \sigma)$ are infinite, we deduce by (3.14) that, $\forall i = 1, 2$, $c_i(P_i(c_1d_2), \sigma) = \phi$. Then, by (3.11), $Sup(s, \sigma) = \phi$, which violates Cond. (3.4), and thus, there is no Inf_N-supervisor that can enforce the plant to conform to the specification.

Case *$p = 2$: Consider the decomposition $\{\mathcal{E}_\sigma^1, \mathcal{E}_\sigma^2\}$ of $\mathcal{E}_\sigma$ where $\mathcal{E}_\sigma^1 = \{a_1d_2\sigma^*, a_1a_2\sigma^*, b_2b_1\sigma^*\}$ and $\mathcal{E}_\sigma^2 = \{c_1c_2\sigma^*\}$. The language pairs $(\mathcal{E}_\sigma^j[k], \mathcal{D}_\sigma^j[k])_{k\geq 0}$ are then computed as follows :*

For $j = 1$:

First step ($k = 1$), *we compute* $\mathcal{D}^1_\sigma[1] = \{b_2a_1\}$ *and* $\mathcal{E}^1_\sigma[1] = \{a_1d_2\sigma^*\}$.

Second step ($k = 2$), *we compute* $\mathcal{D}^1_\sigma[2] = \emptyset$ *and* $\mathcal{E}^1_\sigma[2] = \emptyset$.

For $j = 2$:

First step ($k = 1$), *we compute* $\mathcal{D}^2_\sigma[1] = \emptyset$ *and* $\mathcal{E}^2_\sigma[1] = \{c_1c_2\sigma^*\}$.

Second step ($k = 2$), *we compute* $\mathcal{D}^2_\sigma[2] = \emptyset$ *and* $\mathcal{E}^2_\sigma[2] = \emptyset$.

Therefore, $\mathcal{D}^1_\sigma[k] = \mathcal{E}^1_\sigma[k] = \mathcal{E}^2_\sigma[k] = \emptyset$, $\forall k \geq 2$, *and* $\mathcal{D}^2_\sigma[k] = \emptyset$, $\forall k \geq 1$. *Using (3.10)-(3.15), Table 3.1 (resp., Table 3.2) presents the local and global decisions taken by the* Inf_1*-supervisor* Sup^1 *(resp.,* Inf_0*-supervisor* Sup^2*) for all traces where a decision on event* σ *is relevant, i.e., traces at the term of which* σ *is permitted by the plant (formally, traces of* $\mathcal{E}_\sigma \cup \mathcal{D}_\sigma$*). Note that, for* $j \in \{1, 2\}$, *the expressions* $n_1^{d,j}$, $n_1^{e,j}$ *and* Sup_1^j *(resp.,* $n_2^{d,j}$, $n_2^{e,j}$ *and* Sup_2^j*) are computed for* $P_1(s)$ *(resp.,* $P_2(s)$*) and* Sup^j *is computed for* s. *Table 3.3 presents the effective decisions computed using (3.8), taken by the* $\vee$-(Inf_1, Inf_0)*-supervisor that combines disjunctively the decisions of* Sup^1 *and* Sup^2. *We see in Table 3.3 that* $Sup(s, \sigma) = 1$ *for all traces* $s \in \mathcal{E}_\sigma$, *and* $Sup(s, \sigma) = 0$ *for all traces* $s \in \mathcal{D}_\sigma$, *which corresponds to satisfying Conds (3.3) and (3.4). We will explain in Section 3.5.3 why we have selected exactly* $N_1 = 1$ *and* $N_2 = 0$.

This example illustrates the fact that the set of languages achievable by the multi-decision inference architecture (for $p \geq 1$*) includes the set of languages achievable by the inference architecture of [Kumar et Takai, 2007] (corresponding to* $p = 1$*).*

Trace	$n_1^{d,1}$	$n_1^{e,1}$	Sup_1^1	$n_2^{d,1}$	$n_2^{e,1}$	Sup_2^1	Sup^1
$s = c_1d_2$	0	1	(0,0)	2	1	(1,1)	0
$s = c_2a_1$	2	2	(ϕ,2)	0	1	(0,0)	0
$s = b_2a_1$	2	2	(ϕ,2)	1	2	(0,1)	0
$s \in a_1d_2\sigma^*$	2	2	(ϕ,2)	2	1	(1,1)	1
$s \in a_1a_2\sigma^*$	2	2	(ϕ,2)	1	0	(1,0)	1
$s \in b_2b_1\sigma^*$	1	0	(1,0)	1	2	(0,1)	1
$s \in c_1c_2\sigma^*$	0	1	(0,0)	0	1	(0,0)	0

Tableau 3.1 Local and global decisions taken by the Inf_1-supervisor Sup^1 (on $\sigma \in \Sigma_c$) computed by (3.10)-(3.15) for the plant and specification of Fig. 3.2 w.r.t. $(\mathcal{E}^1_\sigma, \mathcal{D}_\sigma)$, for $\mathcal{E}^1_\sigma = \{a_1d_2\sigma^*, a_1a_2\sigma^*, b_2b_1\sigma^*\}$.

3.5.3 Existence of solutions for the inference-based multi-decision architecture

In this section, we introduce the notion of $\vee$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-COOBS in order to characterize the class of languages achievable under the control of a nonblocking and admissible

Trace	$n_1^{d,2}$	$n_1^{e,2}$	Sup_1^2	$n_2^{d,2}$	$n_2^{e,2}$	Sup_2^2	Sup^2
$s = c_1 d_2$	2	1	(1,1)	0	1	(0,0)	0
$s = c_2 a_1$	0	1	(0,0)	2	1	(1,1)	0
$s = b_2 a_1$	0	1	(0,0)	0	1	(0,0)	0
$s \in a_1 d_2 \sigma^*$	0	1	(0,0)	0	1	(0,0)	0
$s \in a_1 a_2 \sigma^*$	0	1	(0,0)	0	0	(ϕ,0)	ϕ
$s \in b_2 b_1 \sigma^*$	0	0	(ϕ,0)	0	1	(0,0)	ϕ
$s \in c_1 c_2 \sigma^*$	2	1	(1,1)	2	1	(1,1)	1

Tableau 3.2 Local and global decisions taken by the Inf_0-supervisor Sup^2 (on $\sigma \in \Sigma_c$) computed by (3.10)-(3.15) for the plant and specification of Fig. 3.2 w.r.t. $(\mathcal{E}_\sigma^1, \mathcal{D}_\sigma)$, for $\mathcal{E}_\sigma^2 = \{c_1 c_2 \sigma^*\}$.

Trace	$Sup^1(s)$	$Sup^2(s)$	$Sup(s)$
$s = c_1 d_2$	0	0	0
$s = c_2 a_1$	0	0	0
$s = b_2 a_1$	0	0	0
$s \in a_1 d_2 \sigma^*$	1	0	1
$s \in a_1 a_2 \sigma^*$	1	ϕ	1
$s \in b_2 b_1 \sigma^*$	1	ϕ	1
$s \in c_1 c_2 \sigma^*$	0	1	1

Tableau 3.3 Global and effective decisions taken by the $\vee - (Inf_0, Inf_1)$-supervisor (on $\sigma \in \Sigma_c$) computed by applying (3.8) to the decisions $Sup^1(s)$ and $Sup^2(s)$ of Tables 3.1 and 3.2.

$\vee$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-supervisor. By using the notion of Inf_N-COOBS defined in Definition 3.5.2, we introduce in the following definition the notion of $\vee$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-COOBS.

Definition 3.5.3 *Consider $K \subseteq \mathcal{L}_m$, and $\sigma \in \Sigma_c$. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is said $\vee$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-*COOBS *if there exists a decomposition $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p)$ of $\mathcal{E}_\sigma$ such that, $\forall j \in J$, $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is Inf_{N_j}-*COOBS. *$(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is said $\vee$-$Inf_{\leq N}^{\geq 1}$-*COOBS *if it is $\vee$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-*COOBS *for some $p \geq 1$ and some positive integers $N_1, \ldots, N_p \leq N$. K is said $\vee$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-*COOBS *if, $\forall \sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-*COOBS. *$K$ is said $\vee$-$Inf_{\leq N}^{\geq 1}$-*COOBS *if, $\forall \sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf_{\leq N}^{\geq 1}$-*COOBS.

Since the inference-based architecture is eligible, as shown in Lemma 3.5.3, the following corollary, which is a straightforward result of Theorem 3.4.2, states necessary and sufficient conditions for a language to be achievable under the $\vee$-($Inf_{N_1}, \ldots, Inf_{N_p}$) architecture.

Corollary 3.5.1 *Given a language $K \subseteq \mathcal{L}_m$, there exists a nonblocking and admissible $\vee$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-supervisor $Sup = ((Sup^j)_{j \in J}, \vee)$ such that $\mathcal{L}(Sup/G) = \overline{K}$ and $\mathcal{L}_m(Sup/G) = K$ if and only if K is $\mathcal{L}_m$-closed, controllable and $\vee$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-*COOBS.

Let us return to the example 3.5.1 where $\mathcal{E}_\sigma$ is represented by the automaton $\mathcal{A}_{\mathcal{E}_\sigma}$ obtained from the automaton of Fig. 3.2 by removing the states 3, 7 and 10, and marking the states 6, 8 and 9. From Def. 3.5.2, we have that $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is *not* Inf_N-COOBS, because $\mathcal{D}_\sigma[k] = \{c_2a_1, c_1d_2\} \neq \emptyset$, $\forall k \geq 2$, and $\mathcal{E}_\sigma[k] = \{a_1d_2\sigma^*, c_1c_2\sigma^*\} \neq \emptyset$, $\forall k \geq 1$. From Corollary 3.5.1, we deduce that there exists no Inf_N-supervisor that can enforce the plant to conform to the specification. Note that this result was observed in the case $p = 1$ of Example 3.5.1. If we use the decomposition $\{\mathcal{E}_\sigma^1, \mathcal{E}_\sigma^2\}$ of $\mathcal{E}_\sigma$ where $\mathcal{E}_\sigma^1 = \{a_1d_2\sigma^*, a_1a_2\sigma^*, b_2b_1\sigma^*\}$ and $\mathcal{E}_\sigma^2 = \{c_1c_2\sigma^*\}$, corresponding to states (8,9) and 6, respectively, we compute $\mathcal{D}_\sigma^1[2] = \emptyset$ and $\mathcal{D}_\sigma^2[1] = \emptyset$. From Defs. 3.5.2 and 3.5.3, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-(Inf_1,Inf_0)-COOBS. K is controllable because it is obtained from the plant by forbidding $\sigma \in \Sigma_c$ in some states. K is also $\mathcal{L}_m$-closed because $\mathcal{L}$ and K are prefix-closed. From Corollary 3.5 1, we deduce that there exists a $\vee$-(Inf_1,Inf_0)-supervisor that can enforce the plant to conform to the specification. Such a $\vee$-(Inf_1,Inf_0)-supervisor was in fact computed in the case $p = 2$ of Example 3.5.1 of Subsection 3.5.2, and was represented in Tables 3.1, 3.2 and 3.3.

By comparing Defs. 3.4.6-3.4.7 with Def. 3.5.3, and since the inference-based architecture is eligible, we deduce straightforwardly (and hence, the proof is omitted) the following proposition

Proposition 3.5.1 *If $K \subseteq \mathcal{L}_m$ is $\vee$-$Inf_{\leq N}^{\geq 1}$-*COOBS *then it is $\vee$-$ELA^{\geq 1}$-*COOBS.

3.6 Some properties related to multi-decision architectures

We have identified many properties related to the coobservabilty of the $\vee$-ELA$^{\geq 1}$ architecture. In this section, we have selected some of the most relevant properties. In the next proposition, we state a necessary condition for a language to be $\vee$-ELA$^{\geq 1}$-COOBS.

Proposition 3.6.1 *If a language $K \subseteq \mathcal{L}_m$ is $\vee$-ELA$^{\geq 1}$-*COOBS *then, $\forall \sigma \in \Sigma_c$, $\forall s, t \in \overline{K}$ s.t. $s\sigma, t\sigma \in \mathcal{L}$, we have : $[\forall i \in I_\sigma, P_i(s) = P_i(t)] \Rightarrow [s \in \mathcal{E}_\sigma \Leftrightarrow t \in \mathcal{E}_\sigma]$.*

3.6.1 Comparison with other architectures

The existence of eligible architecture D such that K is D-COOBS is simply obtained from the definition of $\vee$-ELA$^{\geq 1}$-COOBS by restricting ourself, for each $\sigma \in \Sigma_c$, to the trivial partition $\{\mathcal{E}_\sigma\}$. Hence, we deduce straightforwardly (thus, the proof is omitted) the following proposition which implies that any $\vee$-ELA$^{\geq 1}$ architecture permits to achieve more languages than any eligible decentralized architecture.

Proposition 3.6.2 *If $K \subseteq \mathcal{L}_m$ is not $\vee$-ELA$^{\geq 1}$-*COOBS*, then K is not* D-COOBS *for any eligible architecture* D.

If we apply Prop. 3.6.2 to the particular case of inference-based architecture (which is eligible, from Lemma 3.5.3), we have : if K is Inf_N-COOBS then it is $\vee$-ELA$^{\geq 1}$-COOBS. It is proved in [Kumar et Takai, 2007] that each of C&P-COOBS, D&A-COOBS, COND-C&P-COOBS and COND-D&A-COOBS implies Inf_N-COOBS. Then, we deduce straightforwardly (and hence the proof is omitted) the following proposition :

Proposition 3.6.3 *If $K \subseteq \mathcal{L}_m$ is not $\vee$-ELA$^{\geq 1}$-*COOBS*, then K is none of the following : C&P-*COOBS*, D&A-*COOBS*, COND-C&P-*COOBS*, COND-D&A-*COOBS *or Inf_N-*COOBS.

Now we compare the centralized architecture under partial observation with the $\vee$-ELA$^{\geq 1}$ architecture. Consider the natural projection $P : \Sigma^* \longrightarrow \Sigma_o^*$ that models the partial observation of a centralized architecture. After the execution of a trace s, a centralized supervisor has observed $P(s)$ and issues a decision $dec(P(s)) \in \{1, 0, \phi\}$. Observability is defined by :

Definition 3.6.1 *[Lin et Wonham, 1988b] K is said observable w.r.t. $\mathcal{L}$, Σ_o if and only if, $\forall \sigma \in \Sigma_c$, $\forall s, t \in \overline{K}$, s.t. $s\sigma, t\sigma \in \mathcal{L}$: $[P(s) = P(t)] \Rightarrow [s \in \mathcal{E}_\sigma \Leftrightarrow t \in \mathcal{E}_\sigma]$.*

The following proposition states that the centralized architecture permits to control more languages than the disjunctive multi-decision architecture.

Proposition 3.6.4 *If $K \subseteq \mathcal{L}_m$ is not observable w.r.t. $\mathcal{L}$, Σ_o, then K is not $\vee$-ELA$^{\geq 1}$-*COOBS *w.r.t. $\mathcal{L}$, $\Sigma_{o,1}$, ..., $\Sigma_{o,n}$.*

Note that the converse of Proposition 3.6.4 is not true. Consider for example the language plant $\mathcal{L} = \overline{\{\alpha a_1 a_2 \sigma, \beta a_2 a_1 \sigma\}}$ and a specification $K = \overline{\{\alpha a_1 a_2 \sigma, \beta a_2 a_1\}}$, where $\Sigma_o = \{a_1, a_2\}$ and $\Sigma_c = \{\sigma\}$. K is observable w.r.t. $\mathcal{L}$ and Σ_o, because the single supervisor of a centralized architecture observes the order in which a_1 and a_2 are executed, and thus, can correctly decide if σ must be enabled or disabled. Let us show that the specification is not $\vee$-ELA$^{\geq 1}$-COOBS w.r.t. $\mathcal{L}$, $\Sigma_{o,1} = \{a_1\}$ and $\Sigma_{o,2} = \{a_2\}$. In fact, since $\alpha a_1 a_2 \in \mathcal{E}_\sigma$, $\beta a_2 a_1 \in \mathcal{D}_\sigma$ and $P_i(\alpha a_1 a_2) = P_i(\beta a_2 a_1)$ $(i = 1, 2)$, we have, from Proposition 3.6.1, K is not $\vee$-ELA$^{\geq 1}$-COOBS.

In the following proposition we compare the disjunctive multi-decision architecture with the eligible architectures in the case where $n = 1$.

Proposition 3.6.5 *When a single local supervisor is used ($n = 1$, centralized architectures in parallel), then the following assertions are equivalent :*

1. *K is* D-COOBS *for some eligible architecture* D,
2. *K is $\vee$-ELA$^{\geq 1}$-*COOBS,
3. *K is observable.*

Proposition 3.6.5 means that the multi-decision framework is irrelevant when $n = 1$, that is, it is useless to take $p > 1$ when $n = 1$. More precisely, several centralized supervisors in parallel produce the same observability as a single centralized supervisor.

3.6.2 Closure under union and intersection

If a language K is $\vee$-ELA$^{\geq 1}$-COOBS then there exist an integer $p \geq 1$ and some eligible architectures $\mathrm{D}^1, \ldots, \mathrm{D}^p$ such that K is $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS, which is a *necessary* condition for the existence of a multi-decision supervisor. Hence, it is relevant to determine how to tackle the situation where a language K is *not* $\vee$-ELA$^{\geq 1}$-COOBS. Traditionally, depending on the exact objective, the idea is to determine whether there exists a supremal sublanguage or an infimal superlanguage of K which is $\vee$-ELA$^{\geq 1}$-COOBS. The existence of supremal and infimal languages is related implicitly to the closure under union and intersection of classes of languages. We therefore provide some algebraic properties of $\vee$-ELA$^{\geq 1}$-coobservability such as closure under intersection and union. We have the following proposition related to the closure of $\vee$-ELA$^{\geq 1}$-COOBS under union of languages.

Proposition 3.6.6 *$\vee$-ELA$^{\geq 1}$-*COOBS *is not preserved under union of languages.*

Proposition 3.6.6 can be proved by the following example. Let $\mathcal{L} = \overline{\{\alpha a_1 a_2 \sigma, \beta a_2 a_1 \sigma\}}$ and a specification $K = \overline{\{\alpha a_1 a_2 \sigma, \beta a_2 a_1\}}$, where $\Sigma_{c,1} = \Sigma_{c,2} = \{\sigma\}$, $\Sigma_{o,1} = \{a_1\}$ and $\Sigma_{o,2} = \{a_2\}$. We have K is not $\vee$-ELA$^{\geq 1}$-COOBS since for $\alpha a_1 a_2 \in \mathcal{E}_\sigma$ and $\beta a_2 a_1 \in \mathcal{D}_\sigma$ we have $P_i(\alpha a_1 a_2) = P_i(\beta a_2 a_1)$, $i = 1, 2$. Thus, by Proposition 3.6.1, K is not $\vee$-ELA$^{\geq 1}$-COOBS. Let us decompose K into $K_1 = \overline{\{\alpha a_1 a_2 \sigma\}}$ and $K_2 = \overline{\{\beta a_1 a_2\}}$, and thus, $K = K_1 \cup K_2$. K_1 and K_2 are Inf_0-COOBS (and thus $\vee$-ELA$^{\geq 1}$-COOBS, from Prop. 3.6.3) because $\mathcal{D}_\sigma(K_1) = \emptyset$ and $\mathcal{E}_\sigma(K_2) = \emptyset$, respectively.

The closure of the $\vee$-ELA$^{\geq 1}$-COOBS under the intersection is in general not preserved except for a special case. Before showing that $\vee$-ELA$^{\geq 1}$-COOBS is not preserved in general, we treat the special case where $\vee$-ELA$^{\geq 1}$-COOBS is preserved under the intersection of prefix-closed languages.

Theorem 3.6.1 *Given two prefix-closed languages K_1 and K_2 such that K_1 is $\vee$-ELA^{p_1}-COOBS and K_2 is $\vee$-ELA^{p_2}-COOBS, then $K_1 \cap K_2$ is $\vee$-ELA$^{p_1 p_2}$-COOBS.*

Intuitively, consider a prefix-closed K_1 (resp., K_2) that can be achieved under the control of p_1 (resp., p_2) eligible architectures running in parallel and combined disjunctively. From Theorem 3.6.1, we can deduce that $K_1 \cap K_2$ can be achieved under the control of $p_1 p_2$ eligible architectures running in parallel and combined disjunctively. Note that, the proof of Theorem 3.6.1 is constructive in the sense that it permits to construct, from multi-decision architectures A_1 and A_2 for which two prefix-closed languages L_1 and L_2 are respectively A_1-COOBS and A_2-COOBS, a multi-decision architecture A for which $L_1 \cap L_2$ is A-COOBS.

Now we present a result which states that $\vee$-ELA$^{\geq 1}$-COOBS is not necessarily preserved under intersection of K_1 and K_2 when K_1 or K_2 is not prefix-closed.

Proposition 3.6.7 *$\vee$-ELA$^{\geq 1}$-COOBS is not preserved under intersection of languages.*

The proof of the above proposition can be illustrated in the following example. For $\Sigma_{c,1} = \Sigma_{c,2} = \{\sigma\}$, $\Sigma_{o,1} = \{a_1\}$ and $\Sigma_{o,2} = \{a_2\}$, consider $\mathcal{L} = \overline{\{\alpha a_2 a_1 \sigma, a_1 a_2 \sigma \alpha, a_1 a_2 \sigma \beta\}}$ and specifications $K_1 = \{\alpha a_2 a_1 \sigma, a_1 a_2, a_1 a_2 \sigma \alpha\}$, $K_2 = \{\alpha a_2 a_1 \sigma, a_1 a_2, a_1 a_2 \sigma \beta\}$ and $K = K_1 \cap K_2 = \{\alpha a_2 a_1 \sigma, a_1 a_2\}$. We have $\mathcal{E}_\sigma(K_1) = \mathcal{E}_\sigma(K_2) = \{\alpha a_2 a_1 \sigma, a_1 a_2\}$ and $\mathcal{D}_\sigma(K_1) = \mathcal{D}_\sigma(K_2) = \emptyset$, hence K_1 and K_2 are Inf_0-COOBS, and then K_1 and K_2 are $\vee$-ELA$^{\geq 1}$-COOBS. We have $\mathcal{E}_\sigma(K) = \{\alpha a_2 a_1\}$ and $\mathcal{D}_\sigma(K) = \{a_1 a_2\}$, and since $P_i(\alpha a_2 a_1) = P_i(a_1 a_2)$ $(i = 1, 2)$, we have, from Prop. 3.6.1, K is not $\vee$-ELA$^{\geq 1}$-COOBS.

3.7 Multi-decision architectures with finite decompositions

From Theorem 3.4.2, determining the existence of a nonblocking and admissible $\vee$-(D^1, $\ldots, \mathrm{D}^p$)-supervisor necessitates to determine whether K is $\mathcal{L}_m$-closed, controllable and $\vee$-(D^1, $\ldots, \mathrm{D}^p$)-COOBS. $\mathcal{L}_m$-closure and controllability are classical notions that can be checked in the usual way since they do not depend on the control architecture. It remains therefore to determine whether K is $\vee$-($\mathrm{D}^1, \ldots, \mathrm{D}^p$)-COOBS. From Def. 3.4.6, we have to answer the following question :

Question 1 . *Does there exist a decomposition* $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p)$ *of* $\mathcal{E}_\sigma$ *such that,* $\forall j \in J$, $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ *is* D^j-COOBS *?*

What makes Question 1 difficult is that $\mathcal{E}_\sigma$ is in general infinite and decomposing infinite languages is known to be a challenging problem. We propose here a solution that transforms the problem of decomposing an infinite regular language into a problem of decomposing a *finite* set of states in a FSA. Our solution for decomposing $\mathcal{E}_\sigma$ is based on the use of a FSA $\mathcal{A}_{\mathcal{E}_\sigma}$ accepting $\mathcal{E}_\sigma$, that is, every trace $s \in \mathcal{E}_\sigma$ leads to a marked state of $\mathcal{A}_{\mathcal{E}_\sigma}$. We consider uniquely the decompositions satisfying the following assumption

A1 : Given a FSA $\mathcal{A}_{\mathcal{E}_\sigma}$ accepting $\mathcal{E}_\sigma$, the only eligible decompositions $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p)$ of $\mathcal{E}_\sigma$ are such that every $\mathcal{E}_\sigma^j$ consists of the traces leading to one or several marked states of $\mathcal{A}_{\mathcal{E}_\sigma}$. In other words, every $\mathcal{E}_\sigma^j$ corresponds to a subset of the set of marked states of $\mathcal{A}_{\mathcal{E}_\sigma}$.

The important thing is that with Assumption **A1**, and assuming regular languages, the number of eligible decompositions becomes *finite*. With **A1**, we have transformed the problem of decomposing an infinite regular language into a problem of decomposing a finite state set

Remark 3.7.1 *The present assumption* ***A1*** *is less restrictive than the assumption used in [Chakib et Khoumsi, 2008a,b], in the sense that the latter permits less decompositions. This is due to the fact that the present* ***A1*** *can be based on a nondeterministic FSA, while the assumption used in [Chakib et Khoumsi, 2008a,b] is expressed in a form which implies that the associated FSA is necessarily deterministic.*

We have the following definition to specify that a language is $\vee$-(D^1, $\ldots, \mathrm{D}^p$)-COOBS for decompositions satisfying **A1** w.r.t a specific FSA $\mathcal{A}_{\mathcal{E}_\sigma}$.

Definition 3.7.1 *Consider* $K \subseteq \mathcal{L}_m$, *and a tuple of FSAs* $[\mathcal{A}_{\mathcal{E}_\sigma}]_{\sigma \in \Sigma_c}$. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ *is said* $\vee$-(D^1, $\ldots, \mathrm{D}^p$)-COOBS *w.r.t.* $\mathcal{A}_{\mathcal{E}_\sigma}$, *if there exists a decomposition* $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ *of* $\mathcal{E}_\sigma$ *satis-*

fying ***A1*** *w.r.t.* $\mathcal{A}_{\mathcal{E}_\sigma}$ *such that,* $\forall j \in J$, $(\mathcal{E}^j_\sigma, \mathcal{D}_\sigma)$ *is* D^j-COOBS. *K is said* $\vee$-$(\mathrm{D}^1, \ldots \mathrm{D}^p)$-COOBS *w.r.t.* $[\mathcal{A}_{\mathcal{E}_\sigma}]_{\sigma\in\Sigma_c}$, *if,* $\forall\sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ *is* $\vee$-$(\mathrm{D}^1,\ldots,\mathrm{D}^p)$-COOBS *w.r.t* $\mathcal{A}_{\mathcal{E}_\sigma}$.

The stronger notion of $\vee$-ELA$^{\geq 1}$-COOBS (i.e., w.r.t. a FSM) can be adapted straightforwardly as in Definition 3.4.7. And as in Definition 3.4.6, we obtain the stronger notions of $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS and $\vee$-ELA$^{\geq 1}$-COOBS for a language K.

We have the following theorem that states a practical necessary and sufficient condition for $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ to be $\vee$-ELA$^{\geq 1}$-COOBS w.r t. a FSA. For that, we denote by $\mathcal{E}^x_\sigma$ the set of traces reaching the state x of a FSA $\mathcal{A}_{\mathcal{E}_\sigma}$ accepting $\mathcal{E}_\sigma$. Then, the specific decomposition (in fact a partition) $\{\mathcal{E}^{x_1}_\sigma, \ldots, \mathcal{E}^{x_p}_\sigma\}$ satisfies Assumption **A1**.

Theorem 3.7.1 *Consider a FSA* $\mathcal{A}_{\mathcal{E}_\sigma}$ *accepting* $\mathcal{E}_\sigma$, *and let* $\{x_1, \ldots, x_p\}$ *be the set of marked states of* $\mathcal{A}_{\mathcal{E}_\sigma}$. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ *is* $\vee$-*ELA*$^{\geq 1}$-COOBS *w.r.t.* $\mathcal{A}_{\mathcal{E}_\sigma}$ *if and only if for every marked state* x_j *of* $\mathcal{A}_{\mathcal{E}_\sigma}$, $(\mathcal{E}^{x_j}_\sigma, \mathcal{D}_\sigma)$ *is* D^j-COOBS *for some eligible architecture* D^j.

By considering only decompositions satisfying **A1** w.r.t. a FSA $\mathcal{A}_{\mathcal{E}_\sigma}$, p is bounded by the number of marked states of $\mathcal{A}_{\mathcal{E}_\sigma}$. The following corollary is deduced from Theorem 3 4 2.

Corollary 3.7.1 *Consider eligible architectures* $(\mathrm{D}^j)_{j\in J}$, *a language* $K \subseteq \mathcal{L}_m$ *and tuple of FSAs* $[\mathcal{A}_{\mathcal{E}_\sigma}]_{\sigma\in\Sigma_c}$. *If K is* $\mathcal{L}_m$*-closed, controllable and* $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS *w.r.t.* $[\mathcal{A}_{\mathcal{E}_\sigma}]_{\sigma\in\Sigma_c}$, *then there exists a nonblocking and admissible* $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$*-supervisor* $Sup = ((Sup^j)_{j\in J}, \vee)$ *such that* $\mathcal{L}(Sup/G) = \overline{K}$ *and* $\mathcal{L}_m(Sup/G) = K$.

The following Remark 3.7.2 can be seen as a proof of Corollary 3 7 1. Such intuitive proof is sufficient since the deduction of Corollary 3.7.1 from Theorem 3.4.2 is quite easy to understand.

Remark 3.7.2 *If we compare Theorem 3 4.2 and Corollary 3.7.1, we see that the latter contains a* sufficient *condition (use of* if*) while the former contains a* necessary and sufficient *condition (use of* iff*). This is due to the fact that in Corollary 3.7.1, we have used Assumption* ***A1****, which is not actually necessary for the existence of* $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-*supervisor. Assumption* ***A1*** *has been added for solving the problem of decomposing infinite languages.*

3.8 On the verification of $\vee$-$Inf^{\geq 1}_{\leq N}$-Coobs

In relation with the study of the disjunctive inference-based multi-decision architecture, we have studied how to check its corresponding coobservability. We have developed an algorithm that checks whether $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS w.r t. a given FSA $\mathcal{A}_{\mathcal{E}_\sigma}$, and if

yes, computes a specific decomposition satisfying Assumption **A1**. (As already mentioned, the decomposition of $\mathcal{E}_\sigma$ is transformed into a decomposition of the set of marked states of the automaton $\mathcal{A}_{\mathcal{E}_\sigma}$ representing $\mathcal{E}_\sigma$.)

An article which has just been finalized presents such algorithm and studies its complexity in terms of memory and computation. Let us give a brief outline of such study. In fact, the algorithm works on the two points in parallel, that is, for a given $\sigma \in \Sigma_c$, it constructs gradually a partition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ verifying a specific condition C. Our approach is justified by the fact that if for a partition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ satisfying condition C, there exists $\mathcal{E}_\sigma^j$ such that $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is not Inf_N-COOBS, then $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS w.r.t. the given FSA $\mathcal{A}_{\mathcal{E}_\sigma}$. Note that condition C is satisfied by the trivial partition where each $\mathcal{E}_\sigma^j$ corresponds to a marked state of $\mathcal{A}_{\mathcal{E}_\sigma}$. We have used the multi-marking principle of [de Queiroz *et al.*, 2005] to construct partitions $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p)$ of $\mathcal{E}_\sigma$ and determine whether $\mathcal{E}_\sigma^j[k]$ or $\mathcal{D}_\sigma^j[k]$ is empty, for $k \geq 0$ and $j \leq p$.

We have compared the complexities of the algorithm in the following two cases : 1) $\mathcal{E}_\sigma$ is not decomposed, 2) decomposition is permitted. We obtain the same worst-case complexity, in terms of Big-oh notation, for the two cases. Consequently, the worst case complexities for checking $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS and the Inf_N-COOBS are comparable. Intuitively, this result may be surprising. A possible explanation is that we have restricted the set of possible decompositions in two ways :

1. We compute a specific decomposition satisfying **A1** (Section 3.7). In this way, we have eliminated the decompositions that do not satisfy **A1** w.r.t. a given FSA. The number of these eliminated decompositions may be infinite, while the number of the remaining eligible decompositions is certainly finite.
2. Our algorithm considers only certain partitions respecting a given condition C, such that if we have not coobservability for these partitions, there exists no decomposition respecting **A1** for which we have coobservability.

In the following explanations, we consider uniquely the particular case where the constructed partition $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p)$ is such that each $\mathcal{E}_\sigma^j$ corresponds to a single marked state of $\mathcal{A}_{\mathcal{E}_\sigma}$. We use the multi-marking principle of [de Queiroz *et al.*, 2005] to determine whether $\mathcal{E}_\sigma^j[k]$ or $\mathcal{D}_\sigma^j[k]$ is empty, for $k \geq 0$ and $j \leq p$. For each $j \in J$, a color (say c^j) is associated to the state of $\mathcal{A}_{\mathcal{E}_\sigma}$ that is reached by traces of $\mathcal{E}_\sigma^j$. And $c^j \neq c^{j'}$ when $j \neq j'$, that is, two distinct $\mathcal{E}_\sigma^j$ and $\mathcal{E}_\sigma^{j'}$ have different colors. On the other hand, $\mathcal{D}_\sigma$ has a single marking since it is not decomposed. This multi-marking (or multi-coloration) of $\mathcal{A}_{\mathcal{E}_\sigma}$ is then easily propagated on $(\mathcal{A}_{\mathcal{E}_\sigma[1]}, \mathcal{A}_{\mathcal{D}_\sigma[1]})$, $(\mathcal{A}_{\mathcal{E}_\sigma[2]}, \mathcal{A}_{\mathcal{D}_\sigma[2]})$, ..., as follows : each color is propagated as we usually propagate the simple-marking of states. Note that, we use the synchronous

composition with multi-marking to compute $(\mathcal{A}_{\mathcal{E}_\sigma[k+1]}, \mathcal{A}_{\mathcal{D}_\sigma[k+1]})$ from $(\mathcal{A}_{\mathcal{E}_\sigma[k]}, \mathcal{A}_{\mathcal{D}_\sigma[k]})$. A formal definition of the synchronous composition $A_1||A_2$ of two multi-marked FSA A_1 and A_2, where $A_1||A_2$ is also a multi-marked FSA, is given in [de Queiroz *et al.*, 2005]. Then, for $k \geq 0$ and $j \leq p$, we have that $\mathcal{E}_\sigma^j[k]$ (resp. $\mathcal{D}_\sigma^j[k]$) is empty if and only if $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ (resp., $\mathcal{A}_{\mathcal{D}_\sigma[k]}$) has no state of color c^j. Indeed, the verification of $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS turns out to simply check if the states of $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ are colored.

3.9 Conclusion

We have developed a new framework, called multi-decision control, that is intended to be applicable to any existing decentralized architecture in order to generalize the latter. The basic principle of multi-decision control consists in using several decentralized supervisors Sup^j $(j \in J)$ running in parallel. Each decentralized supervisor Sup^j performs its local and global decisions according to a given eligible decentralized architecture D^j. The global decisions of all decentralized supervisors Sup^j are then combined according to a binary operator $\mathbf{D}$ in order to obtain the effective decision that is actually applied to the plant. We have then defined the notion of multi-decision supervisor $Sup = ((Sup^j)_{j \in J}, \mathbf{D})$ that has for goal to control the plant in order to achieve a given specification.

We have identified sufficient conditions that make a decentralized architecture eligible to be used in the multi-decision framework. Then, we have studied more thoroughly the cases where the final decision is obtained by combining the global decisions of all decentralized supervisors Sup^j $(j \in \{1, \ldots, p\})$ either disjunctively $(\mathbf{D} = \vee)$ or conjunctively $(\mathbf{D} = \wedge)$. In the case $\mathbf{D} = \vee$ (resp., $\mathbf{D} = \wedge$), for every $\sigma \in \Sigma_c$, we use a decomposition $(\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p)$ of $\mathcal{E}_\sigma$ (resp., $(\mathcal{D}_\sigma^1, \ldots, \mathcal{D}_\sigma^p)$ of $\mathcal{D}_\sigma$) such that each decentralized supervisor Sup^j carries out its decisions based on $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ (resp., $(\mathcal{E}_\sigma, \mathcal{D}_\sigma^j)$). Then, we have continued our study for $\mathbf{D} = \vee$ (i.e., decomposition of $\mathcal{E}_\sigma$), but we have explained how it can be adapted to the case $\mathbf{D} = \wedge$ (i.e., decomposition of $\mathcal{D}_\sigma$).

In the case $\mathbf{D} = \vee$, we have defined the notion of $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS, which is useful to characterize the class of achievable languages under the $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$ architecture. Moreover, we have shown that the class of languages achievable under the $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$ architecture englobe the class of languages achievable under the D^j architecture, for any $j \in \{1, \ldots, p\}$.

In order to show the applicability of our framework, we have illustrated the multi-decision framework in the special case where several (say p) inference-based Inf_{N_j}-supervisors $(j = 1 \ldots p)$ running in parallel and whose global decisions are fused disjunctively.

A difficult problem inherent to the multi-decision approach is the decomposition of infinite languages. We solve this problem by transforming the problem of decomposing an infinite regular language ($\mathcal{E}_\sigma$ or $\mathcal{D}_\sigma$) into a problem of decomposing the finite state set of an automaton accepting the regular language in question. We thus define the decidable notion of $\vee$(or $\wedge$)-($\mathrm{D}^1, \ldots, \mathrm{D}^p$)-COOBS w.r.t. some FSA.

As a future work, we will investigate more efficient methods for obtaining decidable versions of the multi-decision framework. We will investigate if it is possible to solve the decomposition problem by weakening Assumption **A1**. We also plan to study thoroughly the multi-decision framework when applied to other instances of decision-making, such as diagnosis and prognosis of DES. And as we have mentioned in Remark 3.4.1, other fusion operators than the disjunctive and conjunctive ones should be investigated. Maybe, we can find a fusion operator **D** which implies a decomposition of $\mathcal{E}_\sigma$ and $\mathcal{D}_\sigma$ in the same time.

CHAPITRE 4

Vérification de la coobservabilité dans le contexte du contrôle multi-décisionnel de SED

Article : H. Chakib and A. Khoumsi, *Verification of Coobservability in the Context of Multi-decision Supervisory Control of Discrete Event Systems*, Submitted to IEEE Transactions on Automatic Control. 2011.

Avant-propos

Auteurs et affiliation :

- Hicham Chakib : étudiant au doctorat, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.
  Ahmed Khoumsi : professeur, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Date de soumission : 4 mars 2011

Revue : IEEE Transactions on Automatic Control

Titre français : Vérification de la coobservabilité dans le contexte du contrôle multi-décisionnel de SED

Contribution au document : Dans la contribution précédente (chapitre 3), nous avons développé, entre autres, le contrôle multi-décisionnel par inférence, c'est-à-dire le contrôle multi-décisionnel avec des architectures par inférence fonctionnant en parallèle.

Dans la présente contribution, nous développons une méthode efficace (avec l'algorithme correspondant) pour vérifier si une spécification est coobservable selon le contrôle multi-décisionnel par inférence. Par efficace, nous voulons dire que la méthode développée n'est pas plus complexe (en temps de calcul et en mémoire utilisée) que des méthodes développées pour des architectures de contrôle moins générales.

Résumé français : Cet article étudie le contrôle multi-décisionnel défini dans [Chakib et Khoumsi, 2011b], où plusieurs superviseurs décentralisés fonctionnent en parallèle et coopèrent dans le but de réaliser une spécification donnée. Les auteurs dans [Chakib et Khoumsi, 2011b] ont étudié en détail le contrôle multi-décisionnel fondé sur l'inférence par ambiguïté, c.à.d., plusieurs superviseurs décentralisés qui utilisent la technique de

l'inférence par ambiguïté sont utilisés en parallèle. Généralement, la réalisabilité d'une spécification est reliée à une notion de coobservabilité. Dans cet article, on propose un algorithme qui vérifie si une spécification est coobservable dans le contexte du contrôle multi-décisionnel basé sur l'inférence par ambiguïté. On montre que notre algorithme a, dans le pire cas, une complexité de vérification de la coobservabilité multi-décisionnelle qui n'augmente pas avec le nombre de superviseurs décentralisés fonctionnant en parallèle. Plus précisément, dans le pire cas, on obtient le même ordre de complexité qu'avec la méthode de l'inférence par ambiguïté de [Kumar et Takai, 2007].

Note : À la suite des corrections demandées par les membres du jury, le contenu de cet article diffère de celui qui a été soumis.

Abstract : This article deals with the multi-decision decentralized supervisory control framework (or more briefly *multi-decision control*) defined in [Chakib et Khoumsi, 2011b], where a set of decentralized supervisors work in parallel and cooperate in order to achieve a given global specification by controlling a discrete event system. The authors of [Chakib et Khoumsi, 2011b] studied in detail the inference-based multi-decision control, that is, the case where inference-based decentralized supervisors with different ambiguity levels are used in parallel. As usual, the achievability of a specification is related to a notion of coobservability. In the present paper, we propose an algorithm that checks if a given specification is coobservable in the context of inference-based multi-decision control. We show that with our algorithm, the worst-case computational complexity for checking coobservability in the multi-decision framework is not increased by the number of decentralized supervisors in parallel. That is, in the worst case we obtain the same order of complexity as in the inference-based framework of [Kumar et Takai, 2007].

4.1 Introduction

This paper deals with decentralized control of DES [Cieslak *et al.*, 1988; Jiang et Kumar, 2000; Kumar et Takai, 2007; Lin et Wonham, 1988a, 1990; Overkamp et van Schuppen, 2001; Prosser *et al.*, 1997; Ricker et Rudie, 2000, 2003; Rudie et Willems, 1995; Rudie et Wonham, 1992; Takai et Ushio, 2005; Yoo et Lafortune, 2002a, 2004], where a set of local supervisors cooperate according to their observations in order to restrict the behaviour of a *plant* so that it respects a given global *specification*. Each local supervisor has a local observation of the plant and takes local decisions without communicating with the other

local supervisors. And the local decisions taken by the various supervisors are merged in order to issue an effective decision about enabling or disabling some controllable events.

Recently, the authors of [Chakib et Khoumsi, 2008a,b, 2011b] have proposed a multi-decision control framework whose basic principle consists in using several existing decentralized architectures working in parallel. For example, we can have a *conjunctive* and a *conditionally disjunctive* architectures [Rudie et Wonham, 1992; Yoo et Lafortune, 2002a, 2004] running in parallel. The global decisions of all these decentralized architectures are combined adequately in order to obtain an effective decision. The motivation of multi-decision framework is to obtain an architecture that generalizes all the decentralized architectures that compose it.

The authors of [Chakib et Khoumsi, 2011b] have first studied thoroughly the case where the supervisors in parallel are of a generic class of decentralized architectures qualified as eligible. Then, they have studied the more specific case of several decentralized inference-based supervisors whose global decisions are combined *disjunctively*. They defined a notion of coobservability for each developed architecture (the generic one, and the specific one), that characterizes specifications for which there exists a supervisor that can generate decisions which force the plant to conform to the specification. In the present paper, we study the coobservability of the specific case (i.e., inference-based supervisors running in parallel and combined disjunctively). Our purpose is to develop a method that checks if a specification is coobservable, and then to determine the computational complexity of the developed method.

The paper is organized as follows. Section 4.2 introduces the decentralized supervisory control of DES and presents pertinent definitions and results necessary for our framework. Sections 4.3 and 4.4 present the contributions of [Chakib et Khoumsi, 2011b] related to the specific case of inference-based supervisors combined disjunctively, which are useful for the understanding of the present article. Section 4.3 presents the motivation and principle of multi-decision control. Sections 4.4 to 4.6 study the case of inference-based architectures whose global decisions are combined *disjunctively*, but their contents can be straightforwardly adapted to the case of *conjunctive* combination, because the two cases are symmetrical with each other. Section 4.4 studies the case of inference-based architectures which are combined disjunctively, with an emphasis on a corresponding notion of coobservability. In section 4.5, we develop a method that checks if a given specification is coobservable for the architecture of Section 4.4. In the same section, we show that with our method, the worst-case computational complexity for checking coobservability is not increased by the number of decentralized supervisors in parallel. That is, in the worst case

we obtain the same order of complexity as with the inference-based framework of [Kumar et Takai, 2007]. In Section 4.6, we present an algorithm that implements (correctly) the method of Section 4.5. Section 4.7 illustrates our verification method by an example. The conclusion is presented in Section 4.8. And last but not least, the proofs are presented in an appendix.

4.2 Preliminaries on decentralized control of DES

We consider a plant modeled by a finite state automaton (FSA) $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is a finite set of states, Σ is a finite set of events, a partial function $\delta : Q \times \Sigma \rightarrow Q$ is the transition function, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the set of marked states. Let Σ^* be the set of all finite traces of elements of Σ, including the empty trace ε. The transition function δ can be generalized to $\delta : Q \times \Sigma^* \rightarrow Q$ in the usual way. The generated and marked languages of G, denoted by $\mathcal{L}$ and $\mathcal{L}_m$, respectively, are defined by $\mathcal{L} = \{s \in \Sigma^* | \, \delta(q_0, s) \textit{ is defined}\}$ and $\mathcal{L}_m = \{s \in \mathcal{L} | \, \delta(q_0, s) \in Q_m\}$. Note that, $\mathcal{L}_m \subseteq \mathcal{L}$.

We say that a trace $t \in \Sigma^*$ is a prefix of a trace $s \in \Sigma^*$, if there exists a trace $\lambda \in \Sigma^*$ such that $s = t\lambda$. We denote by $\overline{K}$ the set of all prefixes of traces of a language $K \subseteq \Sigma^*$. K is said prefix-closed if $K = \overline{K}$.

We consider a specification modeled by a language $K \subseteq \mathcal{L}_m$. The languages $\mathcal{L}$, $\mathcal{L}_m$ and K are defined over the same alphabet Σ, which is partitioned into Σ_c and Σ_{uc}, the sets of *controllable* and *uncontrollable* events, respectively. For every controllable event $\sigma \in \Sigma_c$, we denote by $\mathcal{E}_\sigma = \{s \in \overline{K} | \, s\sigma \in \overline{K}\}$ the set of traces of $\overline{K}$ after which σ is accepted by $\overline{K}$. Similarly, we denote by $\mathcal{D}_\sigma = \{s \in \overline{K} | \, s\sigma \in \mathcal{L} \backslash \overline{K}\}$ the set of traces of $\overline{K}$ after which σ is accepted by $\mathcal{L}$ and forbidden by $\overline{K}$.

A supervisory system (supervisor for short) *Sup* restricts the behavior of the plant so that it conforms to K, by taking effective enabling/disabling decisions that are applied to the plant. Let then $Sup(s, \sigma) \in \{\phi, 0, 1\}$ denote the effective enabling/disabling decision taken on an event $\sigma \in \Sigma$ after the execution of a trace $s \in \mathcal{L}$. $Sup(s, \sigma) = 1$ (resp. 0) means that σ is enabled (resp. disabled) after the execution of s. A decision $Sup(s, \sigma) = \phi$ means a "don't care" or "unsure" decision. A fundamental property that is respected by the effective decision $Sup(s, \sigma)$ of any supervisor is : $Sup(s, \sigma) = 1$, for any $s \in \mathcal{L}$ and $\sigma \in \Sigma_{uc}$. A supervisor *Sup* is said *admissible* w.r.t. K if for every $\sigma \in \Sigma_c$, $\forall s \in \mathcal{E}_\sigma \cup \mathcal{D}_\sigma$, $Sup(s, \sigma) \neq \phi$. The prefix-closed language $\mathcal{L}(Sup/G)$ generated by the plant under the

control of an admissible supervisor Sup is defined as follows :

$$\epsilon \in \mathcal{L}(Sup/G), \text{ and } [s \in \mathcal{L}(Sup/G) \wedge s\sigma \in \mathcal{L} \wedge Sup(s,\sigma) = 1] \Leftrightarrow [s\sigma \in \mathcal{L}(Sup/G)].$$

The corresponding marked language is defined by $\mathcal{L}_m(Sup/G) = \mathcal{L}(Sup/G) \cap \mathcal{L}_m$. A supervisor Sup is called *nonblocking* if $\overline{\mathcal{L}_m(Sup/G)} = \mathcal{L}(Sup/G)$.

Decentralized control [Kumar et Takai, 2007; Rudie et Wonham, 1992; Yoo et Lafortune, 2002a, 2004] is performed by n local supervisors $(Sup_i)_{1 \leq i \leq n}$. Each Sup_i has its own set of observable events $\Sigma_{o,i}$ and own set of controllable events $\Sigma_{c,i}$. The local supervisors together can then observe $\Sigma_o = \Sigma_{o,1} \cup \cdots \cup \Sigma_{o,n}$ and control $\Sigma_c = \Sigma_{c,1} \cup \cdots \cup \Sigma_{c,n}$. The sets of unobservable and uncontrollable events are denoted, respectively, by $\Sigma_{uo} = \Sigma \setminus \Sigma_o$ and $\Sigma_{uc} = \Sigma \setminus \Sigma_c$. We denote by $I = \{1, \cdots, n\}$ the indexing set of all supervisors. For any controllable event $\sigma \in \Sigma_c$, we define by $I_\sigma = \{i \in I | \sigma \in \Sigma_{c,i}\}$ the indexing set of local supervisors controlling σ, and $n_\sigma = |I_\sigma|$ denotes the cardinality of I_σ. We denote by $P_i : \Sigma^* \to \Sigma^*_{o,i}$ the natural projection that hides the events of $\Sigma \setminus \Sigma_{o,i}$ from any trace $s \in \Sigma^*$. The inverse projection is defined as $P_i^{-1}(s) = \{t \in \Sigma^* : P_i(t) = s\}$.

In decentralized control, each local supervisor $Sup_i, i \in I$, is a function $Sup_i : \Sigma^*_{o,i} \times \Sigma_c \to LD$, that associates a local decision $Sup_i(P_i(s), \sigma) \in LD$ to every controllable event σ and every observed trace $P_i(s)$. LD is an arbitrary finite set of local decisions including ϕ. The latter accounts for silent decision, this situation can occur when the local supervisor is not sure about the decision it can take following the observation of $P_i(s)$, we say that the local supervisor is *unsure*. The silent decision can also refer to a *don't care* situation. The effective decision that is applied to the plant is obtained by combining the local decisions of the n local supervisors $(Sup_i)_{i \in I}$ using a so-called *fusion operator* $\mathrm{D} : LD^n \to \{\phi, 0, 1\}$, we say that D is defined over LD. The system enclosing the tuple of local supervisors $(Sup_i)_{i \in I}$ and the coordinating module D is called a *decentralized supervisor*, and denoted by $((Sup_i)_{i \in I}, \mathrm{D})$. The effective decision $Sup(s, \sigma) \in \{\phi, 0, 1\}$ of a decentralized supervisor $Sup = ((Sup_i)_{i \in I}, \mathrm{D})$ is computed as follows :

$$Sup(s, \sigma) = \mathrm{D}((Sup_i(P_i(s), \sigma))_{\{i \in I_\sigma\}}) \tag{4.1}$$

Eq. (4.1) means that $Sup(s, \sigma)$ is computed by applying the operator D to all $Sup_i(P_i(s), \sigma)$.

Definition 4.2.1 *A* D*-supervisor is a decentralized supervisor defined by (4.1) for a given coordinating module* D *and any local supervisors* $(Sup_i)_{i \in I}$. *The set of all* D*-supervisors is called a* D*-architecture.*

For example, we have the conjunctive architecture ($\wedge$-architecture) and the disjunctive architecture ($\vee$-architecture) of [Rudie et Wonham, 1992; Yoo et Lafortune, 2002a], the conditional conjunctive architecture (COND-$\wedge$-architecture) and conditional disjunctive architecture (COND-$\vee$-architecture) of [Yoo et Lafortune, 2004], and the N-inference architecture (Inf_N-architecture) of [Kumar et Takai, 2007].

4.3 Multi-decision decentralized supervisory control : several architectures running in parallel

In the multi-decision control, we have several (say p) decentralized supervisors $(Sup^j)_{j\in\{1,\cdots,p\}}$ running in parallel and whose global decisions are fused into an *effective decision* using an operator $\mathbf{D}$. Hereafter, we denote by $J = \{1, \cdots, p\}$ the indexing set of the decentralized supervisors running in parallel. Each decentralized supervisor Sup^j achieves its control according to a given decentralized architecture as shown in Section 4.2 and (4.1), but with a superscript j. That is, for every $j \in J$, $Sup^j = ((Sup_i^j)_{i\in I}, \mathrm{D}^j)$ contains n local supervisors $(Sup_i^j)_{i\in I}$ and a coordinating module D^j. The global decision issued by Sup^j, following the execution of a trace $s \in \mathcal{L}$, is computed by the following equation, which corresponds to (4.1) with a superscript j added to Sup, Sup_i and D :

$$Sup^j(s,\sigma) = \mathrm{D}^j((Sup_i^j(P_i(s),\sigma))_{\{i\in I_\sigma\}}) \tag{4.2}$$

The global decisions of all the decentralized supervisors $(Sup^j)_{j\in J}$ are combined using an operator $\mathbf{D}$ in order to obtain an adequate effective decision. $\mathbf{D}$ is defined as a function $\mathbf{D} : \{\phi, 0, 1\}^p \rightarrow \{\phi, 0, 1\}$. The system enclosing the tuple of local supervisors $(Sup_i^j)_{i\in I, j\in J}$, the fusion operators $(\mathrm{D}^j)_{j\in J}$ and the global fusion operator $\mathbf{D}$ is called a *multi-decision supervisor*, and denoted by $((Sup^j)_{j\in J}, \mathbf{D}) = (((Sup_i^j)_{i\in I}, \mathrm{D}^j)_{j\in J}, \mathbf{D})$. The effective decision of a multi-decision supervisor $Sup = ((Sup^j)_{j\in J}, \mathbf{D})$, $Sup(s,\sigma) \in \{\phi, 0, 1\}$, is computed as follows :

$$Sup(s,\sigma) = \mathbf{D}((Sup^j(s,\sigma))_{\{j\in J\}}) \tag{4.3}$$

Eq. (4.3) means that the effective decision $Sup(s,\sigma)$ is computed by applying the operator $\mathbf{D}$ to all the global decisions $Sup^j(s,\sigma)$.

Definition 4.3.1 *A $\boldsymbol{D}$-$(\mathrm{D}^1, \cdots, \mathrm{D}^p)$-supervisor is a multi-decision supervisor defined by (4.2) and (4.3) for given $\boldsymbol{D}$ and $(Sup^j)_{j\in J}$. The set of all $\boldsymbol{D}$-$(\mathrm{D}^1, \cdots, \mathrm{D}^p)$-supervisors is called a $\boldsymbol{D}$-$(\mathrm{D}^1, \cdots, \mathrm{D}^p)$-architecture.*

When all the D^j-supervisors Sup^j (for $j \in J$) correspond to the same decentralized architecture ψ, we obtain a so-called $\mathbf{D}$-ψ^p architecture, while Sup is called $\mathbf{D}$-ψ^p supervisor.

Let us for example take $p = 2$, $D^1 = \vee$, $D^2 = \text{COND-}\wedge$, and $\mathbf{D} = \wedge$. This means that we have a disjunctive (D^1) and a conditionally-conjunctive (D^2) architectures running in parallel and whose global decisions are fused conjunctively ($\mathbf{D}$) for obtaining the effective decision. And thus we have a $\wedge$-($\vee$,COND-$\wedge$) architecture and supervisor.

At the present state of our study, we have considered two cases of $\mathbf{D}$ when $p > 1$: $\mathbf{D}$ is disjunctive ($\vee$) or conjunctive ($\wedge$). In the case $\mathbf{D} = \vee$, we are in the presence of a $\vee$-($D^1, \cdots, D^p$) architecture. The corresponding $\vee$-($D^1, \cdots, D^p$)-supervisor consists of a set of p D^j-supervisors Sup^j (for $j \in J$) running in parallel and whose global decisions are fused *disjunctively*. By disjunctive, we mean :

$$Sup(s,\sigma) = \begin{cases} 0 & \text{if, } \forall j \in J, Sup^j(s,\sigma) = 0, \\ 1 & \text{if, } \exists j \in J, Sup^j(s,\sigma) = 1, \\ \phi & \text{otherwise.} \end{cases} \tag{4.4}$$

This case is based in considering, for each $\sigma \in \Sigma_c$, a decomposition $(\mathcal{E}_\sigma^1, \cdots, \mathcal{E}_\sigma^p)$ of $\mathcal{E}_\sigma$ such that $\mathcal{E}_\sigma = \mathcal{E}_\sigma^1 \cup \cdots \cup \mathcal{E}_\sigma^p$. We use the word *decomposition* instead of *partition* because the various $\mathcal{E}_\sigma^j$ are not necessarily disjoint with each other.

The case $\mathbf{D} = \wedge$ can be obtained by adapting straightforwardly the *disjunctive* case, because the two cases are symmetrical with each other (see [Chakib et Khoumsi, 2011b] for more details). The following sections 4.4-4.6 study the case of inference-based architectures whose global decision are combined *disjunctively*.

4.4 Inference-based multi-decision architecture

In this section, we study the inference-based multi-decision architecture. An inference-based multi-decision supervisor $Sup = ((Sup^j)_{j \in J}, \vee)$ consists on p inference-based decentralized supervisors $(Sup^j)_{j \in J}$ running in parallel and whose global decisions are fused disjunctively ($\mathbf{D} = \vee$) for obtaining an effective decision that is actually applied to the plant. The inference-based architecture of [Kumar et Takai, 2007] is summarized in the next subsection.

4.4.1 Inference-based architecture

Inference-based supervisor

Each inference-based decentralized supervisor *Sup* consists of n *inference-based* local supervisors $(Sup_i)_{i\in I}$, and its decisions are computed using the results of [Kumar et Takai, 2007] as follows. After the execution of $s \in \overline{K}$, each Sup_i has observed $P_i(s)$ and issues a local decision $c_i(P_i(s), \sigma)$ for each $\sigma \in \Sigma_c$. An ambiguity level $n_i(P_i(s), \sigma)$ is associated to each local decision. Formally :

$$Sup_i(P_i(s), \sigma) = (c_i(P_i(s), \sigma), n_i(P_i(s), \sigma)). \tag{4.5}$$

The generic global decision of each *Sup* was defined by (4.2). In the present case of inference-based supervisor *Sup*, its global decision is computed as follows [Kumar et Takai, 2007] : $\forall \sigma \in \Sigma_c, \forall s \in \overline{K}$,

$$Sup(s, \sigma) = \begin{cases} 1, & \text{if } \forall i \in I_\sigma; [n(s, \sigma) = n_i(P_i(s), \sigma) \Rightarrow c_i(P_i(s), \sigma) = 1], \\ 0, & \text{if } \forall i \in I_\sigma; [n(s, \sigma) = n_i(P_i(s), \sigma) \Rightarrow c_i(P_i(s), \sigma) = 0], \\ \phi, & otherwise. \end{cases} \tag{4.6}$$

Where $n(s, \sigma)$ denotes the minimal ambiguity level of local decisions of *Sup*, i.e.,

$$n(s, \sigma) = \min_{i \in I_\sigma} n_i(P_i(s), \sigma) \tag{4.7}$$

Definition 4.4.1 *Given an integer $N \geq 0$, an inference-based decentralized supervisor (i.e., defined by (4.5)-(4.7)) is said* N-inferring *(w.r.t. K), and denoted Inf_N-supervisor, if for any $\sigma \in \Sigma_c$, $\min\{n^d_\sigma, n^e_\sigma\} \leq N$, where $n^d_\sigma = \max_{\{s \in \overline{K} | s\sigma \in \mathcal{L} \wedge Sup(s,\sigma)=0\}} n(s, \sigma)$ and $n^e_\sigma = \max_{\{s \in \overline{K} | s\sigma \in \mathcal{L} \wedge Sup(s,\sigma)=1\}} n(s, \sigma)$.*

Note that N is the maximal ambiguity level that is used in a Inf_N-supervisor.

Computing the local decisions $c_i(P_i(s), \sigma)$ and $n_i(P_i(s), \sigma)$

We proceed as in [Kumar et Takai, 2007], but for a convenient presentation of the inference-based multi-decision (in the following Section 4.4.2), we use languages (E, D) such that $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$, instead of $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$. We define inductively a monotonically decreasing sequence of language pairs $(E[k], D[k])$ as follows.

Basis : $E[0] = E$ and $D[0] = D$,

Inductive step ($k \geq 1$) : $E[k+1] = E[k] \cap \bigcap_{i \in I_\sigma} P_i^{-1} P_i(D[k])$ and $D[k+1] = D[k] \cap \bigcap_{i \in I_\sigma} P_i^{-1} P_i(E[k])$.

The decisions of every *Sup* are defined by (4.5)-(4.7) from $(c_i(P_i(s), \sigma), n_i(P_i(s), \sigma))_{i \in I_\sigma}$. Now we present a method of [Kumar et Takai, 2007] to compute $c_i(P_i(s), \sigma)$ and $n_i(P_i(s), \sigma)$, for $i \in I_\sigma$. More precisely, using the sequence $(E[k], D[k])$ of language pairs, every local supervisor Sup_i ($i \in I_\sigma$) computes, for every $s \in \overline{K}$ and $\sigma \in \Sigma_c$, $n_i(P_i(s), \sigma)$ and $c_i(P_i(s), \sigma)$ as follows.

$$\begin{aligned} n_i^e(P_i(s), \sigma) &= \min\{k \in \mathbb{Z}^+ \mid P_i^{-1} P_i(s) \cap D[k] = \emptyset\}, \\ n_i^d(P_i(s), \sigma) &= \min\{k \in \mathbb{Z}^+ \mid P_i^{-1} P_i(s) \cap E[k] = \emptyset\}. \end{aligned} \tag{4.8}$$

A local decision is issued by comparing the two ambiguity levels, $n_i^e(P_i(s), \sigma)$ and $n_i^d(P_i(s), \sigma)$, giving preference to the smallest one. This is formalized as follows. For every $\sigma \in \Sigma_c$, the decision and ambiguity level of a local supervisor Sup_i following an observation $P_i(s)$, i.e., $Sup_i(P_i(s), \sigma) = (c_i(P_i(s), \sigma), n_i(P_i(s), \sigma))$, is computed as follows :

$$c_i(P_i(s), \sigma) = \begin{cases} 0, & \text{if } n_i^d(P_i(s), \sigma) < n_i^e(P_i(s), \sigma), \\ 1, & \text{if } n_i^e(P_i(s), \sigma) < n_i^d(P_i(s), \sigma), \\ \phi, & \text{otherwise}, \end{cases} \tag{4.9}$$

and

$$n_i(P_i(s), \sigma) = \min\{n_i^d(P_i(s), \sigma), n_i^e(P_i(s), \sigma)\}. \tag{4.10}$$

We have not explained in detail the (4.5)-(4.10) because they have been taken from the inference-based framework of [Kumar et Takai, 2007].

4.4.2 Parallel inference-based architectures running in parallel

Recall that a $\vee$-$(\mathrm{D}^1, \cdots, \mathrm{D}^p)$ architecture consists of p D^j-architectures running in parallel and whose global decisions are fused disjunctively. When the p Inf_{N_j}-supervisors are inference-based with respective ambiguities $N_1, \cdots, N_p$, we are in the presence of an inference-based multi-decision architecture, denoted $\vee$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-architecture. Any corresponding supervisor Sup=$((Sup)_{j \in J}, \vee)$ consists on p Inf_{N_j}-supervisors $(Sup^j)_{j \in J}$. and is denoted $\vee$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-supervisor. For each Inf_{N_j}-supervisor, the local decisions are computed by using (4.5) but with a superscript j, i.e., $Sup_i^j(P_i(s), \sigma)$=$(c_i^j(P_i(s), \sigma), n_i^j(P_i(s), \sigma))$, and the global decision $Sup^j(s, \sigma)$ is computed by using (4.6) but with a

superscript j. The effective decision $Sup(s,\sigma)$ of the $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-supervisor is computed by combining disjunctively the global decisions $(Sup^j(s,\sigma))_{j\in J}$ using (4.4).

As already mentioned in Section 4.3, we will use a decomposition $\{\mathcal{E}_\sigma^1,\cdots,\mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ for every $\sigma\in\Sigma_c$. This decomposition is a practical rule for computing the inference-based local decisions as described in (4.8)-(4.10) for every Inf_{N_j}-supervisor, by taking $E=\mathcal{E}_\sigma^j$ and $D=\mathcal{D}_\sigma$. Hence, the sequence of language pairs $(\mathcal{E}_\sigma^j[k],\mathcal{D}_\sigma^j[k])$ are computed as follows. For $k=0$, $\mathcal{E}_\sigma^j[0]=\mathcal{E}_\sigma^j$ and $\mathcal{D}_\sigma^j[0]=\mathcal{D}_\sigma$. For $k\geq 1$, $\mathcal{E}_\sigma^j[k+1]=\mathcal{E}_\sigma^j[k]\cap\bigcap_{\iota\in I_\sigma}P_\iota^{-1}P_\iota(\mathcal{D}_\sigma^j[k])$ and $\mathcal{D}_\sigma^j[k+1]=\mathcal{D}_\sigma^j[k]\cap\bigcap_{\iota\in I_\sigma}P_\iota^{-1}P_\iota(\mathcal{E}_\sigma^j[k])$. For simplicity, j is omitted when $p=1$ (i.e., no decomposition).

4.4.3 Existence of solutions

In this subsection, we introduce and study the notion of $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-COOBS in order to characterize the class of languages achievable under the control of a nonblocking and admissible $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-supervisor.

For a convenient presentation of the inference-based multi-decision (in Section 4.4.2), we do not use exactly the definition of [Kumar et Takai, 2007]. We rather generalize it by the use of subsets $E\subseteq\mathcal{E}_\sigma$ and $D\subseteq\mathcal{D}_\sigma$ instead of $\mathcal{E}_\sigma$ and $\mathcal{D}_\sigma$, respectively.

Definition 4.4.2 *Consider $K\subseteq\mathcal{L}_m$, $\sigma\in\Sigma_c$, $E\subseteq\mathcal{E}_\sigma$, $D\subseteq\mathcal{D}_\sigma$ and an integer $N\geq 0$. (E,D) is said Inf_N-*coobservable *(or Inf_N-*COOBS*) if $E[N+1]=\emptyset$ or $D[N+1]=\emptyset$. K is said Inf_N-*COOBS *if, $\forall\sigma\in\Sigma_c$, $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is Inf_N-*COOBS.

Based on the notion of Inf_N-COOBS of Definition 4.4.2, let us define the notion of $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-COOBS.

Definition 4.4.3 *Consider $K\subseteq\mathcal{L}_m$, and $\sigma\in\Sigma_c$. $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is said $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-*COOBS *w.r.t. a decomposition $(\mathcal{E}_\sigma^1,\cdots,\mathcal{E}_\sigma^p)$ of $\mathcal{E}_\sigma$ if, $\forall j\in J$, $(\mathcal{E}_\sigma^j,\mathcal{D}_\sigma)$ is Inf_{N_j}-*COOBS. *$(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is said $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-*COOBS *if there exists a decomposition $(\mathcal{E}_\sigma^1,\cdots,\mathcal{E}_\sigma^p)$ of $\mathcal{E}_\sigma$ such that $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-*COOBS *w.r.t. $(\mathcal{E}_\sigma^1,\cdots,\mathcal{E}_\sigma^p)$.*

Definition 4.4.4 *Given an integer $N\geq 0$, $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is said $\vee$-$Inf_{\leq N}^{\geq 1}$-*COOBS *if it is $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-*COOBS *for some $p\geq 1$ and non-negative integers $N_1,\cdots,N_p\leq N$. $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is said $\vee$-$Inf_{\geq 0}^{\geq 1}$-*COOBS *if it is $\vee$-$Inf_{\leq M}^{\geq 1}$-*COOBS *for some unspecified integer $M\geq 0$.*

Note that by Definition 4.4.4, $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS implies $\vee$-$Inf_{\geq 0}^{\geq 1}$-COOBS.

It has been shown in [Chakib et Khoumsi, 2011b] that there exists a $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-supervisor forcing the plant to conform to K if and only if K is controllable and $\mathcal{L}_m$-closed, and $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-COOBS for every $\sigma \in \Sigma_c$.

$\mathcal{L}_m$-closure and controllability are classical notions in supervisory control of DES that can be checked in the usual way, since they do not depend on the control architecture. It remains therefore to determine whether $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-COOBS for every $\sigma \in \Sigma_c$.

According to Def. 4.4.3, we have to answer the following question for every $\sigma \in \Sigma_c$:

Question 1 : Does there exist a decomposition $(\mathcal{E}_\sigma^1,\cdots,\mathcal{E}_\sigma^p)$ of $\mathcal{E}_\sigma$ and some $N_1,\cdots,N_p \leq N$ such that, $\forall j \in J$, $(\mathcal{E}_\sigma^j,\mathcal{D}_\sigma)$ is Inf_{N_j}-COOBS, i.e., $\mathcal{E}_\sigma^j[N_j+1]=\emptyset$ or $\mathcal{D}_\sigma^j[N_j+1]=\emptyset$?

What makes Question 1 difficult is that $\mathcal{E}_\sigma$ is in general infinite and decomposing infinite languages is known to be a challenging problem. We propose here a solution that transforms the problem of decomposing an infinite regular language into a problem of decomposing a *finite* set of states in a FSA. Our solution for decomposing $\mathcal{E}_\sigma$ is based on the use of a FSA $\mathcal{A}_{\mathcal{E}_\sigma}$ accepting $\mathcal{E}_\sigma$, that is, every trace $s \in \mathcal{E}_\sigma$ leads to a marked state of $\mathcal{A}_{\mathcal{E}_\sigma}$. We consider uniquely the decompositions satisfying the following assumption :

A1 : Given a FSA $\mathcal{A}_{\mathcal{E}_\sigma}$ accepting $\mathcal{E}_\sigma$, the only eligible decompositions $(\mathcal{E}_\sigma^1,\cdots,\mathcal{E}_\sigma^p)$ of $\mathcal{E}_\sigma$ are such that every $\mathcal{E}_\sigma^j$ consists of (all and only) the traces leading to one or several marked states of $\mathcal{A}_{\mathcal{E}_\sigma}$. In other words, every $\mathcal{E}_\sigma^j$ corresponds to a subset of the set of marked states of $\mathcal{A}_{\mathcal{E}_\sigma}$.

With Assumption **A1**, and assuming regular languages, the number of eligible decompositions becomes *finite*. Indeed, with **A1**, we have transformed the problem of decomposing an infinite regular language into a problem of decomposing the finite state set of a FSA.

We have the following definition to specify that $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-COOBS for decompositions satisfying **A1** w.r.t. a specific FSA $\mathcal{A}_{\mathcal{E}_\sigma}$ accepting $\mathcal{E}_\sigma$.

Definition 4.4.5 *Consider $\sigma \in \Sigma_c$ and a FSA $\mathcal{A}_{\mathcal{E}_\sigma}$ accepting $\mathcal{E}_\sigma$. $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is said $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-COOBS w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$, if there exists a decomposition $\{\mathcal{E}_\sigma^1,\cdots,\mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ satisfying **A1** w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$ such that, $\forall j \in J$, $(\mathcal{E}_\sigma^j,\mathcal{D}_\sigma)$ is Inf_{N_j}-COOBS (see Def. 4.4.2).*

The Definition is generalized as well to $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS (and $\vee$-$Inf_{\geq 0}^{\geq 1}$-COOBS) w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$.

4.5 Checking coobservability and constructing decomposition of $\mathcal{E}_\sigma$

We have seen that the notion of coobservability is relevant to determine the existence of supervisors forcing the plant to conform to a specification. We have seen three versions of coobservability in Defs. 4.4.3 and 4.4.4 :

$\vee$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-COOBS for given $p \geq 1$ and $N_1, \cdots, N_p \geq 0$;

$\vee$-$Inf^{\geq 1}_{\geq 0}$-COOBS meaning $\vee$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-COOBS for some unspecified $p \geq 1$ and $N_1, \cdots, N_p \geq 0$;

$\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS meaning $\vee$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-COOBS for some unspecified $p \geq 1$ and $N_1, \cdots, N_p \leq N$.

Contrary to $\vee$-$Inf^{\geq 1}_{\geq 0}$-COOBS, the parameters p and $N_1, \cdots, N_p$ must be specified in $\vee$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-COOBS, which makes $\vee$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-COOBS more restrictive than $\vee$-$Inf^{\geq 1}_{\geq 0}$-COOBS. But the problem with $\vee$-$Inf^{\geq 1}_{\geq 0}$-COOBS is that $N_1, \cdots, N_p$ are not bounded, which makes it undecidable in general. That is why we have defined the (decidable) $\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS by limiting the possible values $N_1, \cdots, N_p$ to be checked. In this section, we are indeed interested by $\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS. More precisely, we propose au automata based method that checks for every $\sigma \in \Sigma_c$, if $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS w.r.t a given FSA $\mathcal{A}_{\mathcal{E}_\sigma}$ (accepting $\mathcal{E}_\sigma$). (In the sequel, the notion of $\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS is implicitly w.r.t. a given FSA $\mathcal{A}_{\mathcal{E}_\sigma}$.) And when the method determines that $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS, it computes a corresponding decomposition (in fact, a partition) of $\mathcal{E}_\sigma$. For that purpose, we first need to compute the pairs $(\mathcal{E}_\sigma[k], \mathcal{D}_\sigma[k])_{k \geq 0}$, or more precisely automata accepting them. This is the subject of Subsection 4.5.1.

4.5.1 Computing automata accepting $\mathcal{E}_\sigma[k]$ and $\mathcal{D}_\sigma[k]$, for $k \geq 0$

Let $\mathcal{A}_K = (R, \Sigma, \xi, r_0, R_m)$ be a finite trim acceptor of $K \subseteq \mathcal{L}_m$, which means that $\mathcal{L}_m(\mathcal{A}_K) = K$ and $\mathcal{L}(\mathcal{A}_K) = \overline{K}$. For every $\sigma \in \Sigma_c$, an automaton accepting $\mathcal{E}_\sigma$, noted $\mathcal{A}_{\mathcal{E}_\sigma} = (X, \Sigma, \alpha, x_0, X_m)$, is obtained from $\mathcal{A}_K$ by replacing the marked state set R_m with $X_m = \{x \in R \mid \xi(x, \sigma) \text{ is defined}\}$ and then removing the states from which no state of X_m is reachable. Complexity of computing $\mathcal{A}_{\mathcal{E}_\sigma}$ is $O(|R| \cdot |\Sigma_c|)$.

Recall that the plant is modeled by an automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$. Computing an automaton $\mathcal{A}_{\mathcal{D}_\sigma} = (Y, \Sigma, \beta, y_0, Y_m)$ accepting $\mathcal{D}_\sigma$ requires the construction of the synchronous composition of G and $\mathcal{A}_K$, noted $G||\mathcal{A}_K = (H, \Sigma, \gamma, h_0, H_m)$. The automaton $\mathcal{A}_{\mathcal{D}_\sigma}$ is obtained from $G||\mathcal{A}_K$ by replacing the marked states set H_m with $Y_m = \{(q, r) \in H \mid \delta(q, \sigma) \text{ is defined},$

$\xi(r,\sigma)$ is not defined}, and then removing the states from which no state of Y_m is reachable. Complexity of computing $\mathcal{A}_{\mathcal{D}_\sigma}$ is $O(|Q|\cdot|R|\cdot|\Sigma|)$.

Let $\mathcal{A}_{\mathcal{E}_\sigma[k]} = (X[k],\Sigma,\alpha[k],x_0[k],X_m[k])$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]} = (Y[k],\Sigma,\beta[k],y_0[k],Y_m[k])$ be the automata accepting $\mathcal{E}_\sigma[k]$ and $\mathcal{D}_\sigma[k]$, respectively, for $k \geq 0$. Recall that $\mathcal{E}_\sigma[0] = \mathcal{E}_\sigma$ and $\mathcal{D}_\sigma[0] = \mathcal{D}_\sigma$. We have just shown how to compute $\mathcal{A}_{\mathcal{E}_\sigma[0]} = \mathcal{A}_{\mathcal{E}_\sigma}$ and $\mathcal{A}_{\mathcal{D}_\sigma[0]} = \mathcal{A}_{\mathcal{D}_\sigma}$. Let us now show how to compute inductively $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$.

Since $\mathcal{E}_\sigma[k+1] = \mathcal{E}_\sigma[k] \cap \bigcap_{i\in I_\sigma} P_i^{-1}P_i(\mathcal{D}_\sigma[k])$, $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ is computed as follows. For each $i \in I_\sigma$, we compute the projection $P_i(\mathcal{A}_{\mathcal{D}_\sigma[k]})$ as indicated in [Barrett et Couch, 1979; Hopcroft et Ullman, 1979]. Note that the result of projection may be nondeterministic, we do not determinize it for avoiding computational complexity. Then, the inverse projection of $P_i(\mathcal{A}_{\mathcal{D}_\sigma[k]})$, i.e., $P_i^{-1}P_i(\mathcal{A}_{\mathcal{D}_\sigma[k]})$ is obtained by simply adding self-loop transitions labeled by events of $\Sigma\backslash\Sigma_{o,i}$ at each state of $P_i(\mathcal{A}_{\mathcal{D}_\sigma[k]})$. After that, a synchronous composition is applied between all $P_i^{-1}P_i(\mathcal{A}_{\mathcal{D}_\sigma[k]})$, $i \in I_\sigma$, and $\mathcal{A}_{\mathcal{E}_\sigma[k]}$. The set of marked states of $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ is obtained in the usual way. $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ is computed with a similar approach. We denote by $X_m[k]$ and $Y_m[k]$ the set of marked states of $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$, respectively. Emptiness of $\mathcal{E}_\sigma[k]$ (resp. $\mathcal{D}_\sigma[k]$) is equivalent to emptiness of $X_m[k]$ (resp. $Y_m[k]$).

We have the following lemmas that evaluate the orders of $|X[k+1]|$, $|Y[k+1]|$, $|\alpha[k+1]|$ and $|\beta[k+1]|$ from $|X[k]|$ and $|Y[k]|$, and evaluate the complexity for computing $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$. That is, we consider one step, from k to $k+1$, for $k > 0$.

Lemma 4.5.1 *$|X[k+1]|$ is in $O(|X[k]|\cdot|Y[k]|^{n_\sigma})$ and $|\alpha[k+1]|$ is in $O(|X[k]|^2\cdot|Y[k]|^{2n_\sigma}\cdot|\Sigma|)$*
$= O(|X[k+1]|^2\cdot|\Sigma|)$. Symmetrically, $|Y[k+1]|$ is in $O(|Y[k]|\cdot|X[k]|^{n_\sigma})$ and $|\beta[k+1]|$ is in $O(|Y[k]|^2\cdot|X[k]|^{2n_\sigma}\cdot|\Sigma|) = O(|Y[k+1]|^2\cdot|\Sigma|)$.

Lemma 4.5.2 *Computing $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ is performed in $O(|\alpha[k+1]|) = O(|X[k]|^2\cdot|Y[k]|^{2n_\sigma}\cdot|\Sigma|)$. Symmetrically, computing $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ is performed in $O(|\beta[k+1]|) = O(|Y[k]|^2\cdot|X[k]|^{2n_\sigma}\cdot|\Sigma|)$.*

For every $k \geq 1$, each state $u \in X_m[k]$ can be expressed in the form $(u_1,\cdots,u_{n_\sigma},u_{n_\sigma+1})$, where $u_i \subseteq Y[k-1]$ and $u_i \cap Y_m[k-1] \neq \emptyset$, for $i \in \mathbb{I}_\sigma = \{1,\cdots,n_\sigma\}$, and $u_{n_\sigma+1} \in X_m[k-1]$. Symmetrically, each state $v \in Y_m[k]$ can be presented in the form $(v_1,\cdots,v_{n_\sigma},v_{n_\sigma+1})$, where $v_i \subseteq X[k-1]$ and $v_i \cap X_m[k-1] \neq \emptyset$, for $i \in \mathbb{I}_\sigma$, and $v_{n_\sigma+1} \in Y_m[k-1]$.

Having seen how to compute $(\mathcal{A}_{\mathcal{E}_\sigma[k]},\mathcal{A}_{\mathcal{D}_\sigma[k]})_{k\geq 0}$, we can now show how to check if $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$. In Subsection 4.5.2, we consider the particular case where

$p = 1$ (no decomposition of $\mathcal{E}_\sigma$) and $N = 0$ (no inference). In Subsection 4.5.4, we consider the general case where $p \geq 1$ and $N \geq 0$. For the following, recall that X_m and Y_m are the marked states of $\mathcal{A}_{\mathcal{E}_\sigma}$ and $\mathcal{A}_{\mathcal{D}_\sigma}$, respectively ; and $X_m[k]$ and $Y_m[k]$ are the marked states of $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$, respectively.

4.5.2 Checking if $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is Inf_0-Coobs

Let us first check the basic case where we have coobservability without decomposing $\mathcal{E}_\sigma$ or $\mathcal{D}_\sigma$ (no multi-decision) and without inference, i.e., $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is Inf_0-COOBS. That is, the aim is to check whether $\mathcal{E}_\sigma[1]$ or $\mathcal{D}_\sigma[1]$ is empty, i.e., $X_m[1]$ or $Y_m[1]$ is empty. The following proposition is deduced from Lemma 4.5.2 by taking $k = 0$.

Proposition 4.5.1 *Checking if $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is Inf_0-COOBS (i.e., if $\mathcal{E}_\sigma[1] = \emptyset$ or $\mathcal{D}_\sigma[1] = \emptyset$) is performed in $O((|X|^{2n_\sigma} \cdot |Y|^2 + |Y|^{2n_\sigma}|X|^2) \cdot |\Sigma|)$.*

For checking if $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$, we first need to define the notion of multi-marking.

4.5.3 Multi-marking in $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$

When $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not Inf_0-COOBS, the aim is to check whether there exists a decomposition $\{\mathcal{E}_\sigma^1, \cdots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ respecting **A1** w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$ such that $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-COOBS for some $p \geq 1$ and some $N_1, \cdots, N_p \leq N$. We use the notion of *multi-marking* of [de Queiroz *et al.*, 2005] to construct partitions $(\mathcal{E}_\sigma^1, \cdots, \mathcal{E}_\sigma^p)$ of $\mathcal{E}_\sigma$ and determine whether $\mathcal{E}_\sigma^j[k]$ or $\mathcal{D}_\sigma^j[k]$ is empty, for $k \geq 0$ and $j \leq p$. Let $\{X_m^1, \cdots, X_m^p\}$ be a decomposition of X_m, a state $v \in Y_m[k]$ (or $v \in X_m[k]$) is said X_m^j-*marked* if v remains marked when only the states of X_m^j (instead of X_m) are marked in $\mathcal{A}_{\mathcal{E}_\sigma}$. This multi-marking is determined inductively in $X_m[k]$ and $Y_m[k]$ (marked states of $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$, respectively), $k \geq 1$, as follows :

Consider a state $u = (u_1, \cdots, u_{n_\sigma}, u_{n_\sigma+1}) \in X_m[1]$ and thus, $u_{n_\sigma+1} \in X_m$. The state u is X_m^j-marked if $u_{n_\sigma+1} \in X_m^j$.

Consider a state $v = (v_1, \cdots, v_{n_\sigma}, v_{n_\sigma+1}) \in Y_m[1]$ and thus, $\forall i \in \mathbb{I}_\sigma$, $v_i \cap X_m \neq \emptyset$. The state v is X_m^j-marked if, $\forall i \in \mathbb{I}_\sigma$, $v_i \cap X_m^j \neq \emptyset$.

Consider a state $v = (v_1, \cdots, v_{n_\sigma}, v_{n_\sigma+1}) \in Y_m[k]$, $k \geq 2$, and thus, $\forall i \in \mathbb{I}_\sigma$, $v_i \cap X_m[k-1] \neq \emptyset$ and $v_{n_\sigma+1} \in Y_m[k-1]$. The state v is X_m^j-marked if, $\forall i \in \mathbb{I}_\sigma$, v_i contains a X_m^j-marked state of $X_m[k-1]$ and $v_{n_\sigma+1}$ is a X_m^j-marked state of $Y_m[k-1]$.

Consider a state $u = (u_1, \cdots, u_{n_\sigma}, u_{n_\sigma+1}) \in X_m[k]$, $k \geq 2$, and thus, $\forall i \in \mathbb{I}_\sigma$, $u_i \cap Y_m[k-1] \neq \emptyset$ and $u_{n_\sigma+1} \in X_m[k-1]$. The state u is X_m^j-marked if, $\forall i \in \mathbb{I}_\sigma$, u_i contains a X_m^j-marked state of $Y_m[k-1]$ and $u_{n_\sigma+1}$ is a X_m^j-marked state of $X_m[k-1]$.

Note that a state may be at the same time X_m^i-marked and X_m^j-marked for $i \neq j$. The multi-marking can be interpreted as follows : A state in $X_m[k]$ (resp. $Y_m[k]$) is X_m^j-marked if and only if it is reached by trace(s) of $\mathcal{E}_\sigma^j[k]$ (resp., $\mathcal{D}_\sigma^j[k]$). Therefore, $\mathcal{E}_\sigma^j[k]$ (resp. $\mathcal{D}_\sigma^j[k]$) is empty if and only if $X_m[k]$ (resp., $Y_m[k]$) contains no X_m^j-marked state.

Given $\mathcal{X} \subseteq X_m$, we denote by $X_m[k]|_{\mathcal{X}} \subseteq X_m[k]$ and $Y_m[k]|_{\mathcal{X}} \subseteq Y_m[k]$ the $\mathcal{X}$-marked states of $X_m[k]$ and $Y_m[k]$, respectively. A decomposition $\{\mathcal{E}_\sigma^1, \cdots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ is formally related to its corresponding decomposition $\{X_m^1, \cdots, X_m^p\}$ of X_m as follows :

$$\mathcal{E}_\sigma^j = \{s \in \overline{K} \,|\, \alpha(x_0, s) \in X_m^j\}. \tag{4.11}$$

Note that, for each decomposition $\{X_m^1, \cdots, X_m^p\}$ of X_m, the decomposition $D = \{\mathcal{E}_\sigma^1, \cdots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$, for which every $\mathcal{E}_\sigma^j \in D$ is given by (4.11), satisfies **A1**. Indeed, each $\mathcal{E}_\sigma^j$ contains all and only the traces leading to the states contained in X_m^j. Hence, whenever we say that a decomposition $D = \{\mathcal{E}_\sigma^1, \cdots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ satisfies **A1**, this means that there exists a decomposition $\{X_m^1, \cdots, X_m^p\}$ of X_m such that every $X_m^j \in D$ is given by (4.11).

The following proposition shows how the languages $\mathcal{E}_\sigma^j[k]$ and $\mathcal{D}_\sigma^j[k]$, $\forall k \geq 1$, are obtained from (4.11).

Proposition 4.5.2 *Consider a decomposition $\{X_m^1, \cdots, X_m^p\}$ of X_m. By considering the decomposition $\{\mathcal{E}_\sigma^1, \cdots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ such that each $\mathcal{E}_\sigma^j$ is given by (4.11), $\mathcal{E}_\sigma^j[k]$ and $\mathcal{D}_\sigma^j[k]$, $\forall k \geq 1$, are computed as follows :*

$$\begin{aligned} \mathcal{E}_\sigma^j[k] &= \{s \in \Sigma^* \,|\, \alpha[k](x_0[k], s) \in X_m[k]|_{X_m^j}\}, \\ \mathcal{D}_\sigma^j[k] &= \{s \in \Sigma^* \,|\, \beta[k](y_0[k], s) \in Y_m[k]|_{X_m^j}\}. \end{aligned} \tag{4.12}$$

4.5.4 Checking if $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf_{\leq N}^{\geq 1}$-Coobs w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$

Let us now show how the multi-marking can be used to check whether $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$. For that purpose, we target to find a decomposition $\{X_m^1, \cdots, X_m^p\}$ of

X_m such that, $\forall j \in J$, X_m^j satisfies the following condition,

$$\forall (v_1, \cdots, v_{n_\sigma}, v_{n_\sigma+1}) \in Y_m[1]|_{X_m^j}, \forall (a_1, \cdots, a_{n_\sigma}) \in (v_1 \cap X_m^j) \times \cdots \times (v_{n_\sigma} \cap X_m^j) : \\ |\bigcup_{i \in \mathbb{I}_\sigma} \{a_i\}| = 1. \tag{4.13}$$

Before explaining Condition (4.13), recall that a state of $Y_m[1]$ is in the form $(v_1, \cdots, v_{n_\sigma}, v_{n_\sigma+1})$ where $v_i \cap X_m \neq \emptyset$, $\forall i \in \mathbb{I}_\sigma$, and $v_{n_\sigma+1} \in Y_m$. Recall also that for a decomposition $\{X_m^1, \cdots, X_m^p\}$ of X_m and for a $j \in J$, a state $v \in Y_m[1]$ is X_m^j-marked if all its v_i, $i = 1, \cdots, n_\sigma$, contain an X_m^j-marked of X_m. Condition (4.13) requires that for every $v \in Y_m[1]$ and every $j \in J$, if all its v_i, $i = 1, \cdots, n_\sigma$, contain states of X_m^j, then all these v_i contain in fact the *same single state* $x \in X_m^j$ and no other state of X_m^j. The relevance of (4.13) is due to the fact that if a state $v \in Y_m[1]$ is X_m^j-marked for some j in a decomposition $(X_m^j)_{j \in J}$ such that $v_i \cap X_m^j = \{x\}, \forall i \in \mathbb{I}_\sigma$, then every other decomposition of X_m has one of its element, say $\mathcal{X}$, such that v is $\mathcal{X}$-marked and $v_i \cap \mathcal{X} = \{x\}, \forall i \in \mathbb{I}_\sigma$, if $x \in \mathcal{X}$. Note that (4.13) is satisfied by the trivial partition $(X_m^j)_{j=1 \cdots |X_m|}$ such that each X_m^j is a singleton.

If for a decomposition $(X_m^j)_{j \in J}$ satisfying (4.13), $Y_m[1]$ has no X_m^j-marked state, i.e., $Y_m[1]|_{X_m^j} = \emptyset$, for every $j \in J$, then every corresponding $\mathcal{D}_\sigma^j[1]$ is empty, that is, every $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is Inf_0-COOBS. Consequently, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$(Inf_0, \cdots, Inf_0)$-COOBS.

In the case where $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not Inf_0-COOBS, if for a decomposition satisfying (4.13), $Y_m[1]$ has at least one X_m^j-marked state for some j, thus we have $\mathcal{D}_\sigma^j[1] \neq \emptyset$ for the $\mathcal{E}_\sigma^j$ corresponding to X_m^j. Therefore, $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is not Inf_0-COOBS for some $j \in J$. In this case, we have to check if $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is Inf_N-COOBS for some $N \geq 0$. Before that, we show in the following lemma an important result that will be used to show the relevance of (4.13).

Lemma 4.5.3 *Consider a decomposition $D = \{X_m^1, \cdots, X_m^p\}$ of X_m satisfying (4.13). $\forall k \geq 1$, $\forall v \in X_m[k] \cup Y_m[k]$, if v is X_m^j-marked for some $X_m^j \in D$, then there exists $x \in X_m^j$ such that v is $\{x\}$-marked and not $\mathcal{X}$-marked, $\forall \mathcal{X} \subseteq X_m^j$ s.t. $\mathcal{X} \neq \{x\}$.*

We have the following theorem which implies that $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS of $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ needs to be checked uniquely in a decomposition satisfying (4.13).

Theorem 4.5.1 *Consider an integer $N \geq 0$, a FSA $\mathcal{A}_{\mathcal{E}_\sigma}$ accepting $\mathcal{E}_\sigma$ and a decomposition $\{X_m^1, \ldots, X_m^p\}$ of X_m satisfying (4.13). If $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not $\vee$-$Inf_{\leq N+1}^{\geq 1}$-COOBS w.r.t. $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$, where $\mathcal{E}_\sigma^j$ is given by (4.11), then $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$.*

Given a decomposition $(X_m^j)_{j\in J}$ satisfying (4.13), we compute iteratively for each $j \in J$: $(\mathcal{A}_{\mathcal{E}_\sigma[k]}, \mathcal{A}_{\mathcal{D}_\sigma[k]})$ until $k = N+1$, or $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ or $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ has no X_m^j-marked states, $\forall j \in J$:

For each $j \in J \cdot (\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is Inf_{k-1}-COOBS if $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ or $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ has no X_m^j-marked state, for $k \leq N$. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$ if, $\forall j \in J$, we have found some $N_j \leq N$ such that $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is Inf_{N_j}-COOBS. Otherwise, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS. $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is not $Inf_{\geq 0}$-COOBS if for some $k < N$, $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ have the same non-empty X_m^j-marked language (an X_m^j-marked language is the set of traces leading to an X_m^j-marked state), and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ have the same non-empty X_m^j-marked language $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not $\vee$-$Inf_{\geq 0}^{\geq 1}$-COOBS if we have found some $j \in J$ such that $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is not $Inf_{\geq 0}$-COOBS.

Consider a decomposition $(X_m^j)_{j\in J}$ and let us evaluate the computational complexity for checking whether $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is Inf_k-COOBS, $\forall j \in J$ and for some $k \in \mathbb{Z}^+$. That is, whether $\mathcal{E}_\sigma^j[k+1]$ or $\mathcal{D}_\sigma^j[k+1]$ is empty, $\forall j \in J$. We are interested by the *worst case* for a *single step* k of inference. By "single step k", we mean, $\forall j \in J$, $\mathcal{A}_{\mathcal{E}_\sigma^j[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma^j[k]}$ are assumed computed. By "worst case", we mean that, $\forall j \in J$, $\mathcal{A}_{\mathcal{E}_\sigma^j[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma^j[k]}$ are assumed non-null, and thus, $\mathcal{A}_{\mathcal{E}_\sigma^j[k+1]}$ and $\mathcal{A}_{\mathcal{D}_\sigma^j[k+1]}$ need to be computed before to know whether they are null. We have the following proposition which can be deduced from Lemma 4.5.2 :

Proposition 4.5.3 *Consider a decomposition $(X_m^j)_{j\in J}$ and a step k of inference. Assuming that $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ are computed, the complexity for checking whether $\mathcal{E}_\sigma^j[k+1]$ or $\mathcal{D}_\sigma^j[k+1]$ is empty, $\forall j \in J$, is in the worst case in the following order :*

$$O((|X[k+1]|^2 + |Y[k+1]|^2) \cdot |\Sigma|) = O((|X[k]|^{2n_\sigma} \cdot |Y[k]|^2 + |Y[k]|^{2n_\sigma} \cdot |X[k]|^2) \cdot |\Sigma|).$$

4.5.5 Procedure of decomposition of X_m

We have noted that (4.13) is satisfied by the trivial partition $(X_m^j)_{j=1\ \ p}$ where each X_m^j is a singleton, and thus, $p = |X_m|$. To reduce execution time and memory space during the execution of control, it is preferable to find a non trivial partition satisfying (4.13) with a smaller p, if any. Let us propose a procedure that computes such a non trivial partition $(X_m^j)_{j\in J}$, if any. Before presenting our partition procedure, we need to define Elig($\mathcal{X}$).

Definition 4.5.1 *Consider $\mathcal{X} \subseteq X_m$ such that $\mathcal{X}$ satisfies (4.13).* Elig($\mathcal{X}$) *contains all the states of $X_m \setminus \mathcal{X}$ that can be added individually to $\mathcal{X}$ without violating (4.13). Formally,*

$$\text{Elig}(\mathcal{X}) = \{x \in X_m \backslash \mathcal{X} \mid \mathcal{X} \cup \{x\} \text{ satisfies (4.13)}\}.$$

In the sequel, $\text{Elig}(\{x\})$ is written $\text{Elig}(x)$. Note that, $\text{Elig}(\emptyset) = X_m$. Our partition procedure will construct every X_m^j by moving iteratively some states from X_m to X_m^j ; let then Z_m denote the current remaining part of X_m (i.e., states of X_m that have not yet been moved to a X_m^j). The basic principle for constructing X_m^1 is as follows :

1. Initializations : (a) $Z_m \leftarrow X_m$, (b) $X_m^1 \leftarrow \emptyset$,
2. While $\text{Elig}(X_m^1) \cap Z_m \neq \emptyset$:
   We select randomly a state $x \in \text{Elig}(X_m^1) \cap Z_m$,
   We move x from Z_m to X_m^1, that is : $X_m^1 \leftarrow X_m^1 \cup \{x\}$, $Z_m \leftarrow Z_m \setminus \{x\}$.

The construction of X_m^1 is completed when the while-loop terminates, i.e., when $\text{Elig}(X_m^1) \cap Z_m = \emptyset$. Note that this while-loop terminates in finite time because the computed sets $\text{Elig}(X_m^1) \cap Z_m$ are finite and define a monotonically decreasing sequence of state sets. The above procedure guarantees that the computed X_m^1 satisfies (4.13) and that (4.13) is not satisfied as soon as we add any other state to X_m^1.

The other X_m^j , $j > 1$, are constructed by repeating the above procedure without Substep 1(a). The construction of the partition is complete when $Z_m = \emptyset$.

The set $\text{Elig}(\mathcal{X})$ has been defined in Def. 4.5.1, let us now see how it is computed for any $\mathcal{X} \subseteq X_m$. We define for every $v = (v_1, \ldots, v_{n_\sigma}, v_{n_\sigma+1}) \in Y_m[1]$, and $\mathcal{X} \subseteq X_m$, $\text{ID}_v(\mathcal{X})$ by :

$$\text{ID}_v(\mathcal{X}) = \{i \in \mathbb{I}_\sigma \mid v_i \cap \mathcal{X} = \emptyset\}, \tag{4.14}$$

therefore, $\text{ID}_v(\emptyset) = \mathbb{I}_\sigma$. In the next lemma we present a property of $\text{ID}_v(.), \forall v \in Y_m[1]$.

Lemma 4.5.4 *Given two subsets $\mathcal{W}, \mathcal{Z} \subseteq X_m$, we have :* $\text{ID}_v(\mathcal{W} \cup \mathcal{Z}) = \text{ID}_v(\mathcal{W}) \cap \text{ID}_v(\mathcal{Z})$.

The following lemma states a new reformulation of (4.13) by using the function $\text{ID}_v(.)$:

Lemma 4.5.5 *A subset $\mathcal{X} \subseteq X_m$ satisfies (4.13) iff :*

$$\forall v \in Y_m[1] : \text{ID}_v(\mathcal{X}) = \emptyset \Rightarrow |\mathcal{X} \cap \bigcup_{i \in \mathbb{I}_\sigma} v_i| = 1 \tag{4.15}$$

We will now show how the functions $\text{ID}_v(.), \forall v \in Y_m[1]$, can be used for computing $\text{Elig}(\mathcal{X})$, for $\mathcal{X} \subseteq X_m$. We will present an *inductive* computation method :

Basis : When $\mathcal{X}$ is a singleton $\{x\}$, $\text{Elig}(x)$ can be computed as follows :

Proposition 4.5.4 *Given a marked state $x \in X_m$,*

$$\mathrm{Elig}(x) = X_m \setminus [\{x\} \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_\iota(x)=\emptyset}} \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_v(x)} v_\iota] \tag{4.16}$$

Inductive step : $\mathrm{Elig}(\mathcal{X} \cup \{x\})$ can be computed from $\mathrm{Elig}(\mathcal{X})$ and $\mathrm{Elig}(x)$ as follows :

Proposition 4.5.5 *Given a subset $\mathcal{X} \subseteq X_m$ satisfying (4.13), for every $x \in \mathrm{Elig}(\mathcal{X})$ we have :*

$$\mathrm{Elig}(\mathcal{X} \cup \{x\}) = (\mathrm{Elig}(\mathcal{X}) \cap \mathrm{Elig}(x)) \setminus [\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_\iota(\mathcal{X} \cup \{x\}) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_\iota(\mathcal{X} \cup \{x\})} v_\iota]. \tag{4.17}$$

Let us evaluate the complexity of the above 2-step partition procedure. For that purpose, we first need to evaluate the complexities of computing : $\mathrm{ID}_v(x)$ and $\mathrm{Elig}(x)$.

Lemma 4.5.6 *Given $x \in X_m$ and $v \in Y_m[1]$, the complexity for computing $\mathrm{ID}_v(x)$ is bounded by $O(n_\sigma \cdot |X|)$.*

Lemma 4.5.7 *Given $x \in X_m$, the complexity for computing $\mathrm{Elig}(x)$ is bounded by $O(n_\sigma \cdot |Y_m[1]| \cdot |X|^2)$.*

Lemma 4.5.8 *Consider $x \in X_m$, $\mathcal{X} \subseteq X_m$, and assume we are given $\mathrm{Elig}(\mathcal{X})$ and $\mathrm{Elig}(x)$. The complexity for computing $\mathrm{Elig}(\mathcal{X} \cup \{x\})$ (by (4.17)) is bounded by $O(n_\sigma \cdot |Y_m[1]| \cdot |X|^2)$.*

Proposition 4.5.6 *The complexity of our partition procedure is bounded by $O(n_\sigma \cdot |Y_m[1]| \cdot |X|^3)$.*

4.5.6 Conclusion on the complexity of our framework

Let us compare the computational complexity of our multi-decision framework with the complexity of the inference-based framework of [Kumar et Takai, 2005]. We have evaluated the complexities of the operations of our framework throughout this section 4.5. The most costly "new" operations that have been added to the framework of [Kumar et Takai, 2005] to construct our multi-decision control are :

In Subsection 4.5.5 : the partition procedure;

In Subsection 4.5.4 : the procedure for checking emptiness of $\mathcal{E}_\sigma^\jmath[k+1]$ and $\mathcal{D}_\sigma^\jmath[k+1]$, $\forall \jmath \in J$, in one step k, that is, $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ are given. For brevity, we will call it "checking procedure".

Consider the operation for computing $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$, which is a part of both frameworks ([Kumar et Takai, 2005] and ours). Let us compare its complexity (evaluated in lemma 4.5.2 of Section 4.5.1) with the complexities of the above two procedures (partition and checking) :

1. The complexity of lemma 4.5.2 is higher than the complexity of the partition procedure evaluated in Proposition 4.5.6 (Section 4.5.5).
2. The complexity of lemma 4.5.2 is higher than the complexity of the checking procedure evaluated in Proposition 4.5.3 (Section 4.5.4).

Consequently, in terms of Big-Oh, the complexity of our multi-decision framework is comparable to the complexity of the inference-based framework of [Kumar et Takai, 2005]. Intuitively, this result may be surprising. A possible explanation is that we have restricted the set of possible decompositions in two ways :

1. We have used a finite state-based approach by using assumption **A1** (Sect. 4.4). In this way, we have eliminated the decompositions that do not satisfy **A1** w.r.t. a given FSA. The number of these eliminated decompositions may be infinite, while the number of the remaining eligible decompositions is certainly finite.
2. We have developed a procedure that computes a *single* partition, which guarantees that if we have not coobservability for this partition, then there exists no decomposition for which we have coobservability.

Therefore, computing $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ is in general more complex in comparison to computing a partition that satisfies (4.13). Hence, the overall complexity to compute a partition satisfying (4.13), and then checking if $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf^{\geq 0}_{\leq N}$-COOBS, is in the order of $O(|Y|^2 \cdot |X|^{2n_\sigma} \cdot |\Sigma|)$.

4.6 Algorithm for computing a partition of $\mathcal{E}_\sigma$ and checking $\vee$-$Inf^{\geq 0}_{\leq N}$-Coobs

Algorithm 1 (represented in the next page) implements our results of Section 4.5. That is, for $\sigma \in \Sigma_c$, it constructs a partition of X_m satisfying (4.13) and then checks if $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf^{\geq 0}_{\leq N}$-COOBS. Therefore, the algorithm must be executed for every $\sigma \in \Sigma_c$. Based on results presented throughout Section 4.5, the following theorem states the correctness of our algorithm.

Theorem 4.6.1 *Algorithm 1 is correct in the sense that each of its three outputs is generated if and only if it is true.*

Input: G, K, N.
Initialization : $np \leftarrow 1$;
Compute $\mathcal{A}_{\mathcal{D}_\sigma}$, $\mathcal{A}_{\mathcal{E}_\sigma}$, $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ and $\mathcal{A}_{\mathcal{E}_\sigma[1]}$; /* See Subsection 4.5.1 */
if $[Y_m[1] = \emptyset] \vee [X_m[1] = \emptyset]$ **then** return "$(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is Inf_0-COOBS"; /* See Subsection 4.5.2 */
/* Compute $\mathrm{ID}_v(x)$ and $\mathrm{Elig}(x)$ as explained in Subsection 4.5.5 */
foreach $x \in X_m$ **do**
  foreach $v \in Y_m[1]$ **do** Compute $\mathrm{ID}_v(x)$; /* by using (4.14) */
  Compute $\mathrm{Elig}(x)$; /* by using (4.16) */
end
/* Compute partition as explained in Subsection 4.5.5 */
$Z_m \leftarrow X_m$, $\jmath \leftarrow 1$;
while $Z_m \neq \emptyset$ **do**
  Select some x in Z_m;
  $Z_m \leftarrow Z_m \backslash \{x\}$, $X^\jmath_m \leftarrow X^\jmath_m \cup \{x\}$; /* the selected x is moved from Z_m to $X^\jmath_m$ */
  while $\mathrm{Elig}(X^\jmath_m) \cap Z_m \neq \emptyset$ **do**
    foreach $v \in Y_m[1]$ **do**
      $\mathrm{ID}_v(X^\jmath_m \cup \{x\}) = \mathrm{ID}_v(X^\jmath_m) \cap \mathrm{ID}_v(x)$; /* Using Lemma 4.5.4 */
    end
    Select some x in $\mathrm{Elig}(X^\jmath_m) \cap Z_m$;
    $X^\jmath_m \leftarrow X^\jmath_m \cup \{x\}$; $Z_m \leftarrow Z_m \backslash \{x\}$; /* the selected x is moved from Z_m to $X^\jmath_m$ */
    Compute $\mathrm{Elig}(X^\jmath_m)$; /* Using (4.17) */
  end
  $\jmath \leftarrow \jmath + 1$;
end
$p \leftarrow j$;
/* Check if $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS as explained in Subsection 4.5.4 */
$\mathrm{NPar} \leftarrow \{1, \ldots, p\}$;
for $k \leftarrow 1$ **to** N **do**
  foreach $\jmath \in \mathrm{NPar}$ **do**
    if $[Y_m[k]|_{X^\jmath_m} = \emptyset] \vee [X_m[k]|_{X^\jmath_m} = \emptyset]$ **then** $N_\jmath \leftarrow k - 1$; Remove j from NPar;
  end
  if $[\mathrm{NPar} = \emptyset]$ **then** return "$(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$Inf_{N_1}, \ldots, Inf_{N_p}$-COOBS";
  if $k \leq N$ **then**
    Compute $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$, $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$; /* See Subsection 4.5.1 */
  else
    Return "$(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not $\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS";
  end
end

Algorithm 1: Constructing a partition of X_m and checking if $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is Inf_0-COOBS or $\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS

4.7 Example

Consider the plant G of Fig. 4.1, the specification is obtained by forbidding the dashed transitions. Supervisor Sup_1 observes $\Sigma_{o,1} = \{a_1, b_1, c_1\}$, Supervisor Sup_2 observes $\Sigma_{o,2} = \{a_2, b_2, c_2\}$, and both supervisors control $\Sigma_{c,1} = \Sigma_{c,2} = \{\sigma\}$. The automaton $\mathcal{A}_{\mathcal{E}_\sigma}$ is obtained from the automaton of Fig. 4.1 by marking the four states $X_m = \{x^1, x^2, x^3, x^4\}$ and removing states 6, 7, 8, 9, y^2 and y^3. The automaton $\mathcal{A}_{\mathcal{D}_\sigma}$ is obtained from the automaton of Fig. 4.1 by marking the three states $Y_m = \{y^1, y^2, y^3\}$ and removing states 1, 2, 4, 7, 8, 9, x^1, x^2, x^3 and x^4.

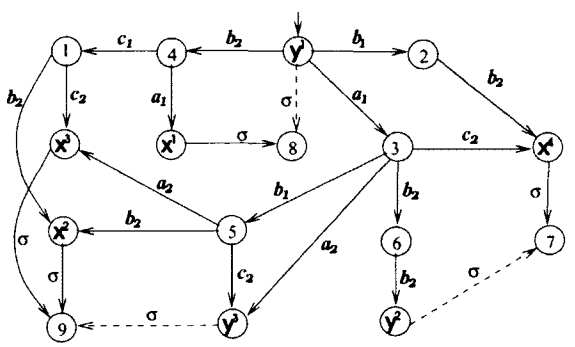

Figure 4.1 Plant and specification. The latter is obtained by forbidding the dashed transitions

The FSA $\mathcal{A}_{\mathcal{E}_\sigma[1]}$ has its states defined by (u_1, u_2, v) where $u_1 \cap Y_m \neq \emptyset$, $u_2 \cap Y_m \neq \emptyset$, and $v \in X_m$. $\mathcal{A}_{\mathcal{E}_\sigma[1]}$ is represented on Fig. 4.2 where each state (u_1, u_2, v) is represented by a three-part square containing u_1, u_2 and v, respectively. After the execution of event b_2, $\mathcal{A}_{\mathcal{E}_\sigma[1]}$ may be in two equivalent states $(\{1,4\}, \{4, y^1\}, 4)$ or $(\{1,4\}, \{6\}, 4)$, which are distinct by their second parts $\{4, y^1\}$ and $\{6\}$. The two states are represented by a single three-part square where their distinct second parts $\{4, y^1\}$ and $\{6\}$ are stacked in the second part of the square. The marked states are shaded.

In a similar way, the automaton $\mathcal{A}_{\mathcal{D}_\sigma[1]}$, represented on Fig. 4.3, has its states defined by (v_1, v_2, u) where $v_1 \cap X_m \neq \emptyset$, $v_2 \cap X_m \neq \emptyset$, and $u \in Y_m$. Each state (v_1, v_2, u) is represented by a three-part square containing v_1, v_2 and u, respectively. The same approach is used to represent several equivalent states reached by the same sequences.

Note that with our representation of states in three-part squares, (4.13) is not satisfied if in a marked state v of $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ (i.e., $v \in Y_m[1]$), we have elements $a \in X_m$ and $b \in X_m$ $(a \neq b)$, in parts 1 and 2 respectively, such that a and b are in the same X_m^j.

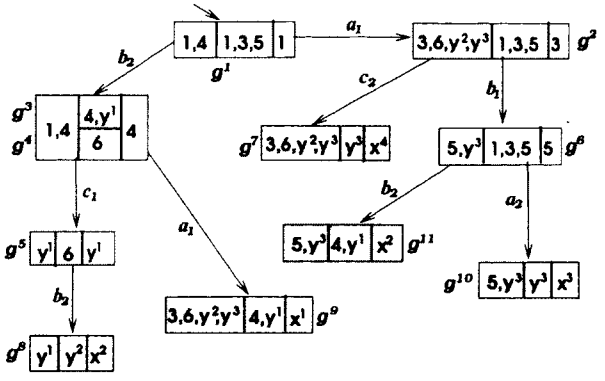

Figure 4.2 $\mathcal{A}_{\mathcal{E}_\sigma[1]}$ for the example of Figure 4.1

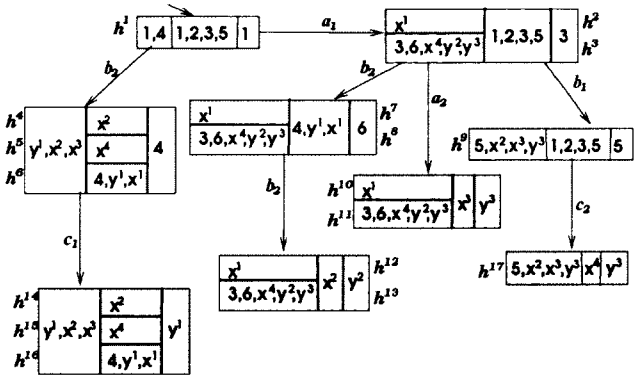

Figure 4.3 $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ for the example of Figure 4.1

Let us show how the procedure of Section 4.5.5 is used to partition X_m. For that purpose, we first compute $\mathrm{ID}_v(\{x\})$ for every $x \in X_m$ (i.e., marked states of $\mathcal{A}_{\mathcal{E}_\sigma}$) and every $v \in Y_m[1]$ (i.e., marked states of $\mathcal{A}_{\mathcal{D}_\sigma[1]}$), which is represented in Table 4.1. Then, by using

$v =$	h^{10}	h^{11}	h^{12}	h^{13}	h^{14}	h^{15}	h^{16}	h^{17}
$\mathrm{ID}_v(x^1)$	$\{2\}$	$\mathbb{I}_\sigma$	$\{2\}$	$\mathbb{I}_\sigma$	$\mathbb{I}_\sigma$	$\mathbb{I}_\sigma$	$\{1\}$	$\mathbb{I}_\sigma$
$\mathrm{ID}_v(x^2)$	$\mathbb{I}_\sigma$	$\mathbb{I}_\sigma$	$\{1\}$	$\{1\}$	$\emptyset$	$\{2\}$	$\{2\}$	$\{2\}$
$\mathrm{ID}_v(x^3)$	$\{1\}$	$\{1\}$	$\mathbb{I}_\sigma$	$\mathbb{I}_\sigma$	$\{2\}$	$\{2\}$	$\{2\}$	$\{2\}$
$\mathrm{ID}_v(x^4)$	$\mathbb{I}_\sigma$	$\{2\}$	$\mathbb{I}_\sigma$	$\{2\}$	$\mathbb{I}_\sigma$	$\{1\}$	$\mathbb{I}_\sigma$	$\{1\}$

Tableau 4.1 Computing $\mathrm{ID}_v(\{x\})$ for all states $v \in Y_m[1]$ and $x \in X_m$.

the functions $\mathrm{ID}_v(.)$, for every $v \in Y_m[1]$, and the Equation (4.16), we compute the sets $\mathrm{Elig}(x)$ of eligible states for every marked state $x \in X_m$:

$x = x^1$: We compute

$$A = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x^1)=\emptyset}} \bigcup_{i \in \mathbb{I}_\sigma} v_i = \emptyset,$$

$$B = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_\iota(x^1) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_v(x^1)} v_\iota = X_m \cap (h_2^{10} \cup h_2^{12} \cup h_1^{16}) = \{x^3\} \cup \{x^2\} \cup \{x^2, x^3\} = \{x^2, x^3\}.$$

Then, we obtain $\mathrm{Elig}(x^1) = X_m \setminus [\{x^1\} \cup A \cup B] = \{x^4\}$.

$x = x^2$: We compute

$$A = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x^2) = \emptyset}} \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota = X_m \cap (h_1^{14} \cup h_2^{14}) = \{x^2, x^3\},$$

$$B = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x^2) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_\iota(x^2)} v_\iota = X_m \cap (h_1^{12} \cup h_1^{13} \cup h_2^{15} \cup h_2^{16} \cup h_2^{17}) = \{x^1, x^4\}.$$

Then, we obtain $\mathrm{Elig}(x^2) = X_m \setminus [\{x^2\} \cup A \cup B] = X_m \setminus \{x^1, x^2, x^3, x^4\} = \emptyset$.

$x = x^3$: We compute

$$A = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x^3) = \emptyset}} \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota = \emptyset,$$

$$B = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x^3) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_v(x^3)} v_\iota = X_m \cap (h_1^{10} \cup h_1^{11} \cup h_2^{14} \cup h_2^{15} \cup h_2^{16} \cup h_2^{17}) = \{x^1, x^2, x^4\}.$$

Then, we obtain $\mathrm{Elig}(x^3) = X_m \setminus [\{x^3\} \cup A \cup B] = X_m \setminus \{x^1, x^2, x^3, x^4\} = \emptyset$.

$x = x^4$: We compute

$$A = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x^4) = \emptyset}} \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota = \emptyset,$$

$$B = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x^4) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_\iota(x^4)} v_\iota = X_m \cap (h_2^{11} \cup h_2^{13} \cup h_1^{15} \cup h_1^{17}) = \{x^2, x^3\}.$$

Then, we obtain $\mathrm{Elig}(x^4) = X_m \setminus [\{x^4\} \cup A \cup B] = X_m \setminus \{x^2, x^3, x^4\} = \emptyset$.

Let us now explain the construction of each X_m^j, using the automaton $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ of Fig. 4.3 and the following intuitive expression of (4.13) : The latter requires that for every $v = (v_1, v_2, w)$ of $Y_m[1]$ and every $j \in J$, if both parts 1 and 2 of v (i.e., v_1 and v_2) contain states of X_m^j, then all these parts contain in fact the same single state of X_m^j and no other state of X_m^j.

a) **Construction of X_m^1 :** Initially $X_m^1 = \emptyset$ and $Z_m = X_m$. We select $x^1 \in \mathrm{Elig}(\emptyset) \cap Z_m = Z_m$ and move it to X_m^1, we obtain $X_m^1 = \{x^1\}$ and $Z_m = \{x^2, x^3, x^4\}$. We compute $\mathrm{Elig}(X_m^1) \cap Z_m = \mathrm{Elig}(x^1) \cap Z_m = \{x^4\}$. Therefore, x^4 is the only state of Z_m that can be added to X_m^1 without violating (4.13). Therefore, we take $X_m^1 = \{x^1, x^4\}$ and we

have $Z_m = \{x^2, x^3\}$. Let us explain intuitively why x^4 can be put with x^1, and why x^2 and x^3 cannot :

If we move x^2 to $X_m^1 = \{x^1\}$, X_m^1 becomes $\{x^1, x^2\}$. Condition (4.13) is violated because in the marked state $h^{12} = (\{x^1\}, \{x^2\}, y^2)$ of $Y_m[1]$, both parts 1 and 2 contain *distinct* states of X_m^1 : x^1 and x^2, respectively. Therefore, x^2 is not moved to X_m^1.

If we move x^3 to $X_m^1 = \{x^1\}$, X_m^1 becomes $\{x^1, x^3\}$. Condition (4.13) is violated because in the marked state $h^{10} = (\{x^1\}, \{x^3\}, y^3)$ of $Y_m[1]$, parts 1 and 2 contain *distinct* states of X_m^1 : x^1 and x^3, respectively. Therefore, x^3 is not moved to X_m^1.

If we move x^4 to $X_m^1 = \{x^1\}$, X_m^1 becomes $\{x^1, x^4\}$. Condition (4.13) is satisfied because there is no marked state of $Y_m[1]$ whose both parts 1 and 2 contain elements of X_m^1. Therefore, we move x^4 to X_m^1 and obtain $X_m^1 = \{x^1, x^4\}$.

b) **Construction of X_m^2 :** Initially $X_m^2 = \emptyset$ and $Z_m = \{x^2, x^3\}$. We select $x^2 \in \text{Elig}(\emptyset) \cap Z_m = Z_m$ and move it to X_m^2, we obtain $X_m^2 = \{x^2\}$ and $Z_m = \{x^3\}$. We compute $\text{Elig}(X_m^2) \cap Z_m = \text{Elig}(x^2) \cap Z_m = \emptyset$ from the fact that $\text{ID}_{h^{14}}(x^2) = \emptyset$. Therefore, no state of Z_m can be added to X_m^2 without violating Condition (4.13). Therefore, we take $X_m^2 = \{x^2\}$ and we have $Z_m = \{x^3\}$.

Let us explain intuitively why x^3 cannot be put with x^2 : If we move x^3 to $X_m^2 = \{x^2\}$, X_m^2 becomes $\{x^2, x^3\}$. Condition (4.13) is violated because in the marked state $h^{14} = (\{y^1, x^2, x^3\}, \{x^2\}, y^1)$ of $Y_m[1]$, parts 1 and 2 contain *distinct* states of X_m^2 : x^3 and x^2, respectively. Therefore, x^3 is not moved to X_m^2.

c) **Construction of X_m^3 :** We take $X_m^3 = \{x^3\}$ because x^3 is the only remaining state in Z_m.

To recapitulate, we have $X_m^1 = \{x^1, x^4\}$, $X_m^2 = \{x^2\}$ and $X_m^3 = \{x^3\}$. We will thus apply the triple-marking : $(X_m^j)_{j=1,2,3}$ to the marked states of $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ as explained in Subsection 4.5.4. Then, we check how the X_m^j-marking is propagated to $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ of Fig. 4.3. Note that with our representation of states in three parts, a marked state u of $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ (i.e., $u \in Y_m[1]$) is X_m^j-marked if both parts 1 and 2 of u contain states of X_m^j. We thus understand visually that h^{14} is X_m^2-marked and no other state is X_m^j-marked. Therefore, $Y_m[1]|_{X_m^1} = Y_m[1]|_{X_m^3} = \emptyset$, and $Y_m[1]|_{X_m^2} = \{h_{14}\} \neq \emptyset$. Hence, $(\mathcal{E}_\sigma^1, \mathcal{D}_\sigma)$ and $(\mathcal{E}_\sigma^3, \mathcal{D}_\sigma)$ are Inf_0-COOBS, and $(\mathcal{E}_\sigma^2, \mathcal{D}_\sigma)$ is *not* Inf_0-COOBS. We now have to check whether $(\mathcal{E}_\sigma^2, \mathcal{D}_\sigma)$ is Inf_1-COOBS. For that purpose, we compute $\mathcal{A}_{\mathcal{E}_\sigma[2]}$ (see Fig. 4.4) and check if it has X_m^2-marked states, that is, states whose both first and second parts contain h^{14} and whose part is g^8 or g^{11} (because h^{14} is the only X_m^2-marked state of $\mathcal{A}_{\mathcal{D}_\sigma[1]}$, while g^8 and g^{11} are the X_m^2-marked states of $\mathcal{A}_{\mathcal{E}_\sigma[1]}$). In fact, $\mathcal{A}_{\mathcal{E}_\sigma[2]}$ has no X_m^2-marked state, and thus, $X_m[2]|_{X_m^2} = \emptyset$. Therefore, $(\mathcal{E}_\sigma^2, \mathcal{D}_\sigma)$ is Inf_1-COOBS.

To recapitulate, $(\mathcal{E}_\sigma^1, \mathcal{D}_\sigma)$, $(\mathcal{E}_\sigma^2, \mathcal{D}_\sigma)$, and $(\mathcal{E}_\sigma^3, \mathcal{D}_\sigma)$ are Inf_0-COOBS, Inf_1-COOBS, and Inf_0-COOBS, respectively. That is, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-(Inf_0, Inf_1, Inf_0)-COOBS.

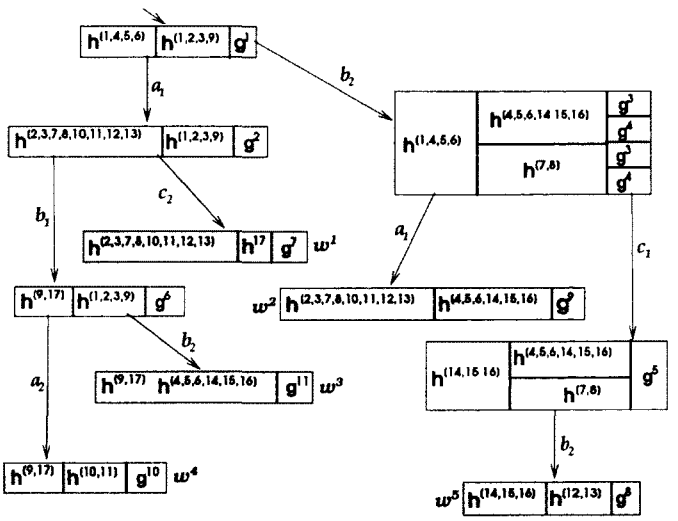

Figure 4.4 $\mathcal{A}_{\mathcal{E}_\sigma[2]}$ computed from $\mathcal{A}_{\mathcal{E}_\sigma[1]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ of Figures 4.2 and 4.3, $h^{[i,\ldots,j]}$ stands for $\{h^i, \ldots, h^j\}$

4.8 Conclusion

This paper is concerned with the verification of the coobservability in the context of the multi-decision control framework defined in [Chakib et Khoumsi, 2011b]. In the present paper, we consider more particularly the disjunctive inference-based multi-decision control, that is, the case where inference-based decentralized supervisors with different ambiguity levels are used in parallel and whose global decisions are combined disjunctively. But it has been shown in [Chakib et Khoumsi, 2011b] that any result obtained with the disjunctive combination can be easily adapted for the conjunctive combination.

The achievability of a specification in the context of (disjunctive) inference-based multi-decision control is related to specific notions of coobservability, namely $\vee$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-COOBS and $\vee$-$Inf_{\geq 0}^{\geq 1}$-COOBS (if p and $N_1, \ldots, N_p$ are not specified and not bounded) of a specification. We have also defined the more applicable notion of $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS for which $N_1, \ldots, N_p$ are bounded by N. Then, we have developed a method and its corresponding algorithm which checks if a given specification is $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS. We show that with our algorithm, the worst-case computational complexity for checking $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS is not increased by the number of decentralized supervisors in parallel. That is,

we obtain the same order of complexity as in the inference-based framework of [Kumar et Takai, 2007].

CHAPITRE 5

Diagnostic multi-décisionnel : architectures décentralisées coopérant pour diagnostiquer la présence de défauts dans des SED

Article : Hicham Chakib et Ahmed Khoumsi, *Multi-Decision Diagnosis : Decentralized Architectures Cooperating for Diagnosing the Presence of Faults in Discrete Event Systems* Submitted to Journal of Discrete Event Dynamical Systems, 2011.

Avant-propos

Auteurs et affiliation :

Hicham Chakib : étudiant au doctorat, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Ahmed Khoumsi : professeur, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Date de soumission : première soumission le 2 février 2010 et la deuxième soumission le 20 novembre 2010

Revue : Journal of Discrete Event Dynamic Systems : Theory & Applications

Titre français : Diagnostic multi-décisionnel : architectures décentralisées coopérant pour diagnostiquer la présence de défauts dans des SED

Contribution au document : Dans les chapitres 3 et 4, nous avons développé le contrôle multi-décisionnel. Plus précisément, une contribution essentielle du chapitre 3 a été d'utiliser le contrôle multi-décisionnel pour généraliser les architectures de contrôle par inférence. Et au chapitre 4, nous avons développé une méthode efficace pour vérifier si une spécification est coobservable selon le contrôle multi-décisionnel par inférence.

Dans [Khoumsi et Chakib, 2008a], nous avons développé le diagnostic multi-décisionnel pour généraliser quelques architectures de diagnostic existantes dans la littérature. Dans [Chakib et Khoumsi, 2009], nous avons utilisé le diagnostic multi-décisionnel pour généraliser une classe générique d'architectures de diagnostic dites éligibles. Nous avons étudié ensuite plus spécifiquement le diagnostic multi-décisionnel pour généraliser une classe particulière d'architectures éligibibles : les architectures disjunctives.

La présente contribution utilise le diagnostic multi-décisionnel pour généraliser les architectures de diagnostic par inférence. Et ensuite, nous développons une méthode efficace pour vérifier si une spécification est codiagnosticable selon le diagnostic multi-décisionnel par inférence.

Résumé français : Cet article étudie le diagnostic décentralisé, où un ensemble de diagnostiqueurs coopèrent pour détecter des défauts dans des SED. Nous proposons une approche, appelée *diagnostic multi-décisionnel*, dont le principe de base consiste à utiliser plusieurs architectures de diagnostic décentralisées fonctionnant en parallèle. On présente en premier lieu une forme générique du diagnostic multi-décisionnel, où plusieurs architectures de diagnostic décentralisées fonctionnent en parallèle et combinent leurs décisions globales disjonctivement ou conjonctivement. Nous étudions ensuite le diagnostic multi-décisionnel basé sur l'inférence par ambiguïté, c.à.d., dans le cas où chaque architecture décentralisée utilisée en parallèle est basée sur la technique de l'inférence par ambiguïté. On développe une méthode qui vérifie si une spécification donnée est diagonalisable sous l'architecture de l'inférence par ambiguïté multi-décisionnelle. Et nous montrons qu'avec notre méthode, la complexité de calcul pour vérifier la codiagnostiquabilité pour notre architecture à inférence par ambiguïté multi-décisionnelle est dans le même ordre de grandeur que la complexité de vérifier la codiagnostiquabilité pour l'architecture à inférence par ambiguïté de [Kumar et Takai, 2009].

Note : À la suite des corrections demandées par les membres du jury, le contenu de cet article diffère de celui qui a été soumis.

Abstract : This article deals with decentralized diagnosis, where a set of diagnosers cooperate for detecting faults in a discrete event system. We propose a new framework, called *multi-decision diagnosis*, whose basic principle consists in using several decentralized diagnosis architectures working in parallel. We first present a generic form of the multi-decision diagnosis, where several decentralized diagnosis architectures work in parallel and combine their global decisions disjunctively or conjunctively. We then study in more detail the inference-based multi-decision diagnosis, that is, in the case where each of the decentralized architectures in parallel is based on the inference-based framework. We develop a method that checks if a given specification is diagnosable under the inference-based multi-decision architecture. And we show that with our method, the worst-case computational complexity for checking codiagnosability for our inference-based multi-decision architecture is in the same order of complexity as checking codiagnosability for an inference-based architecture of [Kumar et Takai, 2009].

5.1 Introduction

This paper deals with the detection of faults (or *fault diagnosis*) in discrete event systems (DES). A fault is a deviation from an expected behavior. The fault diagnosis (or more shortly : diagnosis) in DES aims at detecting a fault within a bounded delay after its occurrence, by observing the DES behavior. The possibility to detect a fault is captured by the notion of (fault) *diagnosability* in DES introduced in [Sampath *et al.*, 1995]. Polynomial tests for diagnosability are given in [Jiang *et al.*, 2001] and [Yoo et Lafortune, 2002b]. The property of diagnosability in DES is related to the ability to infer, from observed event traces, about the occurrence of certain unobservable events (the *faulty* events). For DES modeled by regular languages, diagnosability can be verified by constructing the diagnoser corresponding to the finite-state automaton model of the system. Diagnosers are kinds of observers that carry failure information by means of labels attached to states.

We consider the case of several local diagnosers that cooperate for detecting faults. Each local diagnoser has a partial observation of the plant, the latter denoting the system to be diagnosed. Previous work on cooperating diagnosers includes the *distributed* diagnosis, where the local diagnosers communicate with each other [Debouk *et al.*, 2000; Qiu et Kumar, 2005; Sengupta, 1998], and the *decentralized* diagnosis, where the local diagnosers communicate with a global coordinator [Debouk *et al.*, 2000, 2003].

In this paper, we consider the decentralized diagnosis architecture, where each local diagnoser has a local observation of the plant and makes local diagnoses without communicating with the other local diagnosers. And the local diagnoses made by the various local diagnosers are fused in order to issue a global diagnosis.

We propose a multi-decision framework whose principle consists in using several decentralized architectures working in parallel. For example, we can have a *disjunctive* and a *conditionally conjunctive* architectures [Wang *et al.*, 2005, 2007] running in parallel. In this paper, we study in more detail the inference-based multi-decision, that is, the case where each of the architectures in parallel is inference-based [Kumar et Takai, 2009]. This is not restrictive, because it has been shown in [Kumar et Takai, 2009] that the inference-based architecture is more general than the previously developed architectures. The global diagnoses of all these decentralized architectures are fused adequately in order to obtain an *effective* diagnosis. The motivation of multi-decision framework is to obtain an architecture that generalizes all the decentralized architectures that compose it. Hence, our inference-based multi-decision framework generalizes the inference-based architecture of [Kumar et Takai, 2009].

The organization of the present paper is as follows. Discussion about related work is given in Section 5.2. Section 5.3 introduces some preliminaries about the decentralized diagnosis. Section 5.4 presents the principle of multi-decision for the decentralized diagnosis (or more briefly, *multi-decision diagnosis*). The remaining sections (i.e., 5.5-5.10) study the so-called conjunctive inference-based multi-decision architecture, that is, several inference-based architectures of [Kumar et Takai, 2006, 2009] which are fused *conjunctively*. Section 5.5 presents in detail this architecture, while Section 5.6 studies its existence of solutions and Section 5.7 studies some of its properties. In Section 5.8, we develop a method that checks if a given specification is diagnosable under the considered architecture. In the same section, we show that with our method, the worst-case computational complexity for checking codiagnosability is not increased by the number of decentralized diagnosers in parallel. That is, checking codiagnosability for our inference-based multi-decision architecture is in the same order of complexity as checking codiagnosability for an inference-based architecture of [Kumar et Takai, 2009], although the former framework is more general than the latter one. Section 5.9 contains an algorithm that implements the method of Section 5.8, and Section 5.10 illustrates this method in an example. The conclusion is presented in Section 5.11. And last but not least, the proofs are presented in an appendix.

5.2 Related work

In [Debouk *et al.*, 2000], the authors developed several decentralized diagnosis methods, each one based on a distinct communication protocol between the local diagnosers. They make diagnoses about the occurrence of faults, and then their diagnoses are merged by simple memoryless Boolean disjunction operations.

In [Qiu et Kumar, 2006], the authors suggested the following technique for managing ambiguity. A diagnosis decision is issued by a local diagnoser only when it is unambiguous about it. The authors of [Qiu et Kumar, 2006] introduce the codiagnosability meaning that : for each faulty trace executed by the plant, there is at least one local diagnoser that can unambiguously state this faulty execution, within a bounded delay. The authors of [Wang *et al.*, 2004, 2005] introduced the notion of conditional codiagnosability that is weaker than codiagnosability.

The authors of [Wang *et al.*, 2007] defined the decentralized diagnosis in a general framework, where n local diagnosers $Diag_i$, $i = 1, \ldots, n$, are used for detecting a fault f, each $Diag_i$ taking a local diagnosis h_i that belongs to an arbitrary set LD of local diagnoses. The local diagnoses are fused by using a global function D. The global diagnosis

can be either *positive* (fault detected) or *negative* (no fault detected). They target the following two objectives : "no missed detection", i.e., every fault occurrence is followed after a bounded delay δ by a positive diagnosis; "no false detection", i.e., a positive diagnosis guarantees that a fault has occurred The authors of [Wang *et al.*, 2007] introduce the *D-codiagnosability* to describe the languages for which there exist local diagnosis functions $(h_i)_{i=1,\ldots,n}$ such that every sufficiently long faulty trace is correctly diagnosed globally by fusing the local diagnoses $(h_i)_{i=1,\ldots,n}$ using the global function D. They studied the following cases : $|LD| = 1$, corresponding to *conjunctive codiagnosability* and *disjunctive codiagnosability* ; and $|LD| = 2$, corresponding to *conditionally* conjunctive codiagnosability and *conditionally* disjunctive codiagnosability.

Sengupta and Tripakis in [Sengupta et Tripakis, 2002] studied the case where the coordinating site could be any arbitrary memoryless function and local diagnoses may belong to an infinite set. They called *joint diagnosability* the class of languages diagnosable under these assumptions and they proved that joint diagnosability is undecidable.

Kumar and Takai in [Kumar et Takai, 2006, 2009] proposed the *inference-based ambiguity* that generalizes the special cases $|LD| = 1$ and $|LD| = 2$ of [Wang *et al.*, 2007]. Each local diagnoser uses its observations of the plant behavior to issue its diagnosis decision together with a level of ambiguity for that diagnosis decision. The principle is that when the coordinating site receives several concurrent diagnoses from the local diagnosers, it will select the diagnosis with the lowest ambiguity level.

The authors of [Kumar et Takai, 2006, 2009] target the same "no missed detection" as [Wang *et al.*, 2007], but with the introduction of ϕ, they reformulate a new "no false detection" as follows : a positive (resp. negative) diagnosis guarantees that a fault (resp. no fault) has occurred. Note that if in [Kumar et Takai, 2006, 2009] the global diagnosis ϕ is replaced by 0, then the objectives of [Wang *et al.*, 2007] become equivalent to those of [Kumar et Takai, 2006, 2009].

In [Takai et Kumar, 2006], the N-inference framework has been used for a variety of non-failure diagnosis which targets the following objective : after a sufficiently long healthy execution, the diagnosis system must be able to determine that until now no fault has occurred. We can find other variants of objectives related to the absence of faults, e.g., [Qiu et Kumar, 2006; Wang *et al.*, 2004].

In [Khoumsi et Chakib, 2008a], the multi-decision diagnosis has been defined uniquely in the special case where several disjunctive or several conjunctive architectures are running in parallel, without any mention of its generic form. In [Chakib et Khoumsi, 2009], the

multi-decision diagnosis has been studied in its generic form, that is, with any set of existing decentralized architectures running in parallel.

Note that the multi-decision has also been studied in the decentralized supervisory control of DES [Chakib et Khoumsi, 2008a,b] and prognosis of DES [?].

5.3 Preliminaries on decentralized diagnosis of DES

5.3.1 Diagnosis of DES

The plant to be diagnosed is modeled by a finite state automaton (FSA) $G=(Q,\Sigma,\delta,q_0)$, where Q is a finite set of states, Σ is a finite set of events, a partial function $\delta : Q\times\Sigma \rightarrow Q$ is the transition function and $q_0 \in Q$ is the initial state. Let Σ^* denote the set of all finite traces of elements of Σ, including the empty trace ε. The transition function δ can be generalized to $\delta : Q \times \Sigma^* \rightarrow Q$ in the usual way. For any event trace $s \in \Sigma^*$, $|s|$ denotes the length of s. We say that a trace $t \in \Sigma^*$ is a prefix of a trace $s \in \Sigma^*$, denoted by $t \leq s$, if there exists a trace $\lambda \in \Sigma^*$ such that $s = t\lambda$. We denote by $\overline{K}$ the set of all prefixes of traces of $K \subseteq \Sigma^*$, i.e., $\overline{K} = \{t \in \Sigma^* | \exists s \in K \text{ s.t. } t \leq s\}$. K is said to be prefix-closed if $K = \overline{K}$. The generated (prefix-closed) language of G is defined by $\mathcal{L} = \{s \in \Sigma^* | \delta(q_0, s) \textit{ is defined}\}$. Let $\mathbb{Z}^+$ denote the set of strictly positive integers.

Given a fault f, $\mathcal{L}$ consists of a *non-faulty* language, denoted $\mathcal{H}$, and a *faulty* one, denoted $\mathcal{F}$. The faulty behaviour $\mathcal{F}$ is the set of *faulty* traces characterized by the occurrence of f, i.e., $\mathcal{F} = \{s \in \mathcal{L} | \exists u, v \in \Sigma^* : s = ufv\}$. While $\mathcal{H}$ is the set of *non-faulty* (or *healthy*) traces, i.e., not containing f. We have $\mathcal{L} = \mathcal{H} \cup \mathcal{F}$ and $\mathcal{H} \cap \mathcal{F} = \emptyset$. In the presence of several faults which must be distinguished, we have to do the same study for each fault. Therefore, we consider uniquely a single fault f.

Note that, by construction, $\mathcal{H}$ and $\mathcal{F}$ are regular languages. Moreover, $\mathcal{H}$ is necessarily prefix-closed, because if an execution s is healthy, then its "past" (i.e., the set of prefixes of s) is healthy. We introduce $\mathcal{F}_l$ to denote the set of faulty traces for which the fault f is followed by at least l events, i.e., $\mathcal{F}_l = \{s \in \mathcal{F} : \exists u, v \in \Sigma^*, |v| \geq l, s = ufv\}$.

The detection of the presence (resp., absence) of f will be called *positive* (resp., *negative*) diagnosis. We consider that a diagnoser indicates positive and negative diagnoses using the values 1 and 0, respectively. For the sake of simplicity, we make the following assumptions [Sampath *et al.*, 1995] :

A1 : $\mathcal{L}$ is live, i.e., there is a transition at each state of G.

A2 : Every cycle of G must contain at least one event observable at some site.

Assumption **A1** can be relaxed easily as discussed in [Sampath *et al.*, 1995]. Assumption **A2** ensures that the system will not generate arbitrarily long traces of unobservable events, which would prevent diagnosis within bounded delays.

A diagnoser *Diag* observes the behavior of the plant and takes an effective positive/negative diagnosis decision (or more simply : diagnosis). By *effective diagnosis*, we mean the final diagnosis. (We will see that in decentralized diagnosis some diagnoses are just intermediate results which are used to compute the effective diagnosis.) Let then $Diag(s) \in \{\phi, 0, 1\}$ denote the effective positive/negative diagnosis made after the execution of a trace $s \in \mathcal{L}$. The fact that $Diag(s) = 1$ (resp. 0) means that after the execution of s, the diagnoser has determined that the fault f has (resp., has not) occurred. While $Diag(s) = \phi$ means a "don't care" or "unsure" decision.

In this study we adopt the objectives of [Kumar et Takai, 2006, 2009] defined by Conds. (5.1)-(5.3).

$$\exists l \in \mathbb{Z}^+, \forall s \in \mathcal{F}_l : Diag(s) = 1, \tag{5.1}$$

$$\forall s \in \mathcal{F} : Diag(s) \neq 0, \tag{5.2}$$

$$\forall s \in \mathcal{H} : Diag(s) \neq 1. \tag{5.3}$$

Cond. (5.1) corresponds to the "no missed detection" objective and means that a fault occurrence is followed after a bounded delay (defined by a bounded number l of events) by a positive diagnosis. Conds. (5.2) and (5.3) correspond to the "no false detection" objective of [Kumar et Takai, 2006, 2009] : Cond. (5.2) means that a negative diagnosis (0) is never generated after a fault occurrence, and Cond. (5.3) means that a positive diagnosis (1) is never generated before the first occurrence of a fault. When the three conditions hold, we say that *Diag* satisfies Conds. (5.1)-(5.3) with respect to (or, "w.r.t.") $(\mathcal{F}, \mathcal{H})$. We say that a diagnoser *Diag* "has no missed detection" if it satisfies Cond. (5.1), and *Diag* "has no false detection" if it satisfies Conds. (5.2) (w.r.t. $\mathcal{F}$) and (5.3) (w.r.t. $\mathcal{H}$).

Remark 5.3.1 *It is worth noting that the proposed multi-decision framework can be easily adapted for more ambitious objectives, such as those in [Kumar et Takai, 2009; Qiu et Kumar, 2004; Takai et Kumar, 2006; Wang et al., 2004].*

5.3.2 Decentralized diagnosis principle

In a decentralized diagnosis [Kumar et Takai, 2009; Sampath *et al.*, 1998, 1995], n local diagnosers $(Diag_i)_{i\in I}$ observe the plant and cooperate with each other in order to synthesize a correct diagnosis verdict, where $I = \{1, \ldots, n\}$ is the indexing set of all diagnosers. The local diagnosers then report their diagnoses about fault occurrence; these *local diagnoses* are fused at a *coordinating site* in order to issue a positive or negative *effective diagnosis*. Each $Diag_i$ has a partial view of the plant, that is, its set of observable events is $\Sigma_{o,i} \subseteq \Sigma$. Let $\Sigma_o = \bigcup_{1\leq i\leq n} \Sigma_{o,i}$ and $\Sigma_{uo} = \Sigma \setminus \Sigma_o$. Therefore, an event of Σ_o is observable by at least one diagnoser, and no diagnoser can observe an event of Σ_{uo}. We denote by $P_i : \Sigma^* \to \Sigma^*_{o,i}$ the natural projection that erases the events of $\Sigma \setminus \Sigma_{o,i}$ from any trace $s \in \Sigma^*$. The inverse projection is defined as $P_i^{-1}(s) = \{t \in \Sigma^* : P_i(t) = s\}$. We assume that f is unobservable, i.e., $f \subseteq \Sigma_{uo}$, since an observable fault is trivially diagnosed by just observing it.

Each $Diag_i, i \in I$, is a function $Diag_i : \Sigma^*_{o,i} \to LD$, that associates a local diagnosis $Diag_i(P_i(s)) \in LD$ to every observed trace $P_i(s)$. LD is an arbitrary set of local diagnoses including ϕ. The *effective* diagnosis is obtained by combining the local diagnoses of the n local diagnosers $(Diag_i)_{i\in I}$ using a *fusion operator* $\mathrm{D} : LD^n \to \{\phi, 0, 1\}$, we say that D is defined over LD. The system enclosing the local diagnosers $(Diag_i)_{i\in I}$ and D is called a *decentralized diagnoser*, and denoted by $((Diag_i)_{i\in I}, \mathrm{D})$. The effective diagnosis $Diag(s) \in \{\phi, 0, 1\}$ of a decentralized diagnoser $Diag = ((Diag_i)_{i\in I}, \mathrm{D})$ is computed as follows :

$$Diag(s) = \mathrm{D}((Diag_i(P_i(s)))_{i\in I}). \tag{5.4}$$

Eq. (5.4) means that $Diag(s)$ is computed by applying the operator D to all $Diag_i(P_i(s))$.

Definition 5.3.1 *An architecture obtained using a fusion operator* D *and local diagnosers* $(Diag_i)_{i\in I}$ *is called* D*-architecture, and the corresponding decentralized diagnoser defined by Eq. (5.4) is called* D*-diagnoser.*

For example, we have the conjunctive architecture ($\wedge$-architecture) and the disjunctive architecture ($\vee$-architecture) of [Qiu et Kumar, 2006; Wang *et al.*, 2004, 2007], the conditional conjunctive architecture (COND-$\wedge$-architecture) and the conditional disjunctive architecture (COND-$\vee$-architecture) of [Wang *et al.*, 2005, 2007], and the N-inference architecture (Inf_N-architecture) of [Kumar et Takai, 2006, 2009; Takai et Kumar, 2006].

5.4 Multi-decision decentralized diagnosis : several architectures running in parallel

The multi-decision decentralized diagnosis principle is based on using several (say p) decentralized diagnosers $(Diag^j)_{j\in\{1,\ldots,p\}}$ running in parallel and whose global diagnoses are fused into an *effective diagnosis*. We call this scheme *multi-decision* diagnosis. Hereafter, $J = \{1,\ldots,p\}$. Therefore, each decentralized diagnoser $Diag^j$ achieves its diagnosis according to a given decentralized architecture as shown in Subsection 5.3.2 and Eq. (5.4), but with a superscript j. That is, for every $j \in J$, $Diag^j = ((Diag_i^j)_{i\in I}, \mathrm{D}^j)$ contains n local diagnosers $(Diag_i^j)_{i\in I}$ and a coordinating module D^j. The global diagnosis issued by $Diag^j$, following the execution of $s \in \mathcal{L}$, is computed by Eq. (5.5), which corresponds to Eq. (5.4) with a superscript j added to $Diag$, $Diag_i$ and D :

$$Diag^j(s) = \mathrm{D}^j((Diag_i^j(P_i(s)))_{i\in I}). \tag{5.5}$$

The global diagnoses of all the decentralized diagnosers $(Diag^j)_{j\in J}$ are then fused using a coordinating module $\mathbf{D}$ in order to obtain an effective diagnosis that must satisfy Conds. (5.1)-(5.3). $\mathbf{D}$ is defined as a function $\mathbf{D} : \{\phi, 0, 1\}^p \rightarrow \{\phi, 0, 1\}$. The system enclosing $(Diag_i^j)_{i\in I, j\in J}$, $(\mathrm{D}^j)_{j\in J}$ and $\mathbf{D}$ is called a *multi-decision diagnoser*, and denoted by $((Diag^j)_{j\in J}, \mathbf{D}) = (((Diag_i^j)_{i\in I}, \mathrm{D}^j)_{j\in J}, \mathbf{D})$. The effective diagnosis of a multi-decision diagnoser $Diag = ((Diag^j)_{j\in J}, \mathbf{D})$, $Diag(s) \in \{\phi, 0, 1\}$, is computed as follows :

$$Diag(s) = \mathbf{D}((Diag^j(s))_{j\in J}). \tag{5.6}$$

Eq. (5.6) means that the effective diagnosis $Diag(s)$ is computed by applying the operator $\mathbf{D}$ to all the global diagnoses $Diag^j(s)$.

Definition 5.4.1 *An architecture consisting of coordinating modules* $(\mathrm{D}^j)_{j\in J}$ *and a global coordinating module* $\boldsymbol{D}$ *is called* $\boldsymbol{D}$-$(\mathrm{D}^1,\ldots,\mathrm{D}^p)$ *architecture, and the corresponding multi-decision diagnoser defined by Eqs. (5.5) and (5.6) is called* $\boldsymbol{D}$-$(\mathrm{D}^1,\ldots,\mathrm{D}^p)$*-diagnoser.*

When all the D^j-diagnosers $Diag^j$ (for $j \in J$) correspond to the same decentralized architecture ψ, we obtain a so-called $\mathbf{D}$-ψ^p architecture, while $Diag$ is called $\mathbf{D}$-ψ^p-diagnoser.

Let us for example take $p = 2$, $\mathrm{D}^1 = \vee$, $\mathrm{D}^2 = \textsc{Cond}$-$\wedge$, and $\mathbf{D} = \wedge$. This means that we have a disjunctive (D^1) and a conditionally-conjunctive (D^2) architectures running in parallel and whose global diagnoses are fused conjunctively ($\mathbf{D}$) for obtaining the effective diagnosis. And thus we have a $\wedge$-($\vee$,$\textsc{Cond}$-$\wedge$) architecture and diagnoser.

The multi-decision diagnosis consists therefore in using p D^j-diagnosers $Diag^j$ (for $j \in J$) running in parallel and whose global diagnoses are fused by an operator $\mathbf{D}$ in order to obtain an effective diagnosis. At the present state of our study, we consider two cases of $\mathbf{D}$ when $p > 1$: $\mathbf{D}$ is conjunctive ($\wedge$) or disjunctive ($\vee$). Actually, the terms "conjunctive" and "disjunctive" and their corresponding symbols "$\wedge$" and "$\vee$" are used with an abuse of the language, in the following sense :

By conjunctive, we mean that $Diag(s) = 1$ if all $Diag^j(s) = 1$, $Diag(s) = 0$ if at least one $Diag^j(s) = 0$, and $Diag(s) = \phi$ otherwise.

By disjunctive, we mean that $Diag(s) = 0$ if all $Diag^j(s) = 0$, $Diag(s) = 1$ if at least one $Diag^j(s) = 1$, and $Diag(s) = \phi$ otherwise.

The two cases are based on decomposing the sets $\mathcal{H}$ and $\mathcal{F}$, respectively. They are outlined in the following subsections 5.4.1 and 5.4.2, respectively.

5.4.1 D is conjunctive

When $\mathbf{D}$ is conjunctive, we are thus in the presence of a $\wedge$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$ architecture. The corresponding $\wedge$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-diagnoser consists of a set of p decentralized D^j-diagnosers $Diag^j$ (for $j \in J$) running in parallel and whose global diagnoses are fused *conjunctively*. More precisely, the effective diagnosis $Diag(s)$ is obtained by adapting Eq. (5.6) for $\mathbf{D} = \wedge$, that is,

$$\forall s \in \mathcal{L},\ Diag(s) = \begin{cases} 1 & \text{if, } \forall j \in J, Diag^j(s) = 1, \\ 0 & \text{if, } \exists j \in J, Diag^j(s) = 0, \\ \phi & \text{otherwise.} \end{cases} \tag{5.7}$$

Let us see in more detail how this is realized. We consider a decomposition $(\mathcal{H}^1, \cdots, \mathcal{H}^p)$ of $\mathcal{H}$ such that $\mathcal{H} = \mathcal{H}^1 \cup \cdots \cup \mathcal{H}^p$. Note that we use the word *decomposition* instead of *partition*, because the various $\mathcal{H}^j$ are not necessarily disjoint with each other. We have the following proposition that relates the decomposition of $\mathcal{H}$ and the satisfaction of Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H})$ by a $\wedge$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-diagnoser.

Proposition 5.4.1 *Consider languages $\mathcal{F}$ and $\mathcal{H}$, and D^j-diagnosers $(Diag^j)_{j \in J}$. The $\wedge$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-diagnoser $Diag = ((Diag^j)_{j \in J}, \wedge)$ satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H})$ if and only if there exists a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ such that, $\forall j \in J$, the D^j-diagnoser $Diag^j$ satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H}^j)$.*

The proof of Proposition 5.4.1 is given in the appendix.

5.4.2 D is disjunctive

$\mathbf{D} = \vee$ is dual to $\mathbf{D} = \wedge$, in the sense that we obtain one from the other by switching between conjunction and disjunction, and between decomposing $\mathcal{H}$ and decomposing $\mathcal{F}$. That is, when $\mathbf{D}$ is disjunctive, the effective diagnosis $Diag(s)$ is computed as follows,

$$\forall s \in \mathcal{L},\ Diag(s) = \begin{cases} 0 & \text{if, } \forall j \in J, Diag^j(s) = 0, \\ 1 & \text{if, } \exists j \in J, Diag^j(s) = 1, \\ \phi & \text{otherwise.} \end{cases} \tag{5.8}$$

We consider a decomposition $(\mathcal{F}^1, \ldots, \mathcal{F}^p)$ of $\mathcal{F}$ such that $\mathcal{F} = \mathcal{F}^1 \cup \cdots \cup \mathcal{F}^p$, where the various $\mathcal{F}^j$ are not necessarily disjoint with each other. We have the following proposition that relates the decomposition of $\mathcal{F}$ and the satisfaction of Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H})$ by a $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-diagnoser.

Proposition 5.4.2 *Consider languages $\mathcal{F}$ and $\mathcal{H}$, and D^j-diagnosers $(Diag^j)_{j \in J}$. The $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-diagnoser $Diag = ((Diag^j)_{j \in J}, \vee)$ satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H})$ if and only if there exists a decomposition $\{\mathcal{F}^1, \ldots, \mathcal{F}^p\}$ of $\mathcal{F}$ such that, $\forall j \in J$, the D^j-diagnoser $Diag^j$ satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}^j, \mathcal{H})$.*

The proof of Proposition 5.4.2 is given in the appendix.

5.4.3 Discussion and continuation of the article

$\mathcal{H}$ and $\mathcal{F}$ are *infinite* languages in general, and decomposing an infinite set (e.g., language) into a finite number of subsets satisfying some properties is known to be a challenging problem. A solution to this decomposition problem will be proposed in Section 5.6 in the special case where $\mathbf{D} = \wedge$ and all D^j are inference-based architectures of [Kumar et Takai, 2009]. But note that the proposed solution is easily adaptable for $\mathbf{D} = \vee$, by the following essential modifications : 1) we decompose $\mathcal{F}$ instead of $\mathcal{H}$; and 2) we compute the effective decisions by using Eq. (5.8) instead of Eq. (5.7), that is, we replace the $\wedge$ operator by the $\vee$ operator. For this reason, and for avoiding an excessively long article, in the remaining sections, we will clarify our multi-decision diagnosis in a particular case of Subsection 5.4.1. More precisely, we study the case where $\mathbf{D} = \wedge$ and all D^j have the inference-based architecture of [Kumar et Takai, 2009]. The obtained architecture is denoted $\wedge$-$(Inf_{N_1}, \ldots Inf_{N_p})$, where $p \geq 1$ and each $N_j \geq 0$ is the ambiguity level of D^j, for $j = 1 \cdots p$. When the values of N_j and p are not specified, the architecture is denoted $\wedge$-$Inf^{\geq 1}_{\geq 0}$, where "≥ 1" means "for some $p \geq 1$" and "≥ 0" means "for some

$N_1, \ldots, N_p \geq 0$". Our choice that each of the decentralized diagnosers in parallel has an inference-based architecture, is dictated by the fact that the inference-based architecture is a generalization of all known decidable decentralized architectures [Kumar et Takai, 2009]. Indeed, $\wedge$ and $\vee$ architectures of [Qiu et Kumar, 2006; Wang *et al.*, 2004, 2007] are special cases of inference-based architectures where the maximal ambiguity is $N = 0$, and the conditional architectures of [Wang *et al.*, 2005, 2007] are special cases of inference-based architectures where the maximal ambiguity is $N = 1$. Furthermore, the inference-based ambiguity principle is more suitable when heterogenous decentralized diagnosers are considered, it suffices to consider decentralized diagnosers $Diag^j$ with ambiguity levels N_j, $j \in J$, that are not necessarily the same.

5.5 Multi-decision inference-based architecture

5.5.1 Multi-decision inference-based diagnosers

An inference-based multi-decision diagnoser $Diag = ((Diag^j)_{j \in J}, \wedge)$ consists of p inference-based decentralized diagnosers $(Diag^j)_{j \in J}$ running in parallel whose global diagnoses are fused conjunctively ($\mathbf{D} = \wedge$) for obtaining an effective diagnosis. Each inference-based decentralized diagnoser $Diag^j$ consists of n *inference-based* local diagnosers $(Diag_i^j)_{i \in I}$, and its diagnoses are computed using the results of [Kumar et Takai, 2009] as follows. After the execution of $s \in \mathcal{L}$, each $Diag_i^j$ has observed $P_i(s)$ and issues a local diagnosis $c_i^j(P_i(s))$. An ambiguity level $n_i^j(P_i(s))$ is associated to each local diagnosis, i.e., $LD = \{0, 1, \phi\} \times \mathbb{Z}^+$. Formally :

$$Diag_i^j(P_i(s)) = (c_i^j(P_i(s)), n_i^j(P_i(s))). \tag{5.9}$$

Let $n^j(s)$ denote the minimal ambiguity level of local diagnoses of decentralized diagnoser $Diag^j$, i.e.,

$$n^j(s) = \min_{i \in I} n_i^j(P_i(s)). \tag{5.10}$$

The generic global diagnosis for each $Diag^j$ was defined by Eq. (5.5). In the present case of inference-based diagnoser $Diag^j$, its global diagnosis is computed as follows [Kumar et Takai, 2009] :

$$\forall s \in \mathcal{L}, Diag^j(s) = \begin{cases} 1, & \text{if } \forall i \in I; [n^j(s) = n_i^j(P_i(s)) \Rightarrow c_i^j(P_i(s)) = 1], \\ 0, & \text{if } \forall i \in I; [n^j(s) = n_i^j(P_i(s)) \Rightarrow c_i^j(P_i(s)) = 0], \\ \phi, & otherwise. \end{cases} \tag{5.11}$$

Definition 5.5.1 *An inference-based decentralized diagnoser (i.e., defined by Eqs. (5.9)-(5.11)) $Diag^j$ is said N_j-*inferring*, and denoted Inf_{N_j}-diagnoser, if :*

1. $\forall s \in \mathcal{L} : Diag^j(s) \neq \phi \Rightarrow n^j(s) \leq N_j$, *and*
2. $\forall s, s' \in \mathcal{L} : [Diag^j(s) \neq \phi \wedge n^j(s') \leq n^j(s)] \Rightarrow Diag^j(s') \neq \phi$.

An inference-based multi-decision diagnoser $Diag = ((Diag^j)_{j\in J}, \wedge)$ consisting of p Inf_{N_j}-diagnosers $(Diag^j)_{j\in J}$, is denoted $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-diagnoser.

Note that N_j is the maximal ambiguity level that is used in a Inf_{N_j}-diagnoser. The effective diagnosis $Diag(s)$ of the $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-diagnoser is computed by combining conjunctively the global diagnoses $(Diag^j(s))_{j\in J}$ of Eq. (5.11) using Eq. (5.7).

5.5.2 Decomposing $\mathcal{H}$ and computing the local diagnoses $c_i^j(P_i(s))$ and $n_i^j(P_i(s))$

As already explained in Subsection 5.4.1, we will use a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$. This decomposition is a practical rule for computing the inference-based local diagnoses $Diag_i^j(P_i(s))$ of Eq. (5.9), i.e., $c_i^j(P_i(s))$ and $n_i^j(P_i(s))$.

Given a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$, for each $j \in J$ we apply to $(\mathcal{F}, \mathcal{H}^j)$ the framework of [Kumar et Takai, 2009] as follows. For each $j \in J$, we define inductively a monotonically decreasing sequence of language pairs $(\mathcal{F}^j[k], \mathcal{H}^j[k])$ as follows :

$$\mathcal{F}^j[0] = \mathcal{F}, \mathcal{H}^j[0] = \mathcal{H}^j,$$
$$\forall k \geq 0 : \mathcal{F}^j[k+1] = \mathcal{F}^j[k] \cap \bigcap_{i\in I} P_i^{-1}P_i(\mathcal{H}^j[k]), \mathcal{H}^j[k+1] = \mathcal{H}^j[k] \cap \bigcap_{i\in I} P_i^{-1}P_i(\mathcal{F}^j[k]). \tag{5.12}$$

For simplicity, j is omitted when $p = 1$ (i.e., no decomposition). The global diagnoses of every $Diag^j$ are defined by Eqs. (5.9)-(5.11) from $(c_i^j(P_i(s)), n_i^j(P_i(s)))_{i\in I}$. Now, we use a method of [Kumar et Takai, 2009] to compute $(c_i^j(P_i(s)), n_i^j(P_i(s)))_{i\in I, j\in J}$ which guarantees that every $Diag^j$ is N_j-inferring. For each $j \in J$, using the sequence $(\mathcal{F}^j[k], \mathcal{H}^j[k])_{0\leq k\leq N_j}$, every local diagnoser $Diag_i^j$ computes, for every $s \in \mathcal{L}$:

$$n_i^{f,j}(P_i(s)) = \min\{k \in \mathbb{Z}^+ \mid [P_i(s) \notin P_i(\mathcal{H}^j[k])] \vee [k = N_j + 1]\}, \tag{5.13a}$$
$$n_i^{h,j}(P_i(s)) = \min\{k \in \mathbb{Z}^+ \mid [P_i(s) \notin P_i(\mathcal{F}^j[k])] \vee [k = N_j + 1]\}. \tag{5.13b}$$

A local diagnosis is issued by comparing the two ambiguity levels, $n_i^{f,j}(P_i(s))$ and $n_i^{h,j}(P_i(s))$, giving preference to the smallest one. This is formalized as follows :

$$c_i^j(P_i(s)) = \begin{cases} 0, & \text{if } n_i^{h,j}(P_i(s)) < n_i^{f,j}(P_i(s)), \\ 1, & \text{if } n_i^{f,j}(P_i(s)) < n_i^{h,j}(P_i(s)), \\ \phi, & \text{if } n_i^{f,j}(P_i(s)) = n_i^{h,j}(P_i(s)), \end{cases} \tag{5.14}$$

and

$$n_i^j(P_i(s)) = \min\{n_i^{h,j}(P_i(s)), n_i^{f,j}(P_i(s))\}. \tag{5.15}$$

Note that we adapted the equations of [Kumar et Takai, 2009] with an index or superscript j, since every $Diag^j$ is a N_j-inference diagnoser. See [Kumar et Takai, 2009] for more detail on the inference-based framework.

Remark 5.5.1 *An* inference-based diagnoser *is by definition a decentralized diagnoser that satisfies Eqs. (5.9)-(5.11). Thus, "an inference-based diagnoser satisfying Eqs. (5.13)-(5.15)" means "a decentralized diagnoser satisfying Eqs. (5.9)-(5.15)". But for the sake of clarity, we have opted to always indicate "Eqs. (5.9)-(5.15)" instead of "Eqs. (5.13)-(5.15)", even if we mention "inference-based diagnoser".*

We have the following lemma which is an an adaptation of Lemma 2 of [Kumar et Takai, 2009] by considering $\mathcal{H}^j$ instead of $\mathcal{H}$ (see [Kumar et Takai, 2009] for the proof).

Lemma 5.5.1 *[Kumar et Takai, 2009] Given a subset $\mathcal{H}^j \subseteq \mathcal{H}$, the inference-based decentralized diagnoser defined by Eqs. (5.9)-(5.15) w.r.t $\mathcal{H}^j$ is an Inf_{N_j}-diagnoser (see Definition 5.5.1).*

We have the following proposition which is a straightforward consequence of Definition 5.5.1 and Lemma 5.5.1 (and thus, the proof is omitted).

Proposition 5.5.1 *Given a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$, the multi-decision diagnoser defined by Eqs. (5.9)-(5.15) and (5.7) is a $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-diagnoser.*

Example 5.5.1 *Figure 5.1 presents a plant G such that $\Sigma_{o,1} = \{a_1, b_1, d_1, e\}$ and $\Sigma_{o,2} = \{a_2, b_2, d_2, c_2, e\}$. We have $\mathcal{H} = \{\overline{a_1a_2e^*}, \overline{b_1b_2e^*}, \overline{d_1d_2e^*}, \overline{d_1c_2e^*}\}$ and $\mathcal{F} = \{a_1f\overline{b_2e^*}, b_1f\overline{d_2e^*}, b_1f\overline{c_2e^*}, d_1f\overline{a_2e^*}\}$. Let us consider the cases $p = 1$ and $p = 2$.*

***Case** $p = 1$: Since j takes the single value 1, it will be implicit in all equations. We have $\mathcal{F}[0] = \mathcal{F}$ and $\mathcal{H}[0] = \mathcal{H}$, and there is no Inf_N-diagnoser satisfying Conds. (5.1)-(5.3), $\forall N \in \mathbb{Z}^+$. To show it, let us compute the languages pairs $(\mathcal{F}[k], \mathcal{H}[k])_{k\geq 0}$.*

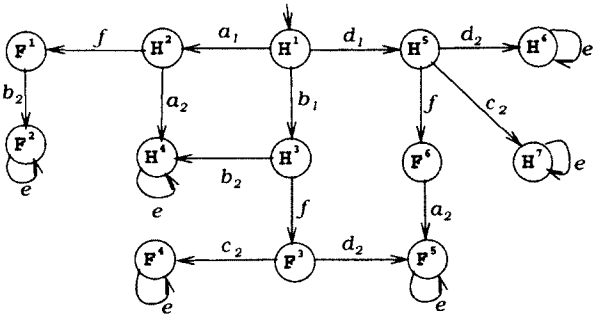

Figure 5 1 Plant $\mathcal{L}$

First step $(k = 1)$: $\mathcal{H}[1] = \{a_1\overline{a_2e^*}, b_1\overline{b_2e^*}, d_1\overline{d_2e^*}, d_1\overline{c_2e^*}\}$ *and* $\mathcal{F}[1] = \mathcal{F}[0]$.

Second step $(k = 2)$: $\mathcal{H}[2] = \mathcal{H}[1]$ *and* $\mathcal{F}[2] = \mathcal{F}[1] = \mathcal{F}[0]$.

Therefore, we obtain . $\forall k \geq 1$, $\mathcal{H}[k] = \{a_1\overline{a_2e^*}, b_1\overline{b_2e^*}, d_1\overline{d_2e^*}, d_1\overline{c_2e^*}\}$, *and,* $\forall k \geq 0$, $\mathcal{F}[k] = \{a_1f\overline{b_2e^*}, b_1f\overline{d_2e^*}, b_1f\overline{c_2e^*}, d_1f\overline{a_2e^*}\}$.

We consider the trace $a_1fb_2e^z \in \mathcal{F}[k]$, *and thus, we have,* $\forall i = 1, 2$, $\forall k \geq 0$, $P_i(a_1fb_2e^z) \in P_i(\mathcal{H}[k])$. *And from Eq. (5.13),* $n_i^f(P_i(a_1fb_2e^z)) = N + 1$, $\forall N \in \mathbb{Z}^+$. *Moreover,* $\forall k \geq 0$, $\forall i = 1, 2$, $P_i(a_1fb_2e^z) \in P_i(\mathcal{F}[k])$ *And from Eq. (5.13),* $n_i^h(P_i(a_1fb_2e^z)) = N + 1$, $\forall N \in \mathbb{Z}^+$. *Since,* $\forall N \in \mathbb{Z}^+$, $n_i^h(P_i(a_1fb_2e^z)) = n_i^f(P_i(a_1fb_2e^z)) = N+1$, *we deduce by Eq. (5.14) that,* $\forall i = 1, 2$, $c_i(P_i(a_1fb_2e^z)) = \phi$. *Then, by Eq. (5.11),* $\forall z \in \mathbb{Z}^+$, $Diag(a_1fb_2e^z) = \phi$, *which violates Cond. (5.1), and thus, there is no* Inf_N*-diagnoser that can make the right diagnosis, i.e., that can respect Conds. (5.1)-(5.3).*

Case $p = 2$: *Consider the decomposition* $\{\mathcal{H}^1, \mathcal{H}^2\}$ *of* $\mathcal{H}$ *where* $\mathcal{H}^1 = \{\overline{a_1a_2e^*}, \overline{b_1b_2e^*}, d_1\}$ *and* $\mathcal{H}^2 = \{d_1d_2e^*, d_1c_2e^*\}$. $(\mathcal{F}^j[k], \mathcal{H}^j[k])_{k\geq 0}$ *are then computed as follows :*

For $j = 1$:

First step $(k = 1)$: $\mathcal{F}^1[1] = \{a_1f\overline{b_2e^*}, d_1f\overline{a_2}, b_1f\}$ *and* $\mathcal{H}^1[1] = \{a_1\overline{a_2e^*}, b_1\overline{b_2e^*}, d_1\}$.

Second step $(k = 2)$: $\mathcal{F}^1[2] = \{a_1f\overline{b_2e^*}, b_1f, d_1f\overline{a_2}\}$ *and* $\mathcal{H}^1[2] = \{a_1\overline{a_2}, d_1, b_1\overline{b_2}\}$.

Third step $(k = 3)$: $\mathcal{F}^1[3] = \{a_1f\overline{b_2}, b_1f, d_1f\overline{a_2}\}$ *and* $\mathcal{H}^1[3] = \emptyset$.

For $j = 2$:

First step $(k = 1)$: $\mathcal{F}^2[1] = \emptyset$ *and* $\mathcal{H}^2[1] = \{d_1d_2e^*, d_1c_2e^*\}$.

Using Eqs. (5.9)-(5.15), Table 5.1 (resp., Table 5.2) presents the local and global diagnoses taken by the Inf_2*-diagnoser* $Diag^1$ *(resp.,* Inf_0*-diagnoser* $Diag^2$*). Note that, for* $j \in \{1, 2\}$, *the expressions* $n_1^{h,j}$, $n_1^{f,j}$ *and* $Diag_1^j$ *(resp.,* $n_2^{h,j}$, $n_2^{f,j}$ *and* $Diag_2^j$*) are computed for* $P_1(s)$ *(resp.,* $P_2(s)$*) and* $Diag^j$ *is computed for* s. *Using Eq. (5.7), Table 5.3 presents the effective diagnoses taken by the* $\wedge$-(Inf_2, Inf_0)*-diagnoser that combines conjunctively the diagnoses of* $Diag^1$ *and* $Diag^2$ *We see in Table 5.3 that* $Diag(s) \neq 1$ *for all traces* $s \in \mathcal{H}$ *(thus, Cond. (5 3)) and* $Diag(s) \neq 0$ *for all traces* $s \in \mathcal{F}$ *(thus, Cond. (5.2)), and* $Diag(s) = 1$

for all traces $s \in \mathcal{F}_2$ (thus, Cond. (5.1)). We will explain in Section 5.6 why we have selected exactly the Inf_2-diagnoser $Diag^1$ and the Inf_0-diagnoser $Diag^2$, that is, $N_1 = 2$ and $N_2 = 0$.

Trace executed	$n_1^{h,1}$	$n_1^{f,1}$	$Diag_1^1$	$n_2^{h,1}$	$n_2^{f,1}$	$Diag_2^1$	$Diag^1$
$s = \varepsilon$	0	1	(0,0)	3	3	(ϕ,3)	0
$s \in a_1\overline{f}$	3	3	(ϕ,3)	3	3	(ϕ,3)	ϕ
$s \in b_1\overline{f}$	3	3	(ϕ,3)	3	3	(ϕ,3)	ϕ
$s \in d_1\overline{f}$	3	3	(ϕ,3)	3	3	(ϕ,3)	ϕ
$s = a_1a_2$	3	3	(ϕ,3)	3	3	(ϕ,3)	ϕ
$s = b_1b_2$	3	3	(ϕ,3)	3	3	(ϕ,3)	ϕ
$s = d_1d_2$	3	3	(ϕ,3)	1	0	(1,0)	1
$s = d_1c_2$	3	3	(ϕ,3)	1	0	(1,0)	1
$s = a_1fb_2$	3	3	(ϕ,3)	3	3	(ϕ,3)	ϕ
$s = b_1fd_2$	3	3	(ϕ,3)	1	0	(1,0)	1
$s = b_1fc_2$	3	3	(ϕ,3)	1	0	(1,0)	1
$s = d_1fa_2$	3	3	(ϕ,3)	3	3	(ϕ,3)	ϕ
$s \in a_1a_2e^+$	3	2	(1,2)	1	2	(0,1)	0
$s \in b_1b_2e^+$	1	2	(0,1)	3	2	(1,2)	0
$s = d_1d_2e^+$	1	0	(1,0)	1	0	(1,0)	1
$s = d_1c_2e^+$	1	0	(1,0)	1	0	(1,0)	1
$s \in a_1fb_2e^+$	3	2	(1,2)	3	2	(1,2)	1
$s \in b_1fd_2e^+$	1	2	(0,1)	1	0	(1,0)	1
$s = b_1fc_2e^+$	1	2	(0,1)	1	0	(1,0)	1
$s \in d_1fa_2e^+$	1	0	(1,0)	1	2	(0,1)	1

Tableau 5.1 Local and global diagnoses taken by the Inf_2-diagnoser $Diag^1$ computed by Eqs. (5.9)-(5.15) for the plant of Fig. 5.1, w.r.t. $(\mathcal{F}, \mathcal{H}^1)$, for $\mathcal{H}^1 = \{\overline{a_1a_2e^*}, \overline{b_1b_2e^*}, d_1\}$.

Trace executed	$n_1^{h,2}$	$n_1^{f,2}$	$Diag_1^2$	$n_2^{h,2}$	$n_2^{f,2}$	$Diag_2^2$	$Diag^2$
$s = \varepsilon$	0	1	(0,0)	1	0	(1,0)	ϕ
$s \in a_1\overline{f}$	1	0	(1,0)	1	0	(1,0)	1
$s \in b_1\overline{f}$	1	0	(1,0)	1	0	(1,0)	1
$s \in d_1\overline{f}$	1	1	(ϕ,1)	1	0	(1,0)	1
$s = a_1a_2$	1	0	(1,0)	1	0	(1,0)	1
$s = b_1b_2$	1	0	(1,0)	1	0	(1,0)	1
$s = d_1d_2$	1	1	(ϕ,1)	1	1	(ϕ,1)	ϕ
$s = d_1c_2$	1	1	(ϕ,1)	1	1	(ϕ,1)	ϕ
$s = a_1fb_2$	1	0	(1,0)	1	0	(1,0)	1
$s = b_1fd_2$	1	0	(1,0)	1	1	(ϕ,1)	1
$s = b_1fc_2$	1	0	(1,0)	1	1	(ϕ,1)	1
$s = d_1fa_2$	1	1	(ϕ,1)	1	0	(1,0)	1
$s \in a_1a_2e^+$	1	0	(1,0)	1	0	(1,0)	1
$s \in b_1b_2e^+$	1	0	(1,0)	1	0	(1,0)	1
$s \in d_1d_2e^+$	1	1	(ϕ,1)	1	1	(ϕ,1)	ϕ
$s \in d_1c_2e^+$	1	1	(ϕ,1)	1	1	(ϕ,1)	ϕ
$s \in a_1fb_2e^+$	1	0	(1,0)	1	0	(1,0)	1
$s \in b_1fd_2e^+$	1	0	(1,0)	1	1	(ϕ,1)	1
$s \in b_1fc_2e^+$	1	0	(1,0)	1	1	(ϕ,1)	1
$s \in d_1fa_2e^+$	1	1	(ϕ,1)	1	0	(1,0)	1

Tableau 5.2 Local and global diagnoses taken by the Inf_0-diagnoser $Diag^2$ computed by Eqs. (5.9)-(5.15) for the plant of Fig. 5.1, w.r.t. $(\mathcal{F}, \mathcal{H}^2)$, for $\mathcal{H}^2 = \{d_1d_2e^*, d_1c_2e^*\}$.

Trace executed	$Diag^1(s)$	$Diag^2(s)$	$Diag(s)$
$s = \varepsilon$	0	ϕ	0
$s \in a_1\overline{f}$	ϕ	1	ϕ
$s \in b_1\overline{f}$	ϕ	1	ϕ
$s \in d_1\overline{f}$	ϕ	1	ϕ
$s = a_1a_2$	ϕ	1	ϕ
$s = b_1b_2$	ϕ	1	ϕ
$s = d_1d_2$	1	ϕ	ϕ
$s = d_1c_2$	1	ϕ	ϕ
$s = a_1fb_2$	ϕ	1	ϕ
$s = b_1fd_2$	1	1	1
$s = b_1fc_2$	1	1	1
$s = d_1fa_2$	ϕ	1	ϕ
$s \in a_1a_2e^+$	0	1	0
$s \in b_1b_2e^+$	0	1	0
$s \in d_1d_2e^+$	1	ϕ	ϕ
$s \in d_1c_2e^+$	1	ϕ	ϕ
$s \in a_1fb_2e^+$	1	1	1
$s \in b_1fd_2e^+$	1	1	1
$s \in b_1fc_2e^+$	1	1	1
$s \in d_1fa_2e^+$	1	1	1

Tableau 5.3 Global and effective diagnoses taken by the $\wedge - (Inf_2, Inf_0)$-diagnoser computed by applying Eq. (5.7) to the diagnoses $Diag^1(s)$ and $Diag^2(s)$ of Tables 5.1 and 5.2.

5.6 Existence of solutions for the $\wedge$-$Inf^{\geq 1}_{\leq N}$ architecture

In this section, we characterize the class of languages for which the fault can be detected in bounded delay by a $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-diagnoser. For that, we have the following lemma which is an adaptation of Lemma 3 of [Kumar et Takai, 2009] by considering $\mathcal{H}^j$ instead of $\mathcal{H}$ (see [Kumar et Takai, 2009] for the proof).

Lemma 5.6.1 *[Kumar et Takai, 2009] Given a subset $\mathcal{H}^j \subseteq \mathcal{H}$ and an integer N_j, the Inf_{N_j}-diagnoser defined by Eqs. (5.9)-(5.15) satisfies Conds. (5.2) and (5.3) w.r.t. $(\mathcal{F}, \mathcal{H}^j)$.*

The following proposition shows that no false detection is made by the multi-decision decentralized diagnoser defined by Eqs. (5.9)-(5.15) and (5.7) for a given decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$.

Proposition 5.6.1 *Given a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ and integers $N_1, \ldots, N_p$, the $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-diagnoser defined by Eqs. (5.9)-(5.15) and (5.7) satisfies Conds. (5.2) and (5.3) w.r.t. $(\mathcal{F}, \mathcal{H})$.*

The proof of Proposition 5.6.1 is given in the appendix.

In order to define the class of diagnosers that have no missed detection (i.e., satisfy Cond. (5.1)) w.r.t. $\mathcal{F}$, we present in the next definition the notion of N-inference F-codiagnosability (or Inf_N-F-CODIAG) as introduced in [Kumar et Takai, 2009]. Definition 5.6.1 is an adaptation of Definition 2 of [Kumar et Takai, 2009], where we consider subsets $\mathcal{H}^j$ instead of $\mathcal{H}$.

Definition 5.6.1 *[Kumar et Takai, 2009] Consider a decomposition $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ of $\mathcal{H}$, and integers $N_1, \ldots, N_p$. For every $j \in J$, $(\mathcal{F}, \mathcal{H}^j)$ is said Inf_{N_j}-F-codiagnosable (or Inf_{N_j}-F-CODIAG) if there exists $l_j \in \mathbb{Z}^+$ such that $\mathcal{F}^j[N_j + 1] \cap \mathcal{F}_{l_j} = \emptyset$.*

By using the notion of Inf_N-F-CODIAG defined in Definition 5.6.1, we define the notions of $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-F-CODIAG, $\wedge$-$Inf^{\geq 1}_{\leq N}$ and $\wedge$-$Inf^{\geq 1}_{\geq 0}$.

Definition 5.6.2 *Given $N_1, \ldots, N_p \in \mathbb{Z}^+$, $(\mathcal{F}, \mathcal{H})$ is said $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-F-CODIAG w.r.t. a decomposition $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ of $\mathcal{H}$ if, $\forall j \in J$, $(\mathcal{F}, \mathcal{H}^j)$ is Inf_{N_j}-F-CODIAG. $(\mathcal{F}, \mathcal{H})$ is said $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-F-CODIAG if there exists a decomposition $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ of $\mathcal{H}$ such that $(\mathcal{F}, \mathcal{H})$ is $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-F-CODIAG w.r.t. $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$.*

Definition 5.6.3 *Given an integer N, $(\mathcal{F}, \mathcal{H})$ is said $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG if it is $\wedge$-($Inf_{N_1}, \cdots, Inf_{N_p}$)-F-CODIAG for some $p \geq 1$ and $N1, \ldots, N_p \leq N$. $(\mathcal{F}, \mathcal{H})$ is said $\wedge$-$Inf^{\geq 1}_{\geq 0}$-F-CODIAG if it is $\wedge$-$Inf^{\geq 1}_{\leq M}$-F-CODIAG for some unspecified $M \in \mathbb{Z}^+$.*

We will explain in Section 5.8 that $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG is more useful than $\wedge$-$Inf_{\geq 0}^{\geq 1}$-F-CODIAG.

We have seen in Lemma 5.6.1 that the Inf_{N_j}-diagnoser defined by Eqs. (5.9)-(5.15) satisfies Conds. (5.2)-(5.3) (i.e., no false detection) w.r.t. $(\mathcal{F}, \mathcal{H}^j)$. The following Lemma 5.6.2 states this Inf_{N_j}-diagnoser satisfies also Cond. (5.1) (i.e., no missed detection) w.r.t. $\mathcal{F}$, if $(\mathcal{F}, \mathcal{H}^j)$ is Inf_{N_j}-F-CODIAG. This lemma is an adaptation of Lemma 4 of [Kumar et Takai, 2009] by considering $\mathcal{H}^j$ instead of $\mathcal{H}$ (see [Kumar et Takai, 2009] for the proof).

Lemma 5.6.2 *[Kumar et Takai, 2009] Given a subset $\mathcal{H}^j \subseteq \mathcal{H}$ and an integer N_j, if $(\mathcal{F}, \mathcal{H}^j)$ is Inf_{N_j}-F-CODIAG then the Inf_{N_j}-diagnoser given by Eqs. (5.9)-(5.15) satisfies Cond. (5.1) w.r.t. $\mathcal{F}$.*

We have seen in Proposition 5.6.1 that the $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-diagnoser defined by Eqs. (5.9)-(5.15) and (5.7) satisfies Conds. (5.2)-(5.3) (i.e., no false detection) w.r.t. $(\mathcal{F}, \mathcal{H})$. The following Proposition 5.6.2 states that this $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-diagnoser satisfies also Cond. (5.1) (i.e., no missed detection) w.r.t. $\mathcal{F}$, if $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG

Proposition 5.6.2 *Given a decomposition $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ of $\mathcal{H}$, if $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG w.r.t. $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$, then the $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-diagnoser given by Eqs. (5.9)-(5.15) and (5.7) satisfies Cond. (5.1) w.r.t. $\mathcal{F}$.*

The proof of Proposition 5.6.2 is given in the appendix.

The following Theorem 5.6.1 states a necessary and sufficient condition under which there exists a Inf_{N_j}-diagnoser that makes the right diagnoses for $(\mathcal{F}, \mathcal{H}^j)$. This theorem is an adaptation of Theorem 1 of [Kumar et Takai, 2009] by considering $\mathcal{H}^j$ instead of $\mathcal{H}$ (see [Kumar et Takai, 2009] for the proof).

Theorem 5.6.1 *[Kumar et Takai, 2009] Consider a subset $\mathcal{H}^j \subseteq \mathcal{H}$ and an integer N_j. $(\mathcal{F}, \mathcal{H}^j)$ is Inf_{N_j}-F-CODIAG if and only if there exists a Inf_{N_j}-diagnoser $Diag^j$ that satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H}^j)$.*

We state in the next theorem a necessary and sufficient condition under which there exists a $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-diagnoser that makes the right diagnoses for $(\mathcal{F}, \mathcal{H})$.

Theorem 5.6.2 *$(\mathcal{F}, \mathcal{H})$ is $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG if and only if there exists a $\wedge$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-diagnoser $Diag = ((Diag^j)_{j \in J}, \wedge)$ that satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H})$.*

The proof of Theorem 5.6.2 is given in the appendix.

From Theorem 5 6.2, determining the existence of a $\wedge$-($Inf_{N_1}, \cdots, Inf_{N_p}$)-diagnoser satisfying Conds. (5.1)-(5 3), for some integers $p \geq 1$ and $N_1, \ldots, N_p$, necessitates to determine whether $(\mathcal{F}, \mathcal{H})$ is $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-F-CODIAG. Hence, according to Def. 5.6.2, we have to answer the following question :

Question 1 : Does there exist a decomposition $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ of $\mathcal{H}$ and some $N_1, \ldots, N_p$ such that, $\forall j \in J$, $(\mathcal{F}, \mathcal{H}^j)$ is Inf_{N_j}-F-CODIAG, i.e., $\mathcal{F}^j[N_j+1] \cap \mathcal{F}_{l_j} = \emptyset$ for some $l_j \in \mathbb{Z}^+$?

What makes Question 1 difficult is that $\mathcal{H}$ is in general infinite and decomposing infinite languages is known to be a challenging problem. We propose here a solution that transforms the problem of decomposing an infinite regular language into a problem of decomposing a *finite* set of states in a FSA Hence, our solution for decomposing $\mathcal{H}$ is based on the use of a FSA $\mathcal{A}_{\mathcal{H}}$ accepting $\mathcal{H}$, that is, every trace $s \in \mathcal{H}$ leads to a marked state of $\mathcal{A}_{\mathcal{H}}$. We consider uniquely the decompositions satisfying the following assumption :

A3 : Given a FSA $\mathcal{A}_{\mathcal{H}}$ accepting $\mathcal{H}$, the only eligible decompositions $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ of $\mathcal{H}$ are such that every $\mathcal{H}^j$ consists of (all and only) the traces leading to one or several marked states of $\mathcal{A}_{\mathcal{H}}$. In other words, every $\mathcal{H}^j$ corresponds to a subset of the set of marked states of $\mathcal{A}_{\mathcal{H}}$.

With Assumption **A3**, we have transformed the problem of decomposing an *infinite* regular language into a problem of decomposing the *finite* state set of a FSA.

Remark 5.6.1 *Assumption **A3** is less restrictive than the assumption used in [Chakib et Khoumsi, 2009; Khoumsi et Chakib, 2008a], in the sense that the latter permits less decompositions. This is due to the fact that Assumption **A3** can be based on a nondeterministic FSA, while the assumption used in [Chakib et Khoumsi, 2009; Khoumsi et Chakib, 2008a] is expressed in a form which implies that the associated FSA is necessarily deterministic.*

The following Definitions 5.6.4 and 5.6.5 are adaptations of Definitions 5.6.2 and 5.6.3 for decompositions satisfying Assumption **A3** w.r.t. a specific FSA $\mathcal{A}_{\mathcal{H}}$ accepting $\mathcal{H}$.

Definition 5.6.4 *Consider integers $N_1, \ldots, N_p$, and a FSA $\mathcal{A}_{\mathcal{H}}$ accepting $\mathcal{H}$. $(\mathcal{F}, \mathcal{H})$ is said $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$, if there exists a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ satisfying Assumption **A3** w.r.t. $\mathcal{A}_{\mathcal{H}}$ such that, $(\mathcal{F}, \mathcal{H})$ is $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-CODIAG w.r.t. $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ (see Definition 5.6.2).*

Definition 5.6.5 *Consider an integer N, and a FSA $\mathcal{A}_{\mathcal{H}}$ accepting $\mathcal{H}$. $(\mathcal{F}, \mathcal{H})$ is said $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$ if it is $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-F-CODIAG w.r.t $\mathcal{A}_{\mathcal{H}}$ for some $p \geq 1$ and some $N1, \ldots, N_p \leq N$. $(\mathcal{F}, \mathcal{H})$ is said $\wedge$-$Inf^{\geq 1}_{\geq 0}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$ if it is $\wedge$-$Inf^{\geq 1}_{\leq M}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$ for some unspecified $M \in \mathbb{Z}^+$.*

The following corollary is a straightforward result obtained from Theorem 5.6.2 (and thus, its proof is omitted).

Corollary 5.6.1 *Given a FSA $\mathcal{A}_{\mathcal{H}}$ accepting $\mathcal{H}$, if $(\mathcal{F},\mathcal{H})$ is $\wedge$-$(Inf_{N_1},\ldots,Inf_{N_p})$-F-*CODIAG *w.r.t. $\mathcal{A}_{\mathcal{H}}$, then there exists a $\wedge$-$(Inf_{N_1},\ldots,Inf_{N_p})$-diagnoser Diag that satisfies Conds, (5.1)-(5.3) w.r.t. $(\mathcal{F},\mathcal{H})$.*

Remark 5.6.2 *If we compare Theorem 5.6.2 and Corollary 5.6.1, we see that the latter contains a* sufficient *condition while the former contains a* necessary and sufficient *condition. This is due to the fact that in Corollary 5.6.1, we have used Assumption **A3**, which is not actually necessary for the existence of $\wedge$-$(Inf_{N_1},\ldots,Inf_{N_p})$-F-diagnoser. Assumption **A3** has been added for solving the problem of decomposing infinite languages.*

Let us return to the example 5.5.1. From Def. 5.6.1, we have that $(\mathcal{F},\mathcal{H})$ is *not* Inf_N-F-CODIAG, because $\mathcal{F}[k] = \{a_1 f\overline{b_2 e^*}, b_1 f\overline{d_2 e^*}, d_1 f\overline{a_2 e^*}\}$, $\forall k \geq 1$, and thus, $\forall l,k \in \mathbb{Z}^+$, $\mathcal{F}_l \cap \mathcal{F}[k] \neq \emptyset$. From Theorem 5.6.1 (applied for $p = 1$), we deduce that there exists no Inf_N-diagnoser that can make the right diagnosis, i.e., that can respect Conds. (5.1)-(5.3). Note that this result was observed in the case $p = 1$ of Example 5.5.1 of Subsection 5.5.2. Consider now an automaton $\mathcal{A}_{\mathcal{H}}$ accepting $\mathcal{H}$, which is obtained from Fig. 5.1 by removing states F^i, $i = 1,\ldots,6$, and marking the remaining states. We use the decomposition $\{\mathcal{H}^1,\mathcal{H}^2\}$ of $\mathcal{H}$ where $\mathcal{H}^1 = \{\overline{a_1 a_2 e^*}, \overline{b_1 b_2 e^*}, d_1\}$ and $\mathcal{H}^2 = \{d_1 d_2 e^*, d_1 c_2 e^*\}$. This decomposition satisfies Assumption **A3** w.r.t. $\mathcal{A}_{\mathcal{H}}$. We compute $\mathcal{F}_2 \cap \mathcal{F}^1[3] = \emptyset$ and $\mathcal{F}_2 \cap \mathcal{F}^2[1] = \emptyset$. From Definitions 5.6.1 and 5.6.2, $(\mathcal{F},\mathcal{H})$ is $\wedge$-(Inf_2,Inf_0)-F-CODIAG. From Theorem 5.6.2 (or Corollary 5.6.1), we deduce that there exists a $\wedge$-(Inf_2,Inf_0)-diagnoser that can make the right diagnosis. Such a $\wedge$-(Inf_2,Inf_0)-diagnoser was in fact computed in the case $p = 2$ of Example 5.5.1 of Section 5.5.2, and was represented in Tables 5.1, 5.2 and 5.3.

5.7 Some properties related to the $\wedge$-$Inf_{\geq 0}^{\geq 1}$ architecture

The following lemma states the effect of inclusion between subsets of $\mathcal{H}^j$ on the Inf_N-F-codiagnosability.

Lemma 5.7.1 *Consider a decomposition $\{\mathcal{H}^1,\ldots,\mathcal{H}^p\}$ of $\mathcal{H}$. For every $\nu_1,\nu_2 \in J$ such that $\mathcal{H}^{\nu_1} \subseteq \mathcal{H}^{\nu_2}$, if $(\mathcal{F},\mathcal{H}^{\nu_2})$ is Inf_N-F-*CODIAG *then $(\mathcal{F},\mathcal{H}^{\nu_1})$ is Inf_N-F-*CODIAG.

The proof of Lemma 5.7.1 is given in the appendix.

Intuitively, if we consider a decomposition $\{\mathcal{H}^1,\ldots,\mathcal{H}^p\}$ of $\mathcal{H}$ and some $N \geq 0$ such that all $(\mathcal{F},\mathcal{H}^j)$ are Inf_N-F-CODIAG, then for every other decomposition $\{\mathcal{J}^1,\ldots,\mathcal{J}^q\}$

of $\mathcal{H}$ such that every $\mathcal{J}^l$ is a subset of some $\mathcal{H}^j$, we have that all $(\mathcal{F}, \mathcal{J}_l)$ are Inf_N-F-CODIAG. Furthermore, consider a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ which may contain some "incorrect" $\mathcal{H}^j$, i.e., such that $(\mathcal{F}, \mathcal{H}^j)$ is not Inf_N-F-CODIAG. By decomposing every "incorrect" $\mathcal{H}^j$, we might obtain a good decomposition, i.e., a decomposition for which each $(\mathcal{F}, \mathcal{H}^j)$ is Inf_N-F-CODIAG.

The following proposition states necessary and sufficient conditions for $(\mathcal{F}, \mathcal{H})$ to be $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG or $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG w.r.t. a FSA. For that, we denote by $\mathcal{L}(\mathcal{A}_{\mathcal{H}}, x)$ the set of healthy traces reaching the state x of a FSA $\mathcal{A}_{\mathcal{H}}$ accepting $\mathcal{H}$.

Proposition 5.7.1 *Consider an integer N and a FSA $\mathcal{A}_{\mathcal{H}}$ accepting $\mathcal{H}$. $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG (resp., $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG) w.r.t. $\mathcal{A}_{\mathcal{H}}$ iff, for every marked state x of $\mathcal{A}_{\mathcal{H}}$, $(\mathcal{F}, \mathcal{L}(\mathcal{A}_{\mathcal{H}}, x))$ is Inf_{N_x}-F-CODIAG for some $N_x \leq N$ (resp., ≥ 0).*

The proof of Proposition 5.7.1 is given in the appendix.

Consider the particular decomposition (in fact, a partition) $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ where each $\mathcal{H}^j$ corresponds to a single state x of $\mathcal{A}_{\mathcal{H}}$, and conversely, each state x of $\mathcal{A}_{\mathcal{H}}$ corresponds to a single $\mathcal{H}^j$. The correspondence between $\mathcal{H}^j$ and x means that $\mathcal{H}^j$ contains (all and only) the traces leading to x. This is in fact the "most refined" partition in the sense that decomposing any $\mathcal{H}^j$ of this partition will violate Assumption **A3**. And by using Lemma 5.7.1, we deduce that if this "most refined" partition contains some $\mathcal{H}^j$ such that $(\mathcal{F}, \mathcal{H}^j)$ is not Inf_N-F-CODIAG, then every other decomposition respecting Assumption **A3** contains some $\mathcal{H}^j$ such that $(\mathcal{F}, \mathcal{H}^j)$ is not Inf_N-F-CODIAG. And by using Proposition 5.7.1, we deduce that $(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$.

Next, we show the impact of the structure of the FSAs on the set of languages diagnosable under $\wedge$-$Inf_{\leq N}^{\geq 1}$ architecture.

Proposition 5.7.2 *Consider two (equivalent) FSAs $\mathcal{A}_{\mathcal{H},1}$ and $\mathcal{A}_{\mathcal{H},2}$ accepting $\mathcal{H}$. Assume that for every marked state x of $\mathcal{A}_{\mathcal{H},1}$, there exists a marked state y of $\mathcal{A}_{\mathcal{H},2}$ such that $\mathcal{L}(\mathcal{A}_{\mathcal{H},1}, x) \subseteq \mathcal{L}(\mathcal{A}_{\mathcal{H},2}, y)$. If $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H},2}$, then it is $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H},1}$.*

The proof of Proposition 5.7.2 is given in the appendix.

From Proposition 5.7.2, we deduce straightforwardly (hence, the proof is omitted) the following result.

Proposition 5.7.3 *Consider two FSAs $\mathcal{A}_{\mathcal{H},1}$ and $\mathcal{A}_{\mathcal{H},2}$ accepting $\mathcal{H}$, such that $\mathcal{A}_{\mathcal{H},1}$ is obtained by splitting some states of $\mathcal{A}_{\mathcal{H},2}$ in the sense that : each state of $\mathcal{A}_{\mathcal{H},2}$ is equivalent*

to one or more states of $\mathcal{A}_{\mathcal{H},1}$. *If* $(\mathcal{F},\mathcal{H})$ *is* $\wedge$-$Inf_{\leq N}^{\geq 1}$-*F*-CODIAG *w.r.t.* $\mathcal{A}_{\mathcal{H},2}$, *then it is* $\wedge$-$Inf_{\leq N}^{\geq 1}$-*F*-CODIAG *w.r.t.* $\mathcal{A}_{\mathcal{H},1}$.

Since $\{\mathcal{H}\}$ forms a trivial decomposition of $\mathcal{H}$, we also deduce straightforwardly the following result from Proposition 5.7.2.

Proposition 5.7.4 *If* $(\mathcal{F},\mathcal{H})$ *is* Inf_N-*F*-CODIAG *then it is* $\wedge$-$Inf_{\leq N}^{\geq 1}$-*F*-CODIAG *w.r.t. every FSA accepting* $\mathcal{H}$.

From [Kumar et Takai, 2009], Inf_N-F-CODIAG is a generalization of F-codiagnosability of [Wang *et al.*, 2005] (which is named codiagnosability in [Qiu et Kumar, 2006] and diagnosability under Protocol 3 in [Debouk *et al.*, 2000]), and of conditional F-codiagnosability [Wang *et al.*, 2005]. Therefore, from Proposition 5.7.4, we have that $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG is a generalization of those codiagnosabilities as well.

Next, we compare $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG with the diagnosability property that characterizes the class of languages diagnosable under the centralized architecture with partial observation [Sampath *et al.*, 1995]. The centralized architecture is the framework for which one diagnoser observes partially the plant and issues an effective diagnosis. Formally, the observability is defined as follows.

Definition 5.7.1 *[Sampath et al., 1995]* $(\mathcal{F},\mathcal{H})$ *is said diagnosable if :*

$$\exists l \in \mathbb{Z}^+, \forall s \in \mathcal{F}_l : u \in P^{-1}P(s) \cap \mathcal{L} \Rightarrow u \in \mathcal{F},$$

where P *is the natural projection from* Σ *to* Σ_o.

Proposition 5.7.5 *If* $(\mathcal{F},\mathcal{H})$ *is* $\wedge$-$Inf_{\leq N}^{\geq 1}$-*F*-CODIAG *then* $(\mathcal{F},\mathcal{H})$ *is diagnosable.*

The proof of Proposition 5.7.5 is given in the appendix.

To prove that the converse of Proposition 5.7.5 is not true, we consider the plant $\mathcal{L}$ of Fig. 5.2 where $\Sigma_{o,1} = \{a_1\}$ and $\Sigma_{o,2} = \{a_2\}$. We have $\mathcal{H} = \overline{\{a_1, a_2a_1c^*\}}$ and $\mathcal{F} = \{a_1f\overline{a_2c^*}\}$. $(\mathcal{F},\mathcal{H})$ is diagnosable because the single diagnoser of a centralized architecture observes the order in which a_1 and a_2 are executed, and thus, when it observes a_1a_2 it is sure that the fault has occurred. Let us show that $(\mathcal{F},\mathcal{H})$ is not $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG, i.e., for every decomposition $\{\mathcal{H}^1,\ldots,\mathcal{H}^p\}$ of $\mathcal{H}$, there exists $j \in J$ such that $\mathcal{F}^j[N+1] \cap \mathcal{F}_l \neq \emptyset$, $\forall N \in \mathbb{Z}^+$. In fact, for every decomposition $\{\mathcal{H}^1,\ldots,\mathcal{H}^p\}$ of $\mathcal{H}$, there exists $j \in J$ such that $a_2a_1c^l \in \mathcal{H}^j$ for some $l \geq 1$. Since $P_1(a_1fa_2c^l) = P_1(a_2a_1c^{l-1}) = a_1c^{l-1}$ and $P_2(a_1fa_2c^{l-1}) = P_2(a_2a_1c^{l-1}) = a_2c^{l-1}$, we have, $\forall N \in \mathbb{Z}^+$ and $\forall l \geq 1$, $a_2a_1c^{l-1} \in \mathcal{H}^j[N+1]$

$a_1 f a_2 c^{l-1} \in \mathcal{F}^J[N+1]$, or more precisely $a_1 f a_2 c^{l-1} \in \mathcal{F}^J[N+1] \cap \mathcal{F}_l$. Therefore, $(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG.

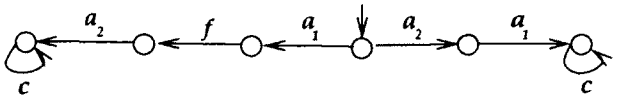

Figure 5.2 Plant $\mathcal{L}$ which is diagnosable and not $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG

5.8 Checking codiagnosability and constructing decomposition of $\mathcal{H}$

We have seen that the notion of codiagnosability is relevant to determine the existence of diagnosers. We have seen three versions of codiagnosability in Definitions 5.6.2 and 5.6.3 :

$\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG for given $p \geq 1$ and $N_1, \ldots, N_p \geq 0$;

$\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG meaning $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG for some unspecified $p \geq 1$ and $N_1, \ldots, N_p \geq 0$;

$\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG meaning $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG for some unspecified $p \geq 1$ and $N_1, \ldots, N_p \leq N$.

Contrary to $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG, the parameters p and $N_1, \ldots, N_p$ must be specified in $\wedge$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-F-CODIAG, which makes $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG more restrictive than $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG. But the problem with $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG is that $N_1, \ldots, N_p$ are not bounded, which makes it undecidable in general. That is why we have defined the (decidable) $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG by limiting the possible values $N_1, \ldots, N_p$ to be checked. In this section, we are indeed interested by $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG. More precisely, we propose au automata based method that checks if $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG w.r.t a given FSA $\mathcal{A}_{\mathcal{H}}$ (accepting $\mathcal{H}$). (In the sequel, the notion of $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG is implicitly w.r.t. a given FSA $\mathcal{A}_{\mathcal{H}}$.) And when the method determines that $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG, it computes a corresponding decomposition (in fact, a partition) of $\mathcal{H}$. For that purpose, we first need to compute the pairs $(\mathcal{H}[k], \mathcal{F}[k])_{k \geq 0}$, or more precisely automata accepting them. This is the subject of Subsection 5.8.1.

5.8.1 Computing automata accepting $\mathcal{F}[k]$ and $\mathcal{H}[k]$, for $k \geq 0$

Recall that the plant is modeled by an automaton $G = (Q, \Sigma, \delta, q_0)$. The FSAs accepting $\mathcal{H}$ and $\mathcal{F}$ are denoted respectively $\mathcal{A}_{\mathcal{H}} = (X, \Sigma, \alpha, x_0, X_m)$ and $\mathcal{A}_{\mathcal{F}} = (Y, \Sigma, \beta, y_0, Y_m)$. X_m and Y_m are the marked states of respectively $\mathcal{A}_{\mathcal{H}}$ and $\mathcal{A}_{\mathcal{F}}$, i.e., $\mathcal{H} = \{s \in \Sigma^* | \alpha(x_0, s) \in X_m\}$

and $\mathcal{F} = \{s \in \Sigma^* | \beta(y_0, s) \in Y_m\}$. $\mathcal{H}$ and $\mathcal{F}$ are constructed as follows. We consider the FSA $A_{H,F}$ with two states, H and F, where H is the initial state. F is reached from H through a transition labeled by the faulty event f. H is self-looped by transitions labeled by the events of $\Sigma \setminus f$, and F is self-looped by transitions labeled by the events of Σ. Note that $G||A_{H,F}$ is universal in the sense that its language is Σ^*. We compute the synchronous composition $G||A_{H,F}$ [Hopcroft et Ullman, 1979], and thus, every state of $G||A_{H,F}$ is identified in the form (x, H) or (x, F), where x is a state of G. $G||A_{H,F}$ and G accept the same language, but $G||A_{H,F}$ has the particularity to distinguish the states reached by traces of $\mathcal{H}$ and $\mathcal{F}$, respectively. Indeed, the states reached by traces of $\mathcal{H}$ (resp. $\mathcal{F}$) are those identified in the form (x, H) (resp. (x, F)). $\mathcal{A}_{\mathcal{H}}$ is obtained from $G||A_{H,F}$ by marking states (x, H) and removing states (x, F). $\mathcal{A}_{\mathcal{H}}$ is obtained from $G||A_{H,F}$ by marking states (x, F) and removing states (x, H) from which no state (x, F) is reachable. Complexity of computing $\mathcal{A}_{\mathcal{H}}$ and $\mathcal{A}_{\mathcal{F}}$ are in $O(|Q|.|\Sigma|)$.

Let $\mathcal{A}_{\mathcal{H}[k]} = (X[k], \Sigma, \alpha[k], x_0[k], X_m[k])$ and $\mathcal{A}_{\mathcal{F}[k]} = (Y[k], \Sigma, \beta[k], y_0[k], Y_m[k])$ be the automata accepting $\mathcal{H}[k]$ and $\mathcal{F}[k]$, respectively, for $k \geq 0$. Recall that $\mathcal{H}[0] = \mathcal{H}$ and $\mathcal{F}[0] = \mathcal{F}$. We have just shown how to compute $\mathcal{A}_{\mathcal{H}[0]} = \mathcal{A}_{\mathcal{H}}$ and $\mathcal{A}_{\mathcal{F}[0]} = \mathcal{A}_{\mathcal{F}}$. Let us now show how to compute inductively $\mathcal{A}_{\mathcal{H}[k+1]}$ and $\mathcal{A}_{\mathcal{F}[k+1]}$ from $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$. Since $\mathcal{H}[k+1] = \mathcal{H}[k] \cap \bigcap_{i \in I} P_i^{-1} P_i(\mathcal{H}[k])$, $\mathcal{A}_{\mathcal{H}[k+1]}$ is computed as follows. For each $i \in I$, we compute the projection $P_i(\mathcal{A}_{\mathcal{H}[k]})$ as indicated in [Barrett et Couch, 1979; Hopcroft et Ullman, 1979]. Note that the result of projection may be nondeterministic, we do not determinize it for avoiding computational complexity. Then, the inverse projection of $P_i(\mathcal{A}_{\mathcal{H}[k]})$, i.e., $P_i^{-1} P_i(\mathcal{A}_{\mathcal{F}[k]})$ is obtained by simply adding self-loop transitions labeled by events of $\Sigma \setminus \Sigma_{o,i}$ at each state of $P_i(\mathcal{A}_{\mathcal{F}[k]})$. After that, a synchronous composition is applied between all $P_i^{-1} P_i(\mathcal{A}_{\mathcal{F}[k]})$, $i \in I$, and $\mathcal{A}_{\mathcal{H}[k]}$. The set of marked states of $\mathcal{A}_{\mathcal{H}[k+1]}$ is obtained in the usual way. $\mathcal{A}_{\mathcal{F}[k+1]}$ is computed with a similar approach. We denote by $X_m[k]$ and $Y_m[k]$ the set of marked states of $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$, respectively. Emptiness of $\mathcal{H}[k]$ (resp. $\mathcal{F}[k]$) is equivalent to emptiness of $X_m[k]$ (resp. $Y_m[k]$). Given $\mathcal{H}^j \subseteq \mathcal{H}$, $\mathcal{A}_{\mathcal{H}^j}$ is obtained from $\mathcal{A}_{\mathcal{H}}$ by deleting states and transitions that lead to traces not in $\mathcal{H}^j$, the above procedure to compute $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$, $\forall k \leq 1$, can be adapted straightforwardly to compute $\mathcal{A}_{\mathcal{H}^j[k]}$ and $\mathcal{A}_{\mathcal{F}^j[k]}$, where $\mathcal{H}^j[k]$ and $\mathcal{F}^j[k]$ are given by Eqs. (5.12).

We have the following lemmas that evaluate the orders of $|Y[k+1]|$, $|X[k+1]|$, $|\beta[k+1]|$ and $|\alpha[k+1]|$ from $|Y[k]|$ and $|X[k]|$, and evaluate the complexity for computing $\mathcal{A}_{\mathcal{F}[k+1]}$ and $\mathcal{A}_{\mathcal{H}[k+1]}$ from $\mathcal{A}_{\mathcal{F}[k]}$ and $\mathcal{A}_{\mathcal{H}[k]}$. That is, we consider one step, from k to $k+1$, for $k > 0$, where $n = |I|$.

Lemma 5.8.1 *$|X[k+1]|$ is in $O(|X[k]|.|Y[k]|^n)$ and $|\alpha[k+1]|$ is in $O(|X[k]|^2.|Y[k]|^{2n}.|\Sigma|) = O(|X[k+1]|^2.|\Sigma|)$. Symmetrically, $|Y[k+1]|$ is in $O(|Y[k]|.|X[k]|^n)$ and $|\beta[k+1]|$ is in $O(|Y[k]|^2.|X[k]|^{2n}.|\Sigma|) = O(|Y[k+1]|^2.|\Sigma|)$.*

The proof of Lemma 5.8.1 is given in the appendix.

Lemma 5.8.2 *Computing $\mathcal{A}_{\mathcal{H}[k+1]}$ from $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$ is performed in $O(|\alpha[k+1]|) = O(|X[k]|^2.|Y[k]|^{2n}.|\Sigma|)$. Symmetrically, computing $\mathcal{A}_{\mathcal{F}[k+1]}$ from $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$ is performed in $O(|\beta[k+1]|) = O(|Y[k]|^2.|X[k]|^{2n}\ |\Sigma|)$*

The proof of Lemma 5 8.2 is given in the appendix.

Each state $u \in Y_m[k]$ can be expressed in the form $(u_1, \ldots, u_n, u_{n+1})$, where $u_i \subseteq X[k-1]$ and $u_i \cap X_m[k-1] \neq \emptyset$, for $i \in I$, and $u_{n+1} \in Y_m[k-1]$, $\forall k \geq 1$. Symmetrically, each state $v \in X_m[k]$ can be presented in the form $(v_1, \ldots, v_n, v_{n+1})$, where $v_i \subseteq Y[k-1]$ and $v_i \cap Y_m[k-1] \neq \emptyset$, for $i \in I$, and $v_{n+1} \in X_m[k-1]$.

Having seen how to compute $(\mathcal{A}_{\mathcal{H}[k]}, \mathcal{A}_{\mathcal{F}[k]})_{k\geq 0}$, we can now show how to check if $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$. The following theorem tests for the Inf_N-CODIAG. The theorem is an adaptation of Theorem 2 of [Kumar et Takai, 2009] by considering $\mathcal{H}^J$, $\mathcal{F}^J[N+1]$ and $\mathcal{A}_{\mathcal{F}^J[N+1]}$, instead of $\mathcal{H}$, $\mathcal{F}[N+1]$ and $\mathcal{A}_{\mathcal{F}[N+1]}$ (see [Kumar et Takai, 2009] for the proof)

Theorem 5.8.1 *Consider a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$, the pair $(\mathcal{F}, \mathcal{H}^J)$ is not Inf_N-F-CODIAG if and only if there exists a cycle of faulty states in the acceptor $\mathcal{A}_{\mathcal{F}^J[N+1]}$ of $\mathcal{F}^J[N+1]$.*

In Subsection 5.8 2, we consider the particular case where $p = 1$ (no decomposition of $\mathcal{H}$) and $N = 0$ (no inference). In Subsection 5.8.4, we consider the general case where $p \geq 1$ and $N \geq 0$.

5.8.2 Checking if $(\mathcal{F}, \mathcal{H})$ is Inf_0-Codiag

Let us first check the basic case where we have codiagnosability without decomposing $\mathcal{H}$ or $\mathcal{F}$ (no multi-decision) and without inference, i.e., $(\mathcal{F}, \mathcal{H})$ is Inf_0-F-CODIAG. That is, the aim is to check whether $\mathcal{H}[1]$ or $\mathcal{F}[1]$ is empty, i.e., $Y_m[1]$ or $X_m[1]$ is empty. The following proposition is deduced from Lemma 5.8.2 by taking $k = 0$.

Proposition 5.8.1 *Checking if $(\mathcal{F}, \mathcal{H})$ is Inf_0-F-CODIAG (i.e., there exists a cycle of faulty states in $\mathcal{A}_{\mathcal{F}[1]}$) is performed in $O(|Y|^2.|X|^{2n}.|\Sigma|)$.*

The proof of Proposition 5.8.1 is given in the appendix.

For checking if $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$, we first need to define the notion of multi-marking.

5.8.3 Multi-marking in $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$

When $(\mathcal{F}, \mathcal{H})$ is not Inf_0-CODIAG, the aim is to check whether there exists a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ respecting Assumption **A3** w.r.t. $\mathcal{A}_{\mathcal{H}}$ such that $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG for some $p \geq 1$ and some $N_1, \ldots, N_p \leq N$. We use the notion of *multi-marking* of [de Queiroz *et al.*, 2005] to construct partitions $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ of $\mathcal{H}$ and determine whether $\mathcal{H}^j[k]$ or $\mathcal{F}^j[k]$ is empty, for $k \geq 0$ and $j \leq p$. Let $\{X^1_m, \ldots, X^p_m\}$ be a decomposition of X_m, a state $w \in X_m[k]$ (or $w \in Y_m[k]$) is said X^j_m-*marked* if w remains marked when only the states of X^j_m (instead of X_m) are marked in $\mathcal{A}_{\mathcal{H}}$. This multiple-marking is determined inductively in $X_m[k]$ and $Y_m[k]$ (marked states of $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$, respectively), $k \geq 1$, as follows :

Consider a state $u = (u_1, \ldots, u_n, u_{n+1}) \in X_m[1]$ and thus, $u_{n+1} \in X_m$. The state u is X^j_m-marked if $u_{n+1} \in X^j_m$.

Consider a state $v = (v_1, \ldots, v_n, v_{n+1}) \in Y_m[1]$ and thus, $\forall i \in I$, $v_i \cap X_m \neq \emptyset$. The state v is X^j_m-marked if, $\forall i \in I$, $v_i \cap X^j_m \neq \emptyset$.

Consider a state $v = (v_1, \ldots, v_n, v_{n+1}) \in Y_m[k]$, $k \geq 2$, and thus, $\forall i \in I$, $v_i \cap X_m[k-1] \neq \emptyset$ and $v_{n+1} \in Y_m[k-1]$. The state v is X^j_m-marked if, $\forall i \in I$, v_i contains an X^j_m-marked state of $X_m[k-1]$ and v_{n+1} is an X^j_m-marked state of $Y_m[k-1]$.

Consider a state $u = (u_1, \ldots, u_n, u_{n+1}) \in X_m[k]$, $k \geq 2$, and thus, $\forall i \in I$, $u_i \cap Y_m[k-1] \neq \emptyset$ and $u_{n+1} \in X_m[k-1]$. The state u is X^j_m-marked if, $\forall i \in I$, u_i contains an X^j_m-marked state of $Y_m[k-1]$ and u_{n+1} is an X^j_m-marked state of $X_m[k-1]$.

Note that a state may be at the same time X^i_m-marked and X^j_m-marked for $i \neq j$. The multiple marking can be interpreted as follows : A state in $X_m[k]$ (resp. $Y_m[k]$) is X^j_m-marked iff it is reached by trace(s) of $\mathcal{H}^j[k]$ (resp., $\mathcal{F}^j[k]$). Therefore, $\mathcal{H}^j[k]$ (resp. $\mathcal{F}^j[k]$) is empty iff $X_m[k]$ (resp., $Y_m[k]$) contains no X^j_m-marked state.

Let $X^j_m \subseteq X_m$, $X_m[k]|_{X^j_m} \subseteq X_m[k]$ and $Y_m[k]|_{X^j_m} \subseteq Y_m[k]$ denote the X^j_m-marked states of $X_m[k]$ and $Y_m[k]$, respectively. A decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ is formally related to its corresponding decomposition $\{X^1_m, \ldots, X^p_m\}$ of X_m as follows :

$$\mathcal{H}^j = \{s \in \overline{K} \mid \alpha(x_0, s) \in X^j_m\}. \tag{5.16}$$

Note that, for each decomposition $\{X^1_m, \ldots, X^p_m\}$ of X_m, the decomposition $D = \{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$, for which every $X^j_m \in D$ is given by Eq. (5.16), satisfies Assumption **A3**. Indeed,

each $\mathcal{H}^j$ contains all and only the traces leading to the states contained in X_m^j. Hence, whenever we say that a decomposition $D = \{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ satisfies Assumption **A3**, this means that there exists a decomposition $\{X_m^1, \ldots, X_m^p\}$ of X_m such that every $X_m^j \in D$ is given by Eq. (5.16).

The following proposition shows how the languages $\mathcal{H}^j[k]$ and $\mathcal{F}^j[k]$, $\forall k \geq 1$, are obtained from Eq. (5.16).

Proposition 5.8.2 *Consider FSAs $\mathcal{A}_\mathcal{F}$ and $\mathcal{A}_\mathcal{H}$, and a decomposition $\{X_m^1, \ldots, X_m^p\}$ of X_m. By considering the decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ such that each $\mathcal{H}^j$ is given by Eq. (5.16), $\mathcal{H}^j[k]$ and $\mathcal{F}^j[k]$, $\forall k \geq 1$, are computed as follows :*

$$\begin{aligned} \mathcal{H}^j[k] &= \{s \in \Sigma^* \mid \alpha[k](x_0[k], s) \in X_m[k]|_{X_m^j}\}, \\ \mathcal{F}^j[k] &= \{s \in \Sigma^* \mid \beta[k](y_0[k], s) \in Y_m[k]|_{X_m^j}\}. \end{aligned} \tag{5.17}$$

The proof of Proposition 5.8.2 is given in the appendix.

5.8.4 Checking if $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-Codiag w.r.t. $\mathcal{A}_\mathcal{H}$

By using Theorem 5.8.1, the following theorem tests for the $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG of a pair $(\mathcal{F}, \mathcal{H})$ (w.r.t. a FSA $\mathcal{A}_\mathcal{H}$).

Theorem 5.8.2 *Consider FSAs $\mathcal{A}_\mathcal{F}$ and $\mathcal{A}_\mathcal{H}$, and a decomposition $\{X_m^1, \ldots, X_m^p\}$ of X_m. $(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG w.r.t. $\mathcal{A}_\mathcal{H}$ if and only if, for every decomposition $D = \{X_m^1, \ldots, X_m^p\}$ of X_m satisfying Assumption **A3**, there exists a cycle of faulty states through which an X_m^j-marked state is reached in $\mathcal{A}_{\mathcal{F}[N+1]}$, for some $X_m^j \in D$.*

The proof of Theorem 5.8.2 is given in the appendix.

The existence of cycles of faulty states can be expressed by the function $\mathcal{C}_f(.)$ defined as follows.

Definition 5.8.1 *Given a state $v \in Y[k]$, $\forall k \geq 1$, $\mathcal{C}_f(v)$ denotes the set of traces reaching v through a cycle of faulty states of $\mathcal{A}_{\mathcal{F}[k]}$.*

Note that, $\mathcal{C}_f(v)$ depends implicitly on k. Let us show how the multi-marking can be used to check whether $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG w.r.t. $\mathcal{A}_\mathcal{H}$. For that purpose, we target to find a decomposition $\{X_m^1, \ldots, X_m^p\}$ of X_m such that, $\forall j \in J$, X_m^j satisfies the following condition,

$$\begin{aligned}&\forall v = (v_1, \ldots, v_n, v_{n+1}) \in Y_m[1]|_{X_m^j} \text{ s.t. } \mathcal{C}_f(v) \neq \emptyset,\\ &\forall (a_1, \ldots, a_n) \in (v_1 \cap X_m^j) \times \cdots \times (v_n \cap X_m^j) : |\bigcup_{i \in I} \{a_i\}| = 1.\end{aligned} \tag{5.18}$$

Cond. (5.18) requires that, $\forall j \in J$, for every $v \in Y_m[1]$ such that $\mathcal{C}_f(v) \neq \emptyset$, if all its v_i, $i = 1, \ldots, n$, contain states of X_m^j, then all these v_i contain in fact the *same single state* $x \in X_m^j$ and no other state of X_m^j. Note that Cond. (5.18) is satisfied by the trivial partition $(X_m^j)_{j=1,\ldots,|X_m|}$ such that each X_m^j is a singleton.

In the case where $(\mathcal{F}, \mathcal{H})$ is not Inf_0-F-CODIAG, if there exists a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ satisfying Cond. (5.18) such that, for some j, $Y_m[1]$ has at least one X_m^j-marked state reached by traces through a cycle of faulty states, then, from Eqs. (5.16) and (5.17) and Theorem 5.8.1, $(\mathcal{F}, \mathcal{H}^j)$ is not Inf_0-F-CODIAG. In this case, we have to check if $(\mathcal{F}, \mathcal{H}^j)$ is Inf_N-F-CODIAG for some $N \geq 0$. Before that, we show in the following lemma an important result that will be used to show the relevance of Cond. (5.18).

Lemma 5.8.3 *Consider a decomposition $D = \{X_m^1, \ldots, X_m^p\}$ of X_m satisfying Cond. (5.18). $\forall k \geq 1$, $\forall v \in Y_m[k]$, if $\mathcal{C}_f(v) \neq \emptyset$ and v is X_m^j-marked for some $X_m^j \in D$, then there exists $x \in X_m^j$ such that v is $\{x\}$-marked and not $\mathcal{X}$-marked, $\forall \mathcal{X} \subseteq X_m^j$ s.t. $\mathcal{X} \neq \{x\}$.*

The proof of Lemma 5.8.3 is given in the appendix.

We have the following theorem which states that $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG of $(\mathcal{F}, \mathcal{H})$ needs to be checked uniquely in a decomposition satisfying Cond. (5.18).

Theorem 5.8.3 *Consider a FSA $\mathcal{A}_{\mathcal{H}}$ accepting $\mathcal{H}$, an integer $N \in \mathbb{Z}^+$ and a decomposition $\{X_m^1, \ldots, X_m^p\}$ of X_m satisfying Cond. (5.18). $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf_{\leq N}^{\geq 1}$-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$ if and only if $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf_{\leq N}^{\geq 1}$-CODIAG w.r.t. $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$, where $\mathcal{H}^j$ is given by Eq. (5.16).*

The proof of Theorem 5.8.3 is given in the appendix.

Given a decomposition $(X_m^j)_{j \in J}$ satisfying Cond. (5.18), by computing iteratively $(\mathcal{A}_{\mathcal{F}[k]}, \mathcal{A}_{\mathcal{H}[k]})$ until $k = N + 1$, we have :

$(\mathcal{F}, \mathcal{H}^j)$ is Inf_k-F-CODIAG if in $\mathcal{A}_{\mathcal{F}[k+1]}$ there is no reachable cycle of faulty states through which an X_m^j-marked state is reached. $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG if, $\forall j \in J$, we have found some $N_j \leq N$ such that $(\mathcal{F}, \mathcal{H}^j)$ is Inf_{N_j}-F-CODIAG. Otherwise, $(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG.

$(\mathcal{F}, \mathcal{H}^j)$ is not $Inf_{\geq 0}$-F-CODIAG if for some $k < N$, $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{H}[k+1]}$ have the same non-empty X_m^j-marked language (an X_m^j-marked language is the set of traces leading to an X_m^j-marked state), and $\mathcal{A}_{\mathcal{F}[k]}$ and $\mathcal{A}_{\mathcal{F}[k+1]}$ have the same non-empty X_m^j-marked language, and in $\mathcal{A}_{\mathcal{F}^j[k+1]}$, there exists a cycle of faulty states through which an X_m^j-marked state is reached. $(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG if we have found some $j \in J$ such that $(\mathcal{F}, \mathcal{H}^j)$ is not $Inf_{\geq 0}$-F-CODIAG.

Consider a decomposition $(X_m^j)_{j \in J}$ and let us evaluate the computational complexity for checking whether $(\mathcal{F}, \mathcal{H}^j)$ is Inf_k-F-CODIAG, $\forall j \in J$ and for some $k \in \mathbb{Z}^+$. We are interested by the *single step* k of inference. By "single step k", we mean $\forall j \in J$, $\mathcal{H}^j[k]$ and $\mathcal{F}^j[k]]$ (rather, their automata $\mathcal{A}_{\mathcal{H}^j[k]}$ and $\mathcal{A}_{\mathcal{F}^j[k]}$) are assumed computed. We have the following proposition which can be deduced from Lemma 5.8.2 :

Proposition 5.8.3 *Consider a decomposition $(X_m^j)_{j \in J}$ and a step k of inference. Assuming that $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$ are computed, the complexity for checking whether in $\mathcal{A}_{\mathcal{F}[k+1]}$ there is no reachable cycle of faulty states through which an X_m^j-marked state is reached, $\forall j \in J$, is in the worst case in $O(|Y[k+1]|^2.|\Sigma|) = O(|X[k]|^{2n}.|Y[k]|^2.|\Sigma|)$.*

The proof of Proposition 5.8.3 is given in the appendix.

5.8.5 Procedure of decomposition of X_m

We have noted that Cond. (5.18) is satisfied by the trivial partition $(X_m^j)_{j=1 \ \ p}$ where each X_m^j is a singleton, and thus, $p = |X_m|$. To reduce execution time and memory space during the execution of diagnosis, it is preferable to find a non trivial partition satisfying Cond. (5.18) with a smaller p, if any. Let us propose a procedure that computes such a non trivial partition $(X_m^j)_{j \in J}$, if any. Before presenting our partition procedure, we need to define Elig($\mathcal{X}$).

Definition 5.8.2 *Consider $\mathcal{X} \subseteq X_m$ such that $\mathcal{X}$ satisfies Cond. (5.18).* Elig($\mathcal{X}$) *contains all the states of $X_m \setminus \mathcal{X}$ that can be added individually to $\mathcal{X}$ without violating Cond. (5.18). Formally,*

$$\text{Elig}(\mathcal{X}) = \{x \in X_m \setminus \mathcal{X} \mid \mathcal{X} \cup \{x\} \textit{ satisfies Cond. (5.18)}\}.$$

In the sequel, Elig($\{x\}$) is written Elig(x). Note that, Elig($\emptyset$) $= X_m$. Our partition procedure will construct every X_m^j by moving iteratively some states from X_m to X_m^j ; let then Z_m denote the current remaining part of X_m (i.e., states of X_m that have not yet been moved to a X_m^j). The basic principle for constructing X_m^1 is as follows :

1. Initializations : (a) $Z_m \leftarrow X_m$, (b) $X_m^1 \leftarrow \emptyset$.

2. While $\mathrm{Elig}(X_m^1) \cap Z_m \neq \emptyset$: a) We select randomly a state $x \in \mathrm{Elig}(X_m^1) \cap Z_m$, and b) we move x from Z_m to X_m^1, that is : $X_m^1 \leftarrow X_m^1 \cup \{x\}$, $Z_m \leftarrow Z_m \setminus \{x\}$.

The construction of X_m^1 is completed when the while-loop terminates, i.e., when $\mathrm{Elig}(X_m^1) \cap Z_m = \emptyset$. Note that this while-loop terminates in finite time, because the computed sets $\mathrm{Elig}(X_m^1) \cap Z_m$ are finite and define a monotonically decreasing sequence of state sets. The above procedure guarantees that the computed X_m^1 satisfies Cond. (5.18) and that Cond. (5.18) is not satisfied as soon as we add any other state to X_m^1.

The other X_m^j , $j > 1$, are constructed by repeating the above procedure without Substep 1(a). The construction of the partition is complete when $Z_m = \emptyset$.

The set $\mathrm{Elig}(\mathcal{X})$ has been defined in Def. 5.8.2, let us now see how it is computed for any $\mathcal{X} \subseteq X_m$. We define for every $v = (v_1, \ldots, v_n, v_{n+1}) \in Y_m[1]$, and $\mathcal{X} \subseteq X$, $\mathrm{ID}_v(\mathcal{X})$ by :

$$\mathrm{ID}_v(\mathcal{X}) = \{\imath \in I \mid v_\imath \cap \mathcal{X} = \emptyset\}, \tag{5.19}$$

therefore, $\mathrm{ID}_v(\emptyset) = I$. In the next lemma we present a property of $\mathrm{ID}_v(.), \forall v \in Y_m[1]$.

Lemma 5.8.4 *Given two subsets $\mathcal{W}, \mathcal{Z} \subseteq X_m$, we have :* $\mathrm{ID}_v(\mathcal{W} \cup \mathcal{Z}) = \mathrm{ID}_v(\mathcal{W}) \cap \mathrm{ID}_v(\mathcal{Z})$.

The proof of Lemma 5.8.4 is given in the appendix.

The following lemma states a new reformulation of Cond. (5.18) by using the functions $\mathrm{ID}_v(.)$:

Lemma 5.8.5 *A subset $\mathcal{X} \subseteq X_m$ satisfies Cond. (5.18) iff :*

$$\forall v \in Y_m[1] \text{ s.t. } \mathcal{C}_f(v) \neq \emptyset : \mathrm{ID}_v(\mathcal{X}) = \emptyset \Rightarrow |\mathcal{X} \cap \bigcup_{\imath \in I} v_\imath| = 1. \tag{5.20}$$

The proof of Lemma 5.8.5 is given in the appendix.

We will now show how the functions $\mathrm{ID}_v(.), \forall v \in Y_m[1]$, can be used for computing $\mathrm{Elig}(\mathcal{X})$, for $\mathcal{X} \subseteq X_m$. We will present an *inductive* computation method :

Basis : When $\mathcal{X}$ is a singleton $\{x\}$, $\mathrm{Elig}(x)$ can be computed as follows :

Proposition 5.8.4 *Given a marked state $x \in X_m$,*

$$\mathrm{Elig}(x) = X_m \setminus [\{x\} \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathcal{C}_f(v) \neq \emptyset \\ \mathrm{ID}_v(x) = \emptyset}} \bigcup_{\imath \in I} v_\imath \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathcal{C}_f(v) \neq \emptyset \\ \mathrm{ID}_v(x) \neq I}} \bigcap_{\imath \in \mathrm{ID}_v(x)} v_\imath]. \tag{5.21}$$

The proof of Proposition 5.8.4 is given in the appendix.

Inductive step : $\text{Elig}(\mathcal{X} \cup \{x\})$ can be computed from $\text{Elig}(\mathcal{X})$ and $\text{Elig}(x)$ as follows :

Proposition 5.8.5 *Given a subset $\mathcal{X} \subseteq X_m$ satisfying Cond. (5.18), for every $x \in \text{Elig}(\mathcal{X})$ we have :*

$$\text{Elig}(\mathcal{X} \cup \{x\}) = (\text{Elig}(\mathcal{X}) \cap \text{Elig}(x)) \setminus [\bigcup_{\substack{v \in Y_m[1]: \\ \text{ID}_v(\mathcal{X} \cup \{x\}) \neq I \\ C_f(v) \neq \emptyset}} \bigcap_{i \in \text{ID}_v(\mathcal{X} \cup \{x\})} v_i]. \tag{5.22}$$

The proof of Proposition 5.8.5 is given in the appendix.

Let us evaluate the complexity of the above 2-step partition procedure. For that purpose, we first need to evaluate the complexities of computing $\text{ID}_v(x)$ and $\text{Elig}(x)$.

Lemma 5.8.6 *Given $x \in X_m$ and $v \in Y_m[1]$, the complexity for computing $\text{ID}_v(x)$ is bounded by $O(n.|X|)$.*

The proof of Lemma 5.8.6 is given in the appendix.

Lemma 5.8.7 *Given $x \in X_m$, the complexity for computing $\text{Elig}(x)$ is bounded by $O(n.|Y_m[1]|.|X|^2)$.*

The proof of Lemma 5.8.7 is given in the appendix.

Lemma 5.8.8 *Consider $x \in X_m$, $\mathcal{X} \subseteq X_m$, and assume we are given $\text{Elig}(\mathcal{X})$ and $\text{Elig}(x)$. The complexity for computing $\text{Elig}(\mathcal{X} \cup \{x\})$ (by Eq. (5.22)) is bounded by $O(n.|Y_m[1]|.|X|^2)$.*

The proof of Lemma 5.8.8 is given in the appendix.

Proposition 5.8.6 *The complexity of our partition procedure is bounded by $O(n.|Y_m[1]|.|X|^3)$.*

The proof of Proposition 5.8.6 is given in the appendix.

5.8.6 Conclusion on the complexity of our framework

Let us compare the computational complexity of our multi-decision framework with the complexity of the inference-based framework of [Kumar et Takai, 2009]. We have evaluated the complexities of the operations of our framework throughout this section 5.8. The most costly "new" operations that have been added to the framework of [Kumar et Takai, 2009] to construct our multi-decision diagnosis are in Subsection 5.8.5, the partition procedure,

and in Subsection 5.8.4, the procedure for checking, $\forall j \in J$, if there exists a cycle of faulty states in $\mathcal{A}_{\mathcal{F}^j[k+1]}$, in one step k, that is, $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$ are given. For brevity, we will call it "checking procedure".

Consider the operation for computing $\mathcal{A}_{\mathcal{H}[k+1]}$ and $\mathcal{A}_{\mathcal{F}[k+1]}$ from $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$, which is a part of both frameworks ([Kumar et Takai, 2009] and ours). Let us compare its complexity (evaluated in lemma 5.8.2 of Section 5.8.1) with the complexities of the above two procedures (partition and checking) :

1. The complexity of lemma 5.8.2 is higher than the complexity of the partition procedure evaluated in Proposition 5.8.6 (Section 5.8.5).

2. The complexity of lemma 5.8.2 is in the same order of the complexity of the checking procedure evaluated in Proposition 5.8.3 (Section 5.8.4).

That is, the complexity of the inference-based framework of [Kumar et Takai, 2009] is in the same order of the complexities of the new operations that have been added to [Kumar et Takai, 2009] to construct our multi-decision framework. Consequently, in terms of Big-Oh, the complexity of our multi-decision framework is comparable to the complexity of the inference-based framework of [Kumar et Takai, 2009]. Intuitively, this result may be surprising. An explanation is that we have restricted the set of possible decompositions in two ways :

1. We have used a finite state-based approach by using Assumption **A3** (Section 5.5). In this way, we have eliminated the decompositions that do not satisfy Assumption **A3** w.r.t. a given FSA. The number of these eliminated decompositions may be infinite, while the number of the remaining eligible decompositions is certainly finite.

2. We have developed a procedure that computes a *single* partition, which guarantees that if we have not codiagnosability for this partition, then there exists no decomposition for which we have codiagnosability.

Therefore, computing $\mathcal{A}_{\mathcal{F}[1]}$ is in general more complex in comparison to computing a partition that satisfies Cond. (5.18). Hence, the overall complexity to compute a partition satisfying Cond. (5.18), and then checking if $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG, is in the order of $O(|Y|^2.|X|^{2n}.|\Sigma|)$.

5.9 Algorithm for computing a partition of $\mathcal{H}$ and checking $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-Codiag

Algorithm 2 implements our results of Section 5.8. That is, it constructs a partition of X_m satisfying Cond. (5.18) and then checks if $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG. Based on results presented throughout Section 5.8, the following theorem states the correctness of our algorithm.

Theorem 5.9.1 *Algorithm 2 is correct in the sense that each of its three outputs is generated if and only if it is true.*

The proof of Theorem 5.9.1 is given in the appendix.

Input: G, N.
Initialization : $np \leftarrow 1$;
Compute $\mathcal{A}_{\mathcal{F}}$, $\mathcal{A}_{\mathcal{H}}$, $\mathcal{A}_{\mathcal{F}[1]}$ and $\mathcal{A}_{\mathcal{H}[1]}$; /* Subsection 5.8.1 */
if $\forall v \in Y_m[1], \mathcal{C}_f(v) = \emptyset$ **then** return "$(\mathcal{F}, \mathcal{H})$ is Inf_0-CODIAG"; /* Subsection 5.8.2 */
/* Compute $\mathrm{ID}_v(x)$ and $\mathrm{Elig}(x)$ as explained in Subsection 5.8.5 */
foreach $x \in X_m$ **do**
  foreach $v \in Y_m[1]$ **do** Compute $\mathrm{ID}_v(x)$; /* by using Eq. (5.19) */
  Compute $\mathrm{Elig}(x)$; /* by using Eq. (5.21) */
end
/* Compute partition as explained in Subsection 5.8.5 */
$Z_m \leftarrow X_m$, $j \leftarrow 1$, $X_m^1 \leftarrow \emptyset$;
while $Z_m \neq \emptyset$ **do**
  Select some x in Z_m;
  $Z_m \leftarrow Z_m \backslash \{x\}$, $X_m^j \leftarrow X_m^j \cup \{x\}$; /* the selected x is moved from Z_m to X_m^j */
  while $\mathrm{Elig}(X_m^j) \cap Z_m \neq \emptyset$ **do**
    Select some x in $\mathrm{Elig}(X_m^j) \cap Z_m$;
    foreach $v \in Y_m[1]$ **do**
      $\mathrm{ID}_v(X_m^j \cup \{x\}) = \mathrm{ID}_v(X_m^j) \cap \mathrm{ID}_v(x)$; /* Using Lemma 5.8.4 */
    end
    $X_m^j \leftarrow X_m^j \cup \{x\}$; $Z_m \leftarrow Z_m \backslash \{x\}$; /* the selected x is moved from Z_m to X_m^j */
    Compute $\mathrm{Elig}(X_m^j)$; /* Using Eq. (5.22) */
  end
  $j \leftarrow j + 1$;
end
$p \leftarrow j$;
/* Check if $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG as explained in Subsection 5.8.4 */
$\mathrm{NPar} \leftarrow \{1, \ldots, p\}$;
for $k \leftarrow 1$ **to** N **do**
  foreach $j \in \mathrm{NPar}$ **do**
    if *there is no cycle of faulty states in* $\mathcal{A}_{\mathcal{F}^j[k]}$ **then** $N_j \leftarrow k - 1$; Remove j from NPar;
  end
  if $[\mathrm{NPar} = \emptyset]$ **then** return "$(\mathcal{F}, \mathcal{H})$ is $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-CODIAG";
  if $k \leq N$ **then**
    Compute $\mathcal{A}_{\mathcal{H}[k+1]}$, $\mathcal{A}_{\mathcal{F}[k+1]}$; /* Subsection 5.8.1 */
  else
    Return "$(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG";
  end
end

Algorithm 2: Constructing a partition of X_m and checking if $(\mathcal{F}, \mathcal{H})$ is Inf_0-F-CODIAG or $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG

5.10 Example

We continue with Example 5.5.1 for which the plant is represented in Fig. 5.1. The automaton $\mathcal{A}_{\mathcal{H}}$ is obtained from the automaton of Fig. 5.1 by marking the states $X_m = \{H^1, H^2, H^3, H^4, H^5, H^6, H^7\}$ and removing states F^1, F^2, F^3, F^4, F^5 and F^6. The automaton $\mathcal{A}_{\mathcal{F}}$ is obtained from the automaton of Fig. 5.1 by marking the states $Y_m = \{F^1, F^2, F^3, F^4, F^5, F^6\}$ and removing states H^4, H^6 and H^7.

Recall that the marked states of $\mathcal{A}_{\mathcal{H}[1]}$ are identified in the form $u = (u_1, \ldots, u_n, v_{n+1})$, where $u_i \subseteq Y[k-1]$ and $u_i \cap Y_m[k-1] \neq \emptyset$, for $i \in I$, and $u_{n+1} \in X_m[k-1]$. For our example, $\mathcal{A}_{\mathcal{H}[1]}$ is represented in Fig. 5.3. The marked states $X_m[1] = \{H_1^j | 2 \leq i \leq 13\}$ are presented in the form $H_1^j = (F^{p_1,p_2,\ }, F^{q_1,q_2,\ }, H^r)$, where $F^{\nu_1,\nu_2,\ }$ denotes the set $\{F^{\nu_1}, F^{\nu_2}, \ldots\} \subseteq Y_m$ and $H^r \in X_m$. In a similar way, the automaton $\mathcal{A}_{\mathcal{F}[1]}$, represented on Fig. 5.4, has its marked states $Y_m[1] = \{F_1^j | 5 \leq i \leq 16\}$ are presented in the form $F_1^j = (H^{p_1,p_2,\ }, H^{q_1,q_2,\ }, F^r)$, where $H^{\nu_1,\nu_2,\ }$ denotes the set $\{H^{\nu_1}, H^{\nu_2}, \ldots\} \subseteq X_m$ and $F^r \in Y_m$. For example, a trace s reaching a state $H_1^j = (F^{p_1,p_2,\ }, F^{q_1,q_2,\ }, H^r)$ means that s reaches the state $H^r \in X_m$ in $\mathcal{A}_{\mathcal{H}}$, and there exists traces t_1 and t_2 that reach marked states in respectively $\{F^{p_1}, F^{p_2}, \ldots\} \subseteq Y_m$ and $\{F^{q_1}, F^{q_2}, \ldots\} \subseteq Y_m$, such that $P_1(s) = P_1(t_1)$ and $P_2(s) = P_2(t_2)$.

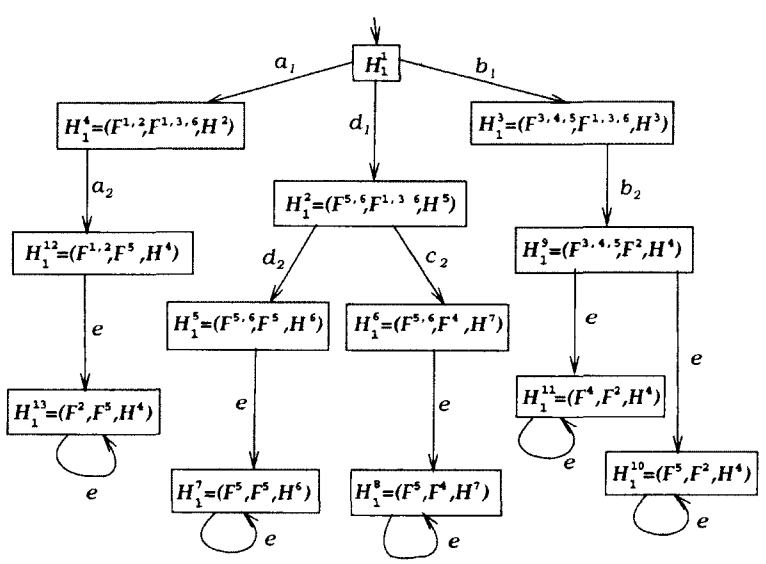

Figure 5.3 $\mathcal{A}_{\mathcal{H}[1]}$ for the example of Figure 5.1

Let us show how the procedure of Section 5.8.5 is used to partition X_m. For that purpose, we first compute $\mathrm{ID}_v(\{x\})$ for every $x \in X_m$ (i.e., marked states of $\mathcal{A}_{\mathcal{H}}$) and every $v \in Y_m[1]$ (i.e., marked states of $\mathcal{A}_{\mathcal{F}[1]}$), which is represented in Table 5.4.

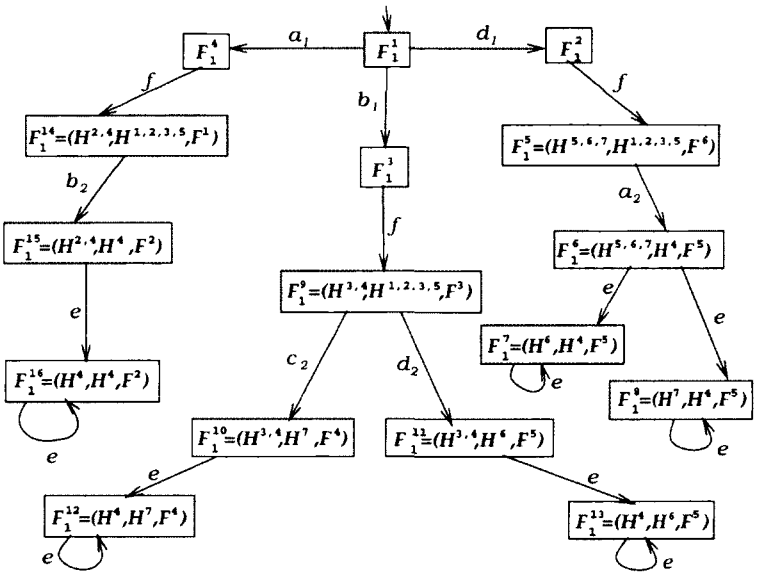

Figure 5.4 $\mathcal{A}_{\mathcal{F}[1]}$ for the example of Figure 5.1

$v =$	F_1^5	F_1^6	F_1^7	F_1^8	F_1^9	F_1^{10}	F_1^{11}	F_1^{12}	F_1^{13}	F_1^{14}	F_1^{15}	F_1^{16}
$\mathrm{ID}_v(H^1)$	$\{1\}$	I	I	I	$\{1\}$	I	I	I	I	$\{1\}$	I	I
$\mathrm{ID}_v(H^2)$	$\{1\}$	I	I	I	$\{1\}$	I	I	I	I	$\emptyset$	$\{2\}$	I
$\mathrm{ID}_v(H^3)$	$\{1\}$	I	I	I	$\emptyset$	$\{2\}$	$\{2\}$	I	I	$\{1\}$	I	I
$\mathrm{ID}_v(H^4)$	I	$\{1\}$	$\{1\}$	$\{1\}$	$\{2\}$	$\{2\}$	$\{2\}$	$\{2\}$	$\{2\}$	$\{2\}$	$\emptyset$	$\emptyset$
$\mathrm{ID}_v(H^5)$	$\emptyset$	$\{2\}$	I	I	$\{1\}$	I	I	I	I	$\{1\}$	I	I
$\mathrm{ID}_v(H^6)$	$\{2\}$	$\{2\}$	$\{2\}$	I	I	I	$\{1\}$	I	$\{1\}$	I	I	I
$\mathrm{ID}_v(H^7)$	$\{2\}$	$\{2\}$	I	$\{2\}$	I	$\{1\}$	I	$\{1\}$	I	I	I	I

Tableau 5.4 Computing $\mathrm{ID}_v(\{x\})$ for all states $v \in Y_m[1]$ and $x \in X_m$.

Then, by using the functions $\mathrm{ID}_v(.)$, for every $v \in Y_m[1]$, and Equation (5.21), we compute the sets $\mathrm{Elig}(x)$ for every marked state $x \in X_m$:

$x = H^1$: We compute

$$A = X_m \cap \bigcup_{\substack{v \in Y_m[1]: \\ \mathrm{ID}_v(H^1)=\emptyset \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcup_{i \in I} v_i = \emptyset, \quad B = X_m \cap \bigcup_{\substack{v \in Y_m[1]: \\ \mathrm{ID}_v(H^1)\neq I \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcap_{i \in \mathrm{ID}_v(H^1)} v_i = \emptyset.$$

Then, we obtain $\mathrm{Elig}(H^1) = X_m \setminus [\{H^1\} \cup A \cup B] = \{H^2, H^3, H^4, H^5, H^6, H^7, H^8\}$.

$x = H^2$: We compute

$$A = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_\iota(H^2)=\emptyset \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcup_{\iota \in I} v_\iota = \emptyset, \quad B = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(H^2)\neq I \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcap_{\iota \in \mathrm{ID}_\iota(H^2)} v_\iota = \emptyset.$$

Then, we obtain $\mathrm{Elig}(H^2) = X_m \setminus [\{H^2\} \cup A \cup B] = \{H^1, H^3, H^4, H^5, H^6, H^7, H^8\}$.
$x = H^3$: We compute

$$A = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(H^3)=\emptyset \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcup_{\iota \in I} v_\iota = \emptyset, \quad B = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(H^3)\neq I \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcap_{\iota \in \mathrm{ID}_v(H^3)} v_\iota = \emptyset.$$

Then, we obtain $\mathrm{Elig}(H^3) = X_m \setminus [\{H^3\} \cup A \cup B] = \{H^1, H^2, H^4, H^5, H^6, H^7, H^8\}$.
$x = H^4$: We compute

$$A = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(H^4)=\emptyset \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcup_{\iota \in I} v_\iota = \{H^4\}, B = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(H^4)\neq I \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcap_{\iota \in \mathrm{ID}_\iota(H^4)} v_\iota = \{H^6, H^7\}.$$

Then, we obtain $\mathrm{Elig}(H^4) = X_m \setminus [\{H^4\} \cup A \cup B] = \{H^1, H^2, H^3, H^5\}$.
$x = H^5$: We compute

$$A = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(H^5)=\emptyset \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcup_{\iota \in I} v_\iota = \emptyset, \quad B = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(H^5)\neq I \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcap_{\iota \in \mathrm{ID}_v(H^5)} v_\iota = \emptyset.$$

Then, we obtain $\mathrm{Elig}(H^5) = X_m \setminus [\{H^5\} \cup A \cup B] = \{H^1, H^2, H^3, H^4, H^6, H^7\}$.
$x = H^6$: We compute

$$A = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(H^6)=\emptyset \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcup_{\iota \in I} v_\iota = \emptyset, \quad B = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(H^6)\neq I \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcap_{\iota \in \mathrm{ID}_v(H^6)} v_\iota = \{H^4\}.$$

Then, we obtain $\mathrm{Elig}(H^6) = X_m \setminus [\{H^6\} \cup A \cup B] = \{H^1, H^2, H^3, H^5, H^7\}$.
$x = H^7$: We compute

$$A = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_\iota(H^7)=\emptyset \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcup_{\iota \in I} v_\iota = \emptyset, \quad B = X_m \cap \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(H^7)\neq I \\ \mathcal{C}_f(v)\neq\emptyset}} \bigcap_{\iota \in \mathrm{ID}_v(H^7)} v_\iota = \{H^4\}.$$

Then, we obtain $\text{Elig}(H^7) = X_m \setminus [\{H^7\} \cup A \cup B] = \{H^1, H^2, H^3, H^5, H^6\}$.

Let us now explain the construction of each X_m^j, using the automaton $\mathcal{A}_{\mathcal{F}[1]}$ of Fig. 5.4 and the following intuitive expression of Cond. (5.18) : The latter requires that for every $v = (v_1, v_2, w) \in Y_m[1]$ (such that $\mathcal{C}_f(v) \neq \emptyset$), $\forall j \in J$, if both parts 1 and 2 of v (i.e., v_1 and v_2) contain states of X_m^j, then all these parts contain in fact the same single state of X_m^j and no other state of X_m^j.

Construction of X_m^1 : Initially $X_m^1 = \emptyset$ and $Z_m = X_m$. We select $H^1 \in \text{Elig}(\emptyset) \cap Z_m = Z_m$ and move it to X_m^1, we obtain $X_m^1 = \{H^1\}$ and $Z_m = \{H^2, H^3, H^4, H^5, H^6, H^7\}$. We compute $\text{Elig}(X_m^1) \cap Z_m = \text{Elig}(H^1) \cap Z_m = \{H^2, H^3, H^4, H^5, H^6, H^7\}$. Therefore, every state of Z_m can be added to X_m^1 without violating Cond. (5.18). For example, we select to add H^2 and obtain $X_m^1 = \{H^1, H^2\}$ and we have $Z_m = \{H^3, H^4, H^5, H^6, H^7\}$. Then, we compute $\text{Elig}(X_m^1)$ (Proposition 5.8.4) :

$$\text{Elig}(X_m^1) = \text{Elig}(H^1) \cap \text{Elig}(H^2) \setminus [\bigcup_{\substack{v \in Y_m[1] \\ \text{ID}_v(\{H^1,H^2\}) \neq I \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcap_{i \in \text{ID}_v(\{H^1,H^2\})} v_i].$$

Then, by using Lemma 5.8.4, we have, $\forall v \in Y_m[1]$ $\text{ID}_v(X_m^1) = \text{ID}_v(\{H^1) \cap \text{ID}_v(H^2)$, and then

$$\text{Elig}(X_m^1) = \text{Elig}(\{H^1, H^2\}) = \{H^3, H^4, H^5, H^6, H^7\}.$$

Let us select $H^3 \in \text{Elig}(\{H^1, H^2\}) \cap Z_m$ and move it to X_m^1. We obtain $X_m^1 = \{H^1, H^2, H^3\}$ and $Z_m = \{H^4, H^5, H^6, H^7\}$. Then, we compute $\text{Elig}(X_m^1)$ (Proposition 5.8.4) :

$$\text{Elig}(X_m^1) = \text{Elig}(\{H^1, H^2\}) \cap \text{Elig}(H^3) \setminus [\bigcup_{\substack{v \in Y_m[1] \\ \text{ID}_v(X_m^1) \neq I \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcap_{i \in \text{ID}_v(X_m^1)} v_i].$$

Then, by using Lemma 5.8.4, we have, $\forall v \in Y_m[1]$, $\text{ID}_v(X_m^1) = \text{ID}_v(H^1, H^2) \cap \text{ID}_v(H^3)$, and then

$$\text{Elig}(X_m^1) = \text{Elig}(\{H^1, H^2, H^3\}) = \{H^4, H^5, H^6, H^7\}.$$

Let us select $H^4 \in \text{Elig}(\{H^1, H^2, H^3\}) \cap Z_m$ and move it to X_m^1. We obtain $X_m^1 = \{H^1, H^2, H^3, H^4\}$ and $Z_m = \{H^5, H^6, H^7\}$. Then, we compute $\text{Elig}(X_m^1)$ (Proposition 5.8.4) :

$$\text{Elig}(X_m^1) = \text{Elig}(\{H^1, H^2, H^3\}) \cap \text{Elig}(H^4) \setminus [\bigcup_{\substack{v \in Y_m[1] \\ \text{ID}_v(X_m^1) \neq I \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcap_{i \in \text{ID}_v(X_m^1)} v_i].$$

Then, by using Lemma 5.8.4, we have, $\forall v \in Y_m[1]$ $\text{ID}_v(X_m^1) = \text{ID}_v(H^1, H^2, H^3) \cap \text{ID}_v(H^4)$, and then

$$\text{Elig}(X_m^1) = \text{Elig}(\{H^1, H^2, H^3, H^4\}) = \{H^5\} \setminus \{H^6, H^7\} = \{H^5\}.$$

Hence H^5 is the only state that can be moved to X_m^1. Then, we obtain $X_m^1 = \{H^1, H^2, H^3, H^4, H^5\}$ and $Z_m = \{H^6, H^7\}$.

Construction of X_m^2 : Initially $X_m^2 = \emptyset$ and $Z_m = \{H^6, H^7\}$. We select $H^6 \in \text{Elig}(\emptyset) \cap Z_m = Z_m$ and move it to X_m^2, we obtain $X_m^2 = \{H^6\}$ and $Z_m = \{H^7\}$. We compute $\text{Elig}(X_m^2) \cap Z_m = \text{Elig}(H^6) \cap Z_m = \{H^7\}$. Therefore, H^7 can be moved to X_m^2. Then, we obtain $X_m^2 = \{H^6, H^7\}$ and $Z_m = \emptyset$.

To recapitulate, we have $X_m^1 = \{H^1, H^2, H^3, H^4, H^5\}$ and $X_m^2 = \{H^6, H^7\}$. We will thus apply the double-marking : $(X_m^j)_{j=1,2}$ to the marked states of $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$ as explained in Subsection 5.8.3. Then, we check how the X_m^j-marking is propagated to $\mathcal{A}_{\mathcal{F}[1]}$ of Fig. 5.4. Note that with our representation of states, a marked state $u = (u_1, u_2, u_3)$ of $\mathcal{A}_{\mathcal{F}[1]}$ (i.e., $u \in Y_m[1]$) is X_m^j-marked if both coordinates u_1 and u_2 contain states of X_m^j. We thus understand visually that there is no X_m^2-marked state, and among the marked states $v \in Y_m[1]$ such that $\mathcal{C}_f(v) \neq \emptyset$, we have F_1^{16} which is X_m^1-marked. In fact, F_1^{16} is reached by the faulty traces $a_1 f b_2 e e^*$ through the self-loop (labeled by e) on F_1^{16}. Hence, $(\mathcal{F}, \mathcal{H}^1)$ is *not* Inf_0-CODIAG, and $(\mathcal{F}, \mathcal{H}^2)$ is Inf_0-CODIAG. We now have to check whether $(\mathcal{F}, \mathcal{H}^1)$ is Inf_1-CODIAG. For that purpose, we compute $\mathcal{A}_{\mathcal{F}[2]}$ (see Fig. 5.5) and check if it has X_m^1-marked states $v = (v_1, v_2, v_3)$, that is, states whose both first and second coordinates v_1 and v_2 contain H_1^{10}, H_1^{11} or H_1^{13} and whose third coordinate v_3 is F_1^{16} (because F_1^{16} is the only X_m^1-marked state of $\mathcal{A}_{\mathcal{F}[1]}$, while H_1^{10}, H_1^{11} and H_1^{13} are the X_m^1-marked states of $\mathcal{A}_{\mathcal{H}[1]}$). The marked states of $\mathcal{A}_{\mathcal{F}[2]}$ are presented in the form $F_2^j = (H_1^{p_1, p_2, \cdots}, H_1^{q_1, q_2, \cdots}, F_1^r)$, where $H_1^{\nu_1, \nu_2, \cdots}$ denotes the set $\{H_1^{\nu_1}, H_1^{\nu_2}, \ldots\} \subseteq X_m[1]$ and $F_1^r \in Y_m[1]$. In fact, F_2^7 is the

only state that is X_m^1-marked of $\mathcal{A}_{\mathcal{F}[2]}$. We have $\mathcal{C}_f(F_2^7) \neq \emptyset$, this implies that, $\forall l \geq 1$, $\mathcal{F}^1[2] \cap \mathcal{F}_l \neq \emptyset$, i.e., $(\mathcal{F}, \mathcal{H}^1)$ is not Inf_1-CODIAG.

In order to check if $(\mathcal{F}, \mathcal{H}^1)$ is Inf_2-CODIAG, we have to check if $\mathcal{A}_{\mathcal{F}[3]}$ contains X_m^1-marked states reached through cycles of faulty states. Before that, we need to compute $\mathcal{A}_{\mathcal{H}[2]}$ represented in Fig. 5.6. The marked states of $\mathcal{A}_{\mathcal{H}[2]}$ are presented in the form $H_2^j = (F_1^{p_1,p_2,\cdots}, F_1^{q_1,q_2,\cdots}, H_1^r)$, where $F_1^{\nu_1,\nu_2,\cdots}$ denotes the set $\{F_1^{\nu_1}, F_1^{\nu_2}, \ldots\} \subseteq Y_m[1]$ and $H_1^r \in X_m[1]$. All the marked states $v = (v_1, v_2, v_3) \in X_m[2]$ such that $\mathcal{C}_f(v) \neq \emptyset$, i.e., H_2^6, H_2^8, H_2^9, H_2^{12} and H_2^{13}, are not X_m^1-marked , that is, none of them is a state whose both first and second coordinates v_1 and v_2 contain F_1^{16} and whose third coordinate v_3 is H_1^{10}, H_1^{11} or H_1^{13}. This implies that in $\mathcal{A}_{\mathcal{F}[3]}$ there is no cycle of faulty states through which an X_m^1-marked state is reached. In fact, we have seen in Example 5.5.1 that $\mathcal{F}^1[3] = \{a_1 f \overline{b_2}, b_1 f, d_1 f \overline{a_2}\}$, and then $\mathcal{F}^1[3] \cap \mathcal{F}_2 = \emptyset$. Therefore, $(\mathcal{F}, \mathcal{H}^2)$ is Inf_2-CODIAG.

To recapitulate, $(\mathcal{F}, \mathcal{H}^1)$ and $(\mathcal{F}, \mathcal{H}^2)$ are Inf_0-F-CODIAG and Inf_2-CODIAG, respectively. That is, $(\mathcal{F}, \mathcal{H})$ is $\wedge$-(Inf_0, Inf_2)-CODIAG.

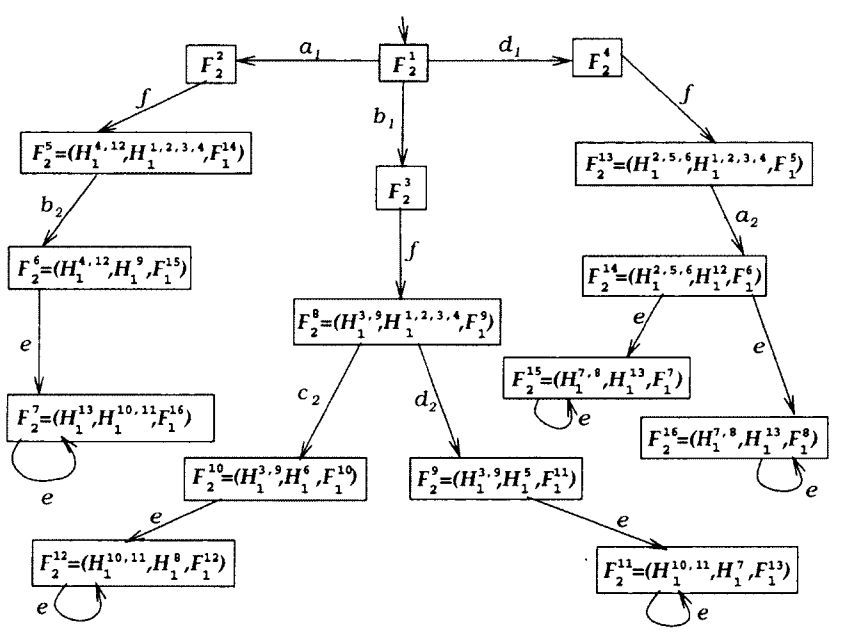

Figure 5.5 $\mathcal{A}_{\mathcal{F}[2]}$ computed from $\mathcal{A}_{\mathcal{H}[1]}$ and $\mathcal{A}_{\mathcal{F}[1]}$ of Figures 5.3 and 5.4

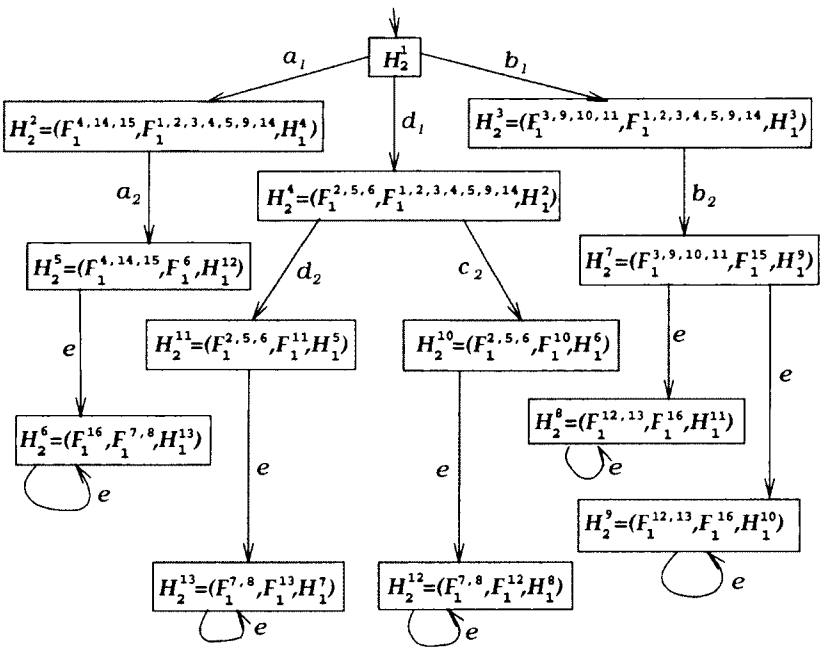

Figure 5.6 $\mathcal{A}_{\mathcal{H}[2]}$ computed from $\mathcal{A}_{\mathcal{H}[1]}$ and $\mathcal{A}_{\mathcal{F}[1]}$ of Figures 5.3 and 5.4

5.11 Conclusion

We have developed a new framework, called multi-decision diagnosis, that is intended to be applicable to any existing decentralized architecture in order to generalize the latter. The basic principle of multi-decision diagnosis consists in using several decentralized diagnosers $Diag^j$ ($j \in J$) running in parallel. Each decentralized diagnoser $Diag^j$ performs its local and global diagnoses according to a given decentralized architecture D^j. The global diagnoses of all decentralized diagnosers $Diag^j$ are then fused using a coordinating module $\mathbf{D}$ in order to obtain the effective diagnosis. We have then defined the notion of multi-decision diagnoser $Diag = ((Diag^j)_{j\in J}, \mathbf{D})$ that has for goal to diagnose the plant in order to detect faulty transitions.

We have studied more thoroughly the cases where the effective diagnosis is obtained by combining the global diagnoses of all decentralized diagnosers $Diag^j$ ($j \in \{1, \ldots, p\}$) either conjunctively ($\mathbf{D} = \vee$) or disjunctively ($\mathbf{D} = \wedge$). In the case $\mathbf{D} = \vee$ (resp., $\mathbf{D} = \wedge$) we use a decomposition $(\mathcal{F}^1, \ldots, \mathcal{F}^p)$ of $\mathcal{F}$ (resp., $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ of $\mathcal{H}$) such that each decentralized diagnoser $Diag^j$ makes its diagnoses based on $(\mathcal{F}^j, \mathcal{H})$ (resp., $(\mathcal{F}, \mathcal{H}^j)$).

In the case $\mathbf{D} = \vee$, we have shown that the existence of a decomposition $(\mathcal{F}^1, \cdots, \mathcal{F}^p)$ of $\mathcal{F}$ such that each decentralized diagnoser $Diag^j$ has no missed and no false detection, i.e.,

satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}^j, \mathcal{H})$, is a necessary and sufficient condition for the multi-decision diagnoser $Diag = ((Diag^j)_{j \in J}, \vee)$ to have no missed and no false detection, i.e., satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H})$. And in the case $\mathbf{D} = \wedge$, we have shown that the existence of a decomposition $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ of $\mathcal{H}$ such that each decentralized diagnoser $Diag^j$ has no missed and no false detection, i.e., satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H}^j)$, is a necessary and sufficient condition for the multi-decision diagnoser $Diag = ((Diag^j)_{j \in J}, \wedge)$ to have no missed and no false detection, i.e., satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H})$.

We have studied in more detail the multi-decision diagnosis in the case of $\wedge$-$Inf_{\leq N}^{\geq 1}$ architecture where several (say p) inference-based Inf_{N_j}-diagnosers ($j = 1 \cdots p$) running in parallel and whose global diagnoses are fused conjunctively (i.e., $\mathbf{D} = \wedge$). The obtained diagnoser is then called $\wedge$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-diagnoser. We have defined and studied the notion of $\wedge$-$(Inf_{N_1}, \cdots, Inf_{N_p})$-F-CODIAG, which is useful to characterize the class of languages that can be diagnosed correctly by a multi-decision diagnoser.

A difficult problem inherent to the multi-decision approach is the decomposition of infinite languages. We solve this problem by using an assumption on the decomposition by transforming the problem of decomposing an infinite regular language ($\mathcal{F}$ or $\mathcal{H}$) into a problem of decomposing the finite state set of an automaton accepting the regular language in question. We thus define the decidable notion of $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG w.r.t. some FSA accepting $\mathcal{H}$.

We have also defined the more applicable notion of $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG for which $N_1, \ldots, N_p$ are bounded by N. Then, we have developed a method and its corresponding algorithm which checks if a given specification is $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG. We show that with our algorithm, the worst-case computational complexity for checking $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG is not increased by the number of decentralized diagnosers in parallel. That is, we obtain the same order of complexity as in the inference-based framework of [Kumar et Takai, 2009].

As a future work, we will investigate more efficient methods for obtaining decidable versions of the multi-decision framework. We will investigate if it is possible to select systematically a FSA $\mathcal{A}_{\mathcal{H}}$ that enhances the performance of multi-decision, based on some criteria to be determined. We sill also investigate if it is possible to solve the decomposition problem without using FSAs.

CHAPITRE 6

Conclusion

Dans ce dernier chapitre, on résume les principales contributions de cette thèse et on termine avec quelques perspectives pour des travaux futurs.

6.1 Contributions

Dans cette thèse, on a développé une approche multi-décisionnelle de prise de décision pour le contrôle et le diagnostic décentralisés de SED. On a proposé deux démarches pour étudier les architectures multi-décisionnelles. La première démarche, et qui présente la réalité physique de l'architecture multi-décisionnelle, considère un seul décideur décentralisé dont les décideurs locaux prennent un n-uplet de décisions au lieu d'une seule décision, d'où le nom "architecture multi-décisionnelle". L'autre démarche considère plusieurs décideurs (superviseurs ou diagnostiqueurs) décentralisés qui fonctionnent simultanément et en parallèle, les décisions globales des décideurs décentralisés sont fusionnées afin d'obtenir une décision effective. Cette deuxième démarche peut être considérée comme une "méta-théorie" dans le sens où elle permet d'utiliser différentes architectures décentralisées existantes, qu'on a nommé "éligibles", pour former une architecture plus générale. Cela nous a permis d'utiliser les résultats existants des architectures décentralisées, comme les conditions d'existences des décideurs décentralisés, pour établir des résultats concernant l'architecture multi-décisionnelle qui généralisent ceux des architectures décentralisées qui forment celle-ci.

Nous avons étudié les cas où la décision effective est obtenue en fusionnant disjonctivement ou conjonctivement les décisions globales des différents décideurs décentralisés. Ainsi, dans cette thèse, on a appliqué l'approche multi-décisionnelle aux architectures C&P, D&A, C&P$\vee$D&A, conditionnelles et non conditionnelles et aux architectures utilisant la technique de l'inférence par ambiguïté. On a montré que l'approche multi-décisionnelle généralise toutes ces architectures. Plus précisément, on a montré que si une architecture décentralisée A coopère avec d'autres architectures décentralisées pour former une architecture multi-décisionnelle B, alors la classe de langages réalisables ou diagnostiquables par B englobe celle des langages réalisables ou diagnostiquables par A.

Le principe de l'approche multi-décisionnelle dans la prise de décision dans le contrôle des SED (le contrôle multi-décisionnel) consiste à utiliser plusieurs (disons p) superviseurs décentralisés $(Sup^j)_{j=1,\ldots,p}$ fonctionnant en parallèle pour former un superviseur multi-décisionnel. Chaque superviseur décentralisé Sup^j prend les décisions locales et globales en appliquant les règles d'une architecture décentralisée DD^j. Les décisions globales de tous les superviseurs décentralisés $(Sup^j)_{j=1,\ldots,p}$ sont ensuite fusionnées selon une fonction $\mathbf{D}$ afin d'obtenir une décision effective du superviseur multi-décisionnel qui sera appliquée au procédé. Nous avons étudié les cas où la décision effective est obtenue en fusionnant les décisions globales des différents p superviseurs décentralisés, disjonctivement ($\mathbf{D} = \vee$) ou conjonctivement ($\mathbf{D} = \wedge$). Dans le cas $\mathbf{D} = \vee$, pour chaque $\sigma \in \Sigma_c$, on utilise une décomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ de $\mathcal{E}_\sigma$ telle que chaque superviseur décentralisé Sup^j prend sa décision en se basant sur le couple $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ au lieu de $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$. Dans le cas $\mathbf{D} = \wedge$, pour chaque $\sigma \in \Sigma_c$, on utilise une décomposition $\{\mathcal{D}_\sigma^1, \ldots, \mathcal{D}_\sigma^p\}$ de $\mathcal{D}_\sigma$ telle que chaque superviseur décentralisé Sup^j prend sa décision en se basant sur le couple $(\mathcal{E}_\sigma, \mathcal{D}_\sigma^j)$ au lieu de $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$. Dans le cas $\mathbf{D} = \vee$, nous avons défini la notion de $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS, qui est utile pour caractériser la classe de langages réalisables sous l'architecture $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$. En plus, nous avons montré que la classe de langages réalisables sous l'architecture $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$ englobe la classe de langages réalisables sous l'architecture D^j, pour chaque $j \in \{1, \ldots, p\}$.

Le principe de l'approche multi-décisionnelle dans la prise de décision dans le diagnostic des SED (ou le diagnostic multi-décisionnel) est basé sur l'utilisation de plusieurs (disons p) diagnostiqueurs décentralisés $(Diag^j)_{j=1,\ldots,p}$ fonctionnant en parallèle pour former un diagnostiqueur multi-décisionnel. Chaque diagnostiqueur décentralisé $Diag^j$ émet des diagnostics locaux et globaux en appliquant les règles d'une architecture décentralisée décentralisée DD^j. Les diagnostics globaux de tous les diagnostiqueurs décentralisés $(Diag^j)_{j=1,\ldots,p}$ sont ensuite fusionnés selon une fonction binaire $\mathbf{D}$ afin d'obtenir un diagnostic effectif du diagnostiqueur multi-décisionnel. Nous avons étudié les cas où le diagnostic effectif est obtenu en fusionnant les diagnostics globaux des différents diagnostiqueurs décentralisés, disjonctivement ($\mathbf{D} = \vee$) ou conjonctivement ($\mathbf{D} = \wedge$). Dans le cas $\mathbf{D} = \vee$, on utilise une décomposition $(\mathcal{F}^1, \ldots, \mathcal{F}^p)$ de $\mathcal{F}$ telle que chaque diagnostiqueur décentralisé $Diag^j$ fait un diagnostic en se basant sur le couple $(\mathcal{F}^j, \mathcal{H})$ au lieu de $(\mathcal{F}, \mathcal{H})$. Dans le cas $\mathbf{D} = \wedge$, on utilise une décomposition $(\mathcal{H}^1, \ldots, \mathcal{H}^p)$ de $\mathcal{H}$ telle que chaque diagnostiqueur décentralisé $Diag^j$ fait un diagnostic en se basant sur le couple $(\mathcal{F}, \mathcal{H}^j)$ au lieu de $(\mathcal{F}, \mathcal{H})$. Nous avons défini et étudié la notion de $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-CODIAG (resp., $\vee$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-CODIAG) pour caractériser la classe de langages diagnostiquables selon l'architecture $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$ (resp., $\vee$-$(Inf_{N_1}, \ldots, Inf_{N_p})$).

Comme la condition d'existence d'un décideur multi-décisionnel est basée sur des décompositions de langages, se pose alors le problème de la décomposition des langages infinis. On a pu résoudre ce problème dans le cas des langages réguliers en proposant une méthode qui transforme la décomposition d'un langage régulier infini X (où X représente $\mathcal{E}_\sigma$, $\mathcal{D}_\sigma$, $\mathcal{F}$ ou $\mathcal{H}$) en une décomposition d'ensemble d'états marqués finis d'un AEF $\mathcal{A}_X$ acceptant X. Ainsi, des conditions plus fortes pour l'existence des déciseurs (superviseurs et diagnostiqueurs) multi-décisionnels ont été définies. Plus précisément, en plus de la condition de $\vee$-($\mathrm{D}^1, \ldots, \mathrm{D}^p$)-COOBS (ou $\wedge$-($\mathrm{D}^1, \ldots, \mathrm{D}^p$)-COOBS) dans le cas du contrôle et $\vee$-($\mathrm{D}^1, \ldots, \mathrm{D}^p$)-CODIAG (ou $\wedge$-($\mathrm{D}^1, \ldots, \mathrm{D}^p$)-CODIAG) dans le cas du diagnostic, on a ajouté une condition que chaque décomposition $(X^1, \ldots, X^p)$ de X doit satisfaire, à savoir, chaque X^j contient toutes les traces qui mènent à un ou plusieurs états marqués de $\mathcal{A}_X$. Ces conditions ont l'avantage d'être vérifiables et moins contraignantes que les conditions d'existence pour les architectures décentralisées utilisées en parallèle pour former une architecture multi-décisionnelle.

On a proposé une méthode (et l'algorithme correspondant) qui vérifie si $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ est $\vee$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-COOBS (resp., $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-COOBS) par rapport à un AEF $\mathcal{A}_{\mathcal{E}_\sigma}$ (resp., $\mathcal{A}_{\mathcal{D}_\sigma}$) donné acceptant $\mathcal{E}_\sigma$ (resp., $\mathcal{D}_\sigma$). De même, on a proposé une méthode (et l'algorithme correspondant) qui vérifie si $(\mathcal{F}, \mathcal{H})$ est $\vee$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-CODIAG (resp., $\wedge$-($Inf_{N_1}, \ldots, Inf_{N_p}$)-CODIAG) par rapport à un AEF $\mathcal{A}_{\mathcal{F}}$ (resp., $\mathcal{A}_{\mathcal{H}}$) donné acceptant $\mathcal{F}$ (resp., $\mathcal{H}$). Nous avons montré que la complexité des algorithmes de la vérification de la coobservabité ou de la diagnosticabilité est, dans le pire cas, du même ordre de grandeur que la complexité de vérification de la coobservabité ou de la diagnosticabilité selon une des architectures décentralisées en parallèle qui constituent l'architecture multi-décisionnelle.

La solution que nous avons proposé permet une décomposition finie d'un langage régulier infini X en procédant à des décompositions d'états d'un AEF A_X qui accepte X. Cependant, cette solution laisse la condition d'existence des superviseurs et des diagnostiqueurs dépendante de la structure des AEF et non du langage en lui même. Et ainsi pour deux AEF acceptant le même langage, on peut avoir une solution pour un AEF et pas de solution pour l'autre. Ainsi, étant donné un langage X (où, $X = \mathcal{E}_\sigma$, $\mathcal{D}_\sigma$, $\mathcal{F}$ ou $\mathcal{H}$), on peut se demander si pour un AEF A_X acceptant X, pour qui la coobservabilité (ou la codiagnosabilité) multi-décisionnelle n'est pas satisfaite, est-il possible de transformer A_X en un autre AEF A'_X acceptant X pour qui la condition de coobservabilité (ou de codiagnosabilité) multi-décisionnelle est satisfaite ? Nous avons proposé dans le chapitre 2 une solution qui permet une transformation dans ce genre, mais il reste que celle-ci est de complexité exponentielle en terme du nombre des décideurs locaux.

6.2 Perspectives

L'approche multi-décisionnelle est une nouvelle méthode qui ouvre la voie à plusieurs perspectives dans le domaine de la prise de décision décentralisée. Parmi ces perspectives on trouve, entre autres,

exploiter d'autres alternatives concernant la fusion des décisions globales des décideurs décentralisés. En effet, à part les deux fonctions booléennes utilisées dans cette thèse, conjonctive et disjonctive, d'autres fonctions peuvent être étudiées, par exemple des fonctions d'exclusion ou des fonctions avec mémoire.

L'application de l'approche multi-décisionnelle dans le test et la vérification des SED.

Chercher d'autres alternatives que la décomposition des langages pour effectuer des prises de décisions multi-décisionnelles.

Trouver une réponse à la question pertinente suivante : est ce qu'il y a d'autres approches qui peuvent être plus générales que l'approche multi-décisionnelle ?

Résoudre le problème de décomposition des langages infinis par d'autres méthodes que celle utilisée dans cette thèse. Avant d'adopter la méthode de décomposition d'ensemble d'états, on a tenté plusieurs méthodes qui n'ont pas abouti à des résultats satisfaisants. Nous comptons dans un futur proche donner suite à nos tentatives pour résoudre ce problème.

En plus de l'aspect théorique, nous comptons appliquer le principe multi-décisionnel aux systèmes présentant une architecture décentralisée ou distribuée. Parmi ces applications, on peut citer comme exemple : la détection et la résolution des interactions de services dans les systèmes de télécommunications et les systèmes réseautiques. La détection et la résolution d'interactions seront respectivement abordées par le contrôle et le diagnostic.

Comme aspect technique, nous comptons développer un logiciel qui implémente l'algorithme présenté dans cette thèse qui vérifie la codiagnostiquabilité d'une spécification donnée avec l'architecture multi-décisionnelle.

Génération de code de contrôleur et de diagnostiqueur multi-décisionnels à partir d'une spécification formelle du contrôleur ou du diagnostiqueur, c.à.d. :

On donne en entrée : les expressions mathématiques de Sup_i^J (ou $Diag_i^J$), D^J et $\mathbf{D}$.

On obtient en sortie les codes (par exemple en C++) qui implémentent chacune de ces fonctions.

ANNEXE A

Preuves du chapitre 3

A.1 Proofs of Section 3.4

A.1.1 Proof of Lemma 3.4.1

Consider a supervisor Sup and $K \subseteq \mathcal{L}_m$. Let us prove that points 1 and 2 are equivalent.

Proof of $1 \Rightarrow 2$ **:** We assume that Sup is nonblocking, admissible w.r.t. K and such that $\mathcal{L}(Sup/G) = \overline{K}$ and $\mathcal{L}_m(Sup/G) = K$, and we have to prove that K is controllable, $\mathcal{L}_m$-closed and Sup is consistent w.r.t. K.

Controllability of K **:** Consider $s \in \overline{K}$ and $\sigma \in \Sigma_{uc}$ such that $s\sigma \in \mathcal{L}$. Since any supervisor enables all the uncontrollable events, we have $Sup(s,\sigma) = 1$. From the definition of $\mathcal{L}(Sup/G)$ and the fact that $Sup(s,\sigma) = 1$, $s \in \overline{K}$, $\mathcal{L}(Sup/G) = \overline{K}$ and $s\sigma \in L$, we deduce $s\sigma \in \mathcal{L}(Sup/G) = \overline{K}$. We have thus proved that $\overline{K}\Sigma_{uc} \cap \mathcal{L} \subseteq \overline{K}$, that is K is controllable.

$\mathcal{L}_m$**-closure of** K **:** $K = \mathcal{L}_m(Sup/G) = \mathcal{L}(Sup/G) \cap \mathcal{L}_m = \overline{K} \cap \mathcal{L}_m$.

Sup **is consistent w.r.t.** K **:** We have to show that K satisfies Conds. (3.3) and (3.4) w.r.t. the admissible and nonblocking D-supervisor Sup. Let $\sigma \in \Sigma_c$ and $s \in \mathcal{E}_\sigma$, which implies that $s \in \overline{K}$ and $s\sigma \in \overline{K}$, and then $s\sigma \in \mathcal{L}$. Hence from the definition of $\mathcal{L}(Sup/G)$ and the fact that $\overline{K} = \mathcal{L}(Sup/G)$, we deduce that $Sup(s,\sigma) = 1$, and thus Cond. (3.3) is satisfied. Assume that $s \in \mathcal{D}_\sigma$, which implies that $s \in \overline{K}$, $s\sigma \in \mathcal{L}$ and $s\sigma \notin \overline{K}$. Hence from the definition of $\mathcal{L}(Sup/G)$ and the fact that $\overline{K} = \mathcal{L}(Sup/G)$, we deduce that $Sup(s,\sigma) \neq 1$ (i.e., $\in \{0, \phi\}$), and since Sup is admissible we have $Sup(s,\sigma) = 0$, and thus Cond. (3.4) is satisfied.

Proof of $2 \Rightarrow 1$ **:** We assume that K is controllable, $\mathcal{L}_m$-closed and Sup is consistent w.r.t. K. We have to prove that Sup is nonblocking, admissible such that $\mathcal{L}(Sup/G) = \overline{K}$ and $\mathcal{L}_m(Sup/G) = K$. From the consistency of Sup with K (Def. 3.3.2), for every $\sigma \in \Sigma_c$, Sup satisfies Conds. (3.3) and (3.4) w.r.t. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$. Let us prove that this supervisor Sup is admissible, nonblocking, and such that $\mathcal{L}(Sup/G) = \overline{K}$ and $\mathcal{L}_m(Sup/G) = K$.

Proof that Sup **is admissible :** From Conds. (3.3) and (3.4), we have $Sup(s,\sigma) \in \{0,1\}$ for $s \in \mathcal{E}_\sigma \cup \mathcal{D}_\sigma$, and thus, $Sup(s,\sigma) \neq \phi$ for $s \in \mathcal{E}_\sigma \cup \mathcal{D}_\sigma$.

Proof that $\mathcal{L}(Sup/G) = \overline{K}$ **:** Let us present a proof by induction on the length of traces.

Basis : We have $\varepsilon \in \mathcal{L}(Sup/G)$ and $\varepsilon \in \overline{K}$.

Inductive step : The aim is to prove that if $s \in \mathcal{L}(Sup/G)$ and $s \in \overline{K}$, then $\forall \sigma \in \Sigma$ s.t. $s\sigma \in \mathcal{L}$, we have : $s\sigma \in \mathcal{L}(Sup/G)$ iff $s\sigma \in \overline{K}$. Assume $s \in \mathcal{L}(Sup/G)$ and $s \in \overline{K}$, and

consider $\sigma \in \Sigma$ s.t. $s\sigma \in \mathcal{L}$, that is $s \in \mathcal{E}_\sigma \cup \mathcal{D}_\sigma$. Consider the two possible cases : $\sigma \in \Sigma_{uc}$ and $\sigma \in \Sigma_c$,

$\sigma \in \Sigma_{uc}$: Hence $Sup(s,\sigma) = 1$. From the definition of $\mathcal{L}(Sup/G)$ and the fact that $s \in \mathcal{L}(Sup/G)$, $s\sigma \in \mathcal{L}$ and $Sup(s,\sigma) = 1$, we deduce that $s\sigma \in \mathcal{L}(Sup/G)$. And from the controllability of K, we deduce that $s\sigma \in \overline{K}$.

$\sigma \in \Sigma_c$: Let us show that : $s\sigma \in \mathcal{L}(Sup/G)$ iff $s\sigma \in \overline{K}$.

Proof of $s\sigma \in \mathcal{L}(Sup/G) \Rightarrow s\sigma \in \overline{K}$ **:** Assume that $s\sigma \in \mathcal{L}(Sup/G)$. From the definition of $\mathcal{L}(Sup/G)$ and the fact that $s \in \mathcal{L}(Sup/G)$, $s\sigma \in \mathcal{L}$ and $s\sigma \in \mathcal{L}(Sup/G)$, we deduce that $Sup(s,\sigma) = 1$. Since $s \in \mathcal{E}_\sigma \cup \mathcal{D}_\sigma$ and $\mathcal{E}_\sigma \cap \mathcal{D}_\sigma = \emptyset$, we have either $s \in \mathcal{E}_\sigma$ or $s \in \mathcal{D}_\sigma$. Assume that $s \in \mathcal{D}_\sigma$, then by Cond. (3.4) we deduce $Sup(s,\sigma) = 0$, which is in contradiction with the previous deduction $Sup(s,\sigma) = 1$. Hence, $s \in \mathcal{E}_\sigma$, i.e., $s\sigma \in \overline{K}$. We have therefore shown that for $\sigma \in \Sigma_c$, if $s\sigma \in \mathcal{L}(Sup/G)$ then $s\sigma \in \overline{K}$.

Proof of $s\sigma \in \overline{K} \Rightarrow s\sigma \in \mathcal{L}(Sup/G)$ **:** Assume that $s\sigma \in \overline{K}$, i.e., $s \in \mathcal{E}_\sigma$, then by Cond. (3.3) we have $Sup(s,\sigma) = 1$. Hence, from the definition of $\mathcal{L}(Sup/G)$ and the fact that $s \in \mathcal{L}(Sup/G)$, $s\sigma \in \mathcal{L}$ and $Sup(s,\sigma) = 1$, we deduce that $s\sigma \in \mathcal{L}(Sup/G)$. We have therefore shown that for $\sigma \in \Sigma_c$, if $s\sigma \in \overline{K}$ then $s\sigma \in \mathcal{L}(Sup/G)$.

We have therefore proved by induction that $\overline{K} = \mathcal{L}(Sup/G)$.

Proof that $\mathcal{L}_m(Sup/G) = K$ **:** By the $\mathcal{L}_m$-closure of K, the fact that $\overline{K} = \mathcal{L}(Sup/G)$ and the definition of $\mathcal{L}_m(Sup/G)$, we have $K = \overline{K} \cap \mathcal{L}_m = \mathcal{L}(Sup/G) \cap \mathcal{L}_m = \mathcal{L}_m(Sup/G)$.

Proof that Sup **is nonblocking :** Since $\mathcal{L}(Sup/G) = \overline{K}$, we deduce $\overline{\mathcal{L}_m(Sup/G)} = \mathcal{L}(Sup/G)$.

A.1.2 Proof of Theorem 3.4.1

"Only if" : Consider for every $j \in J$ a D^j-supervisor Sup^j. Assume that the $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-supervisor $Sup = ((Sup^j)_{j\in J}, \vee)$ is consistent w.r.t. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$, i.e. satisfies Conds. (3.3) and (3.4) w.r.t. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$. Let us consider the set $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of subsets of $\mathcal{E}_\sigma$ defined as follows :

$$\forall j \in J, \mathcal{E}_\sigma^j = \mathcal{E}_\sigma \cap \{s \mid Sup^j(s,\sigma) = 1\}. \tag{A.1}$$

Eq (A.1) implies that $s \notin \mathcal{E}_\sigma^j$ if $Sup^j(s,\sigma) \neq 1$, $\forall j \in J$, or $s \notin \mathcal{E}_\sigma$. Let us now show that $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ is a decomposition of $\mathcal{E}_\sigma$, i.e., $\bigcup_{j\in J} \mathcal{E}_\sigma^j = \mathcal{E}_\sigma$.

Proof that $\bigcup_{j\in J} \mathcal{E}_\sigma^j \subseteq \mathcal{E}_\sigma$ **:** (A.1) implies that $\mathcal{E}_\sigma^j \subseteq \mathcal{E}_\sigma$, $\forall j \in J$. Therefore, $\bigcup_{j\in J} \mathcal{E}_\sigma^j \subseteq \mathcal{E}_\sigma$.

Proof that $\mathcal{E}_\sigma \subseteq \bigcup_{j\in J} \mathcal{E}_\sigma^j$ **:** Assume that $s \in \mathcal{E}_\sigma$. Since Sup is consistent w.r.t. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$, we have $Sup(s,\sigma) = 1$. Thus, from (3.8), $\exists j \in J$ such that $Sup^j(s,\sigma) = 1$. It follows that, from (A.1), $\exists j \in J$ s.t. $s \in \mathcal{E}_\sigma^j$. Hence, $s \in \bigcup_{j\in J} \mathcal{E}_\sigma^j$, i.e., $\mathcal{E}_\sigma \subseteq \bigcup_{j\in J} \mathcal{E}_\sigma^j$. So, $\bigcup_{j\in J} \mathcal{E}_\sigma^j = \mathcal{E}_\sigma$.

Let us show that, $\forall j \in J$, Sup^j is consistent w.r.t. $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$, i.e., Sup^j satisfies Cond. (3.3) w.r.t. $\mathcal{E}_\sigma^j$ and Cond. (3.4) w.r.t. $\mathcal{D}_\sigma$. By the construction of the decomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ defined by (A.1), we have, $\forall j \in J$, Sup^j satisfies Cond. (3.3) w.r.t. $\mathcal{E}_\sigma^j$.

Since Sup satisfies (3.4) w.r.t. $\mathcal{D}_\sigma$, we have : $\forall s \in \mathcal{D}_\sigma$,

$$\begin{aligned} Sup(s,\sigma) = 0 &\Rightarrow \forall \jmath \in J, \forall s \in \mathcal{D}_\sigma, Sup^\jmath(s,\sigma) = 0, \\ &\quad \text{(from (3.8))} \\ &\Leftrightarrow \forall \jmath \in J, Sup^\jmath \text{ satisfies (3.4) w.r.t. } \mathcal{D}_\sigma. \end{aligned}$$

Hence, $\forall \jmath \in J$, $Sup^\jmath$ satisfies Conds. (3.3) and (3.4) w.r.t. $(\mathcal{E}_\sigma^\jmath, \mathcal{D}_\sigma)$.

"If" : Consider $\mathcal{E}_\sigma$ and $\mathcal{D}_\sigma$ associated to $\sigma \in \Sigma_c$. Assume that there exists a decomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ such that, $\forall \jmath \in J$, the $\mathrm{D}^\jmath$-supervisor $Sup^\jmath$ is consistent w.r.t. $(\mathcal{E}_\sigma^\jmath, \mathcal{D}_\sigma)$, i.e.,

$$\forall \jmath \in J, s \in \mathcal{E}_\sigma^\jmath \Rightarrow Sup^\jmath(s,\sigma) = 1, \tag{A.2}$$

$$\forall \jmath \in J, s \in \mathcal{D}_\sigma \Rightarrow Sup^\jmath(s,\sigma) = 0. \tag{A.3}$$

Let us now show that the $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-supervisor $Sup = ((Sup^\jmath)_{\jmath \in J}, \vee)$ is consistent w.r.t. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$, i.e., Sup satisfies Conds. (3.3)-(3.4) w.r.t. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$. Assume that $s \in \mathcal{E}_\sigma$. Hence, since $\mathcal{E}_\sigma = \bigcup_{\jmath \in J} \mathcal{E}_\sigma^\jmath$, $\exists \jmath \in J$ such that $s \in \mathcal{E}_\sigma^\jmath$. It follows from (A.1) that $\exists \jmath \in J$ such that $Sup^\jmath(s,\sigma) = 1$. Hence, from (3.8), $Sup(s,\sigma) = 1$. That is, Sup is consistent w.r.t. $\mathcal{E}_\sigma$. Assume that $s \in \mathcal{D}_\sigma$. Hence, from (A.3), $\forall \jmath \in J$, $Sup^\jmath(s,\sigma) = 0$. It follows from (3.8) that $Sup(s,\sigma) = 0$. That is, Sup is consistent w.r.t. $\mathcal{D}_\sigma$. As required, we have shown that $Sup = ((Sup^\jmath)_{\jmath \in J}, \vee)$ is consistent w.r.t. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$.

A.1.3 Proof of Theorem 3.4.2

Consider eligible architectures $(\mathrm{D}^\jmath)_{\jmath \in J}$, and a language $K \subseteq \mathcal{L}_m$.

"Only if" : Assume that there exists a nonblocking and admissible $\vee$-$(Sup^1, \ldots, Sup^p)$-supervisor $Sup = ((Sup^\jmath)_{\jmath \in J}, \vee)$ such that $\mathcal{L}(Sup/G) = \overline{K}$ and $\mathcal{L}_m(Sup/G) = K$. This means, from Lemma 3.4.1, that K is $\mathcal{L}_m$-closed, controllable and Sup is consistent w.r.t. K. It remains to show that K is $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS.

Sup is consistent w.r.t. K means that, $\forall \sigma \in \Sigma_c$, $Sup = ((Sup^\jmath)_{\jmath \in J}, \vee)$ is consistent w.r.t. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$. Thus, from Theorem 3.4.1, $\forall \sigma \in \Sigma_c$, there exists a decomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ such that, $\forall \jmath \in J$, the $\mathrm{D}^\jmath$-supervisor $Sup^\jmath$ is consistent w.r.t. $(\mathcal{E}_\sigma^\jmath, \mathcal{D}_\sigma)$. From the latter and since, $\forall \jmath \in J$, $\mathrm{D}^\jmath$ is eligible (and then satisfies ELC-a), we have, $\forall \jmath \in J$, $(\mathcal{E}_\sigma^\jmath, \mathcal{D}_\sigma)$ is $\mathrm{D}^\jmath$-COOBS.

Therefore, $\forall \sigma \in \Sigma_c$, there exists a decomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ such that, $\forall \jmath \in J$, $(\mathcal{E}_\sigma^\jmath, \mathcal{D}_\sigma)$ is $\mathrm{D}^\jmath$-COOBS. Thus, from Definition 3.4.6, $\forall \sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS and K is $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS, as required.

"If" : Assume that K is $\mathcal{L}_m$-closed, controllable and $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS. From the latter and Definition 3.4.6, $\forall \sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS. It follows that :

(1) : $\forall \sigma \in \Sigma_c$, there exists a decomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ such that, $\forall \jmath \in J$, $(\mathcal{E}_\sigma^\jmath, \mathcal{D}_\sigma)$ is $\mathrm{D}^\jmath$-COOBS.

From the eligibility of the architectures $(\mathrm{D}^j)_{j\in J}$, ELC-b is satisfied for every architecture D^j. Thus, $\forall j \in J$, there exists a D^j-supervisor $\mathbf{Sup}^j$ such that, $\forall \sigma \in \Sigma_c$, $\forall E \subseteq \mathcal{E}_\sigma$: $(E, \mathcal{D}_\sigma)$ is D^j-COOBS implies that $\mathbf{Sup}^j$ is consistent w.r.t. $(E, \mathcal{D}_\sigma)$. It follows that (1) implies :

(2) : $\forall \sigma \in \Sigma_c$, there exists a decomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ such that, $\forall j \in J$, $\mathbf{Sup}^j$ is consistent w.r.t. $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$.

From Theorem 3.4.1, (2) is equivalent to : $\forall \sigma \in \Sigma_c$, the $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-supervisor $Sup = ((\mathbf{Sup}^j)_{j\in J}, \vee)$ is consistent w.r.t. $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$. From the latter and Definition 3.3.2, it follows that (2) is equivalent to Sup is consistent w.r.t. K. From the latter and the fact that K is $\mathcal{L}_m$-closed and controllable, it follows, from Lemma 3.4.1, that Sup is a nonblocking and admissible $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-supervisor such that $\mathcal{L}(Sup/G) = \overline{K}$ and $\mathcal{L}_m(Sup/G) = K$. As required.

A.2 Proofs of Section 3.5

A.2.1 Proof of Lemma 3.5.1

Consider a Inf_N-supervisor Sup and an event $\sigma \in \Sigma_c$. Given $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$ such that Sup is consistent w.r.t. (E, D), suppose for contradiction that (E, D) is not Inf_N-COOBS. Then, $E[N+1] \neq \emptyset$ and $D[N+1] \neq \emptyset$. By following the same steps of the proof of Lemma 4 of [Kumar et Takai, 2007] but by considering $E[k]$ and $D[k]$ instead of $\mathcal{E}_\sigma[k]$ and $\mathcal{D}_\sigma[k]$, $\forall k \geq 0$, we will find a trace $s_m \in \overline{K}$ such that $n(s_m, \sigma) = 0$ and for which two cases can occur :

$s_m \in D[1]$: then there exists $s_{m_l} \in E[0] = E$ such that $P_l(s_m) = P_l(s_{m_l})$ for some $l \in I_\sigma$ and $Sup(s_{m_l}, \sigma) \neq 1$. This is in contradiction with the consistency of Sup w.r.t. E.

$s_m \in E[1]$: then there exists $s_{m_l} \in D[0] = D$ such that $P_l(s_m) = P_l(s_{m_l})$ for some $l \in I_\sigma$ and $Sup(s_{m_l}, \sigma) \neq 0$. This is in contradiction with the consistency of Sup w.r.t. D.

Therefore, (E, D) is Inf_N-COOBS.

A.2.2 Proof of Lemma 3.5.2

Same as the proof of Lemma 1 of [Kumar et Takai, 2007] by considering subsets $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$ instead of $\mathcal{E}_\sigma$ and $\mathcal{D}_\sigma$, respectively.

A.3 Proofs of Section 3.6

A.3.1 Proof of Proposition 3.6.1

Assume that, $\exists \sigma \in \Sigma_c$, $\exists s \in \mathcal{E}_\sigma$, $\exists t \in \mathcal{D}_\sigma$ s.t., $\forall i \in I_\sigma$, $P_i(s) = P_i(t)$. For every decomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ there exists $l \in J$, s.t. $s \in \mathcal{E}_\sigma^l$. For every eligible architecture D^l,

and any D^l-supervisor $SUP^l = ((LSup_i^l)_{i\in I}, \mathrm{D}^l)$, we have from (3.6),

$$\begin{aligned} SUP^l(s,\sigma) &= \mathrm{D}^l((LSup_i^l(P_i(s),\sigma))_{i\in I_\sigma}) \\ &= \mathrm{D}^l((LSup_i^l(P_i(t),\sigma))_{i\in I_\sigma}) \\ &= SUP^l(t,\sigma). \end{aligned}$$

It follows that SUP^l is not consistent w.r.t. $(\mathcal{E}_\sigma^l, \mathcal{D}_\sigma)$. Thus, since D^l is eligible, and then satisfies ELC-b, $(\mathcal{E}_\sigma^l, \mathcal{D}_\sigma)$ is not D^l-COOBS. Therefore, from Definitions 3.4.6 and 3.4.7, K is not $\vee$-ELA$^{\geq 1}$.

A.3.2 Proof of Proposition 3.6.4

Assume that K is not observable w.r.t. $\mathcal{L}$ and Σ_o. Hence, there exist $\sigma \in \Sigma_c$, $s \in \mathcal{E}_\sigma$ and $t \in \mathcal{D}_\sigma$ such that $P(s) = P(t)$, and then $P_i(s) = P_i(t)$ for any $i \in I_\sigma$. Therefore, from Prop. 3.6.1, K is not $\vee$-ELA$^{\geq 1}$.

A.3.3 Proof of Proposition 3.6.5

Given a language $K \subseteq \mathcal{L}_m$, by Propositions 3.6.2 and 3.6.4, we have respectively "(1)$\Rightarrow$(2)" and "(2)$\Rightarrow$(3)". It remains to show that "(3)$\Rightarrow$(1)". For that, assume that K is observable, and consider the architecture $\mathrm{D} = \mathbb{I}$ ($\mathbb{I}$ stands for the identity function) as the set of $\mathbb{I}$-supervisors $Sup = (LSup(P(s).\sigma), \mathbb{I})$ (recall that $n = 1$ and $P : \Sigma \to \Sigma_o$) defined by : $\forall s \in \overline{K}$, $\forall \sigma \in \Sigma_c$, $LSup(P(s),\sigma) \in \{\phi, 0, 1\}$ and $Sup(s,\sigma) = LSup(P(s),\sigma)$. And consider the $\mathbb{I}$-COOBS property defined by : K is $\mathbb{I}$-COOBS if, $\forall \sigma \in \Sigma_c$, $\forall E \subseteq \mathcal{E}_\sigma$, $\forall D \subseteq \mathcal{D}_\sigma$: (E, D) is $\mathbb{I}$-COOBS iff $P^{-1}P(E) \cap D = \emptyset$.

Let us show that the $\mathbb{I}$-architecture is an eligible architecture (w.r.t. $\mathbb{I}$-COOBS), i.e., satisfies the condition ELC. In order to show that the $\mathbb{I}$-architecture satisfies ELC-a, consider $\sigma \in \Sigma_c$, $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$ such that (E, D) is not $\mathbb{I}$-COOBS. Hence, $P^{-1}P(E) \cap D \neq \emptyset$, which means that, $\exists s \in E$ and $\exists t \in D$ s.t. $P(s) = P(t)$. Thus, we have for every $\mathbb{I}$-supervisor Sup, $Sup(s,\sigma) = LSup(P(s),\sigma) = LSup(P(t),\sigma) = Sup(t,\sigma)$. That is, Sup is not consistent w.r.t. (E, D). Therefore, the $\mathbb{I}$-architecture satisfies ELC-a.

In order to show that the $\mathbb{I}$-architecture satisfies ELC-b, consider a $\mathbb{I}$-supervisor $SUP = (LSUP(P(s),\sigma), \mathbb{I})$ defined by : $\forall \sigma \in \Sigma_c$, $\forall E \subseteq \mathcal{E}_\sigma$, $\forall D \subseteq \mathcal{D}_\sigma$, $\forall s \in \overline{K}$,

$$SUP(s,\sigma) = LSUP(P(s),\sigma) = \begin{cases} 1 & \text{if } P^{-1}P(s) \cap D = \emptyset, \\ 0 & \text{if } P^{-1}P(s) \cap E = \emptyset, \\ \phi & \text{otherwise.} \end{cases}$$

Given $\sigma \in \Sigma_c$, $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$, suppose that SUP is not consistent w.r.t. (E, D). Hence, we have the two following possibilities,

$\exists s \in E$ s.t. $SUP(s,\sigma) \neq 1$, it follows that $P^{-1}P(s) \cap D \neq \emptyset$, and then $\exists u \in D$ s.t. $P(s) = P(u)$. Thus, $P^{-1}P(E) \cap D \neq \emptyset$, i.e., (E, D) is not $\mathbb{I}$-COOBS.

$\exists t \in D$ s.t. $SUP(t,\sigma) \neq 0$, it follows that $P^{-1}P(t) \cap E \neq \emptyset$, and then $\exists v \in E$ s.t. $P(t) = P(v)$. Thus, $P^{-1}P(E) \cap D \neq \emptyset$, i.e., (E, D) is not $\mathbb{I}$-COOBS.

Therefore, the $\mathbb{I}$-architecture satisfies ELC-b. Since K is observable, we have, $\forall \sigma \in \Sigma_c$, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\mathbb{I}$-COOBS. It follows that K is $\mathbb{I}$-COOBS.

We have shown that K is observable implies that K is $\mathbb{I}$-COOBS, where the $\mathbb{I}$-architecture is eligible. Hence, K is observable implies that K is D-COOBS for some eligible D-architecture. That is, "(3)⇒(1)".

A.3.4 Proof of Theorem 3.6.1

First of all, we need to show that for every prefix-closed languages K_1 and K_2, we have :

$$\mathcal{E}_\sigma(K_1 \cap K_2) = \mathcal{E}_\sigma(K_1) \cap \mathcal{E}_\sigma(K_2). \tag{A.4}$$

In fact, by prefix-closure of K_1 and K_2, we have $\overline{K_1 \cap K_2} = \overline{K_1} \cap \overline{K_2}$, and then : $s \in \mathcal{E}_\sigma(K_1 \cap K_2) \Leftrightarrow s\sigma \in \overline{K_1 \cap K_2} \Leftrightarrow s\sigma \in \overline{K_1} \wedge s\sigma \in \overline{K_2} \Leftrightarrow s \in \mathcal{E}_\sigma(K_1) \cap \mathcal{E}_\sigma(K_2)$.

Consider prefix-closed languages $L_1, L_2 \subseteq \mathcal{L}$ such that L_1 is $\vee$-ELA^{p_1}-COOBS and L_2 is $\vee$-ELA^{p_2}-COOBS. Hence, $\forall \sigma \in \Sigma_c$, there exists a decomposition $\{\mathcal{E}_\sigma^1(L_1), \ldots, \mathcal{E}_\sigma^{p_1}(L_1)\}$ of $\mathcal{E}_\sigma(L_1)$, such that, $\forall j \in J_1 = \{1, \ldots, p_1\}$, $(\mathcal{E}_\sigma^j(L_1), \mathcal{D}_\sigma(L_1))$ is D_1^j-COOBS for some eligible architecture D_1^j. And there exists a decomposition $\{\mathcal{E}_\sigma^1(L_2), \ldots, \mathcal{E}_\sigma^{p_2}(L_2)\}$ of $\mathcal{E}_\sigma(L_2)$, such that, $\forall j \in J_2 = \{1, \ldots, p_2\}$, $(\mathcal{E}_\sigma^j(L_2), \mathcal{D}_\sigma(L_2))$ is D_2^j-COOBS for some eligible architecture D_2^j. By using (A.4), we have :

$$\begin{aligned}
\mathcal{E}_\sigma(L) &= \mathcal{E}_\sigma(L_1 \cap L_2) \\
&= \mathcal{E}_\sigma(L_1) \cap \mathcal{E}_\sigma(L_2) \\
&= \bigcup_{j=1}^{p_1} \mathcal{E}_\sigma^j(L_1) \cap \bigcup_{j'=1}^{p_2} \mathcal{E}_\sigma^{j'}(L_2) \\
&= \bigcup_{\substack{1 \leq j \leq p_1 \\ 1 \leq j' \leq p_2}} \mathcal{E}_\sigma^j(L_1) \cap \mathcal{E}_\sigma^{j'}(L_2).
\end{aligned}$$

So, by taking $\mathcal{E}_\sigma^{j,j'}(L) = \mathcal{E}_\sigma^j(L_1) \cap \mathcal{E}_\sigma^{j'}(L_2)$, we have $\{\mathcal{E}_\sigma^{1,1}(L), \mathcal{E}_\sigma^{1,2}(L), \ldots, \mathcal{E}_\sigma^{p_1,p_2-1}(L), \mathcal{E}_\sigma^{p_1,p_2}(L)\}$ is a decomposition of $\mathcal{E}_\sigma(L)$.

It remains to show that, $\forall j \in J_1$, $\forall j' \in J_2$, $(\mathcal{E}_\sigma^{j,j'}(L), \mathcal{D}_\sigma(L))$ is D^l-COOBS for some eligible architecture D^l, where $L = L_1 \cap L_2$. For that, given a language $K \subset \mathcal{L}_m$, let us consider, for every $j \in J_1$ and $j' \in J_2$, the architecture $\mathrm{D}_1^j \wedge \mathrm{D}_2^{j'}$ as the set of $(\mathrm{D}_1^j \wedge \mathrm{D}_2^{j'})$-supervisors

$Sup^{J,J'} = Sup_1^J \wedge Sup_2^{J'}$ defined by : $\forall \sigma \in \Sigma_c$, $\forall s \in \overline{K}$,

$$Sup^{J,J'}(s,\sigma) = \begin{cases} 1 & \text{if } Sup_1^J(s,\sigma) = 1 \\ & \qquad \text{and } Sup_2^{J'}(s,\sigma) = 1, \\ 0 & \text{if } Sup_1^J(s,\sigma) = 0 \\ & \qquad \text{or } Sup_2^{J'}(s,\sigma) = 0, \\ \phi & \text{otherwise,} \end{cases} \tag{A.5}$$

where Sup_1^J (resp., $Sup_2^{J'}$) is a D_1^J-supervisor (resp., $\mathrm{D}_2^{J'}$-supervisor). The corresponding $(\mathrm{D}_1^J \wedge \mathrm{D}_2^{J'})$-coobservability is defined, for every $\sigma \in \Sigma_c$, $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$ as follows : (E, D) is $(\mathrm{D}_1^J \wedge \mathrm{D}_2^{J'})$-COOBS iff, there exists a decomposition $\{D_1, D_2\}$ of D s.t. (E, D_1) is D_1^J-COOBS and (E, D_2) is $\mathrm{D}_2^{J'}$-COOBS.

Next, we will show, for every $J \in J_1$ and $J' \in J_2$, that the architecture $\mathrm{D}_1^J \wedge \mathrm{D}_2^{J'}$ is eligible. In order to show that the architecture $\mathrm{D}_1^J \wedge \mathrm{D}_2^{J'}$ satisfies ELC-a, suppose that, $\exists \sigma \in \Sigma_c$, $\exists E \subseteq \mathcal{E}_\sigma$, $\exists D \subseteq \mathcal{D}_\sigma$, such that (E, D) is not $(\mathrm{D}_1^J \wedge \mathrm{D}_2^{J'})$-COOBS. Thus, for every decomposition $\{D_1, D_2\}$ of D, we have either (E, D_1) is not D_1^J-COOBS or (E, D_2) is not $\mathrm{D}_2^{J'}$-COOBS. Hence, from the eligibility of the architectures D_1^J and $\mathrm{D}_2^{J'}$ (and then satisfy ELC-a), for every D_1^J-supervisor Sup_1^J and for every $\mathrm{D}_2^{J'}$-supervisor $Sup_2^{J'}$, we have either Sup_1^J is not consistent w.r.t. (E, D_1) or $Sup_2^{J'}$ is not consistent w.r.t. (E, D_2) for every decomposition $\{D_1, D_2\}$ of D. We have the two following situations :

(a) Sup_1^J or $Sup_2^{J'}$ is not consistent w.r.t. E. This implies that, $\exists s \in E$ such that $Sup_1^J(s,\sigma) \neq 1$ or $Sup_2^{J'}(s,\sigma) \neq 1$. It follows, from (A.5), that $Sup^{J,J'}(s,\sigma) \neq 1$. Therefore, $Sup^{J,J'}$ is not consistent w.r.t. (E, D).

(b) Sup_1^J is not consistent w.r.t. D_1 or $Sup_2^{J'}$ is not consistent w.r.t. D_2 for every decomposition $\{D_1, D_2\}$ of D. Let us consider the decomposition $\{D_1, D_2\}$ of D such that $D_1 = \{t \in D | \, Sup_1^J(t,\sigma) = 0\}$ and $D_2 = \{t \in D | \, Sup_1^J(t,\sigma) \neq 0\}$. By construction, Sup_1^J is consistent w.r.t. D_1. Thus, $Sup_2^{J'}$ is not consistent w.r.t. D_2. Hence, we have $D_3 = \{\lambda \in D_2 | \, Sup_2^{J'}(\lambda,\sigma) \neq 0\} \neq \emptyset$. Thus, for the nonempty subset $D_3 \subseteq D$, we have, $\forall \lambda \in D_3$, $Sup_1^J(\lambda,\sigma) \neq 0$ and $Sup_2^{J'}(\lambda,\sigma) \neq 0$. That is, from (A.5), $Sup^{J,J'}(\lambda,\sigma) \neq 0$. Therefore, the supervisor $Sup^{J,J'}$ is not consistent w.r.t. (E, D).

We have shown in (a) and (b) that $Sup^{J,J'}$ is not consistent w.r.t. (E, D) for every $(\mathrm{D}_1^J \wedge \mathrm{D}_2^{J'})$-supervisor $Sup^{J,J'}$. That is, the architecture $(\mathrm{D}_1^J \wedge \mathrm{D}_2^{J'})$ satisfies ELC-a.

Let us show that the architecture $\mathrm{D}_1^J \wedge \mathrm{D}_2^{J'}$ satisfies ELC-b. For that, given $\sigma \in \Sigma_c$, $E \subseteq \mathcal{E}_\sigma$ and $D \subseteq \mathcal{D}_\sigma$, suppose that (E, D) is $(\mathrm{D}_1^J \wedge \mathrm{D}_2^{J'})$-COOBS. Thus, there exists a decomposition $\{D_1, D_2\}$ of D s.t. (E, D_1) is D_1^J-COOBS and (E, D_2) is $\mathrm{D}_2^{J'}$-COOBS. Thus, from the eligibility of the architectures D_1^J and $\mathrm{D}_2^{J'}$ (and then satisfy ELC-b), there exist a D_1^J-supervisor $\mathbf{Sup}_1^J$ and a $\mathrm{D}_2^{J'}$-supervisor $\mathbf{Sup}_2^{J'}$, such that $\mathbf{Sup}_1^J$ is consistent w.r.t. (E, D_1) and $\mathbf{Sup}_2^{J'}$ is consistent w.r.t. (E, D_2). Hence, we have,

(a) $\forall s \in E$, $\mathbf{Sup}_1^J(s,\sigma) = 1$ and $\mathbf{Sup}_2^{J'}(s,\sigma) = 1$. That is, from (A.5), $\mathbf{Sup}^{J,J'}(s,\sigma) = 1$. Thus, $\mathbf{Sup}^{J,J'} = \mathbf{Sup}_1^J \wedge \mathbf{Sup}_2^{J'}$ is consistent w.r.t. E.

(b) $\forall t \in D$, we have $t \in D_1$ or $t \in D_2$. Hence, $\mathbf{Sup}_1^{\jmath}(t,\sigma) = 0$ or $\mathbf{Sup}_2^{\jmath'}(t,\sigma) = 0$. That is, from (A.5), $\mathbf{Sup}^{\jmath,\jmath'}(t,\sigma) = 0$. Thus, $\mathbf{Sup}^{\jmath,\jmath'}$ is consistent w.r.t. D.

We have shown in (a) and (b) that $\mathbf{Sup}^{\jmath,\jmath'} = \mathbf{Sup}_1^{\jmath} \wedge \mathbf{Sup}_2^{\jmath'}$ is consistent w.r.t. (E, D) for the special $(\mathrm{D}_1^{\jmath} \wedge \mathrm{D}_2^{\jmath'})$-supervisor $\mathbf{Sup}^{\jmath,\jmath'}$. That is, the architecture $(\mathrm{D}_1^{\jmath} \wedge \mathrm{D}_2^{\jmath'})$ satisfies ELC-b.

We have show, $\forall \jmath \in J_1$, $\forall \jmath' \in J_2$, that the architecture $(\mathrm{D}_1^{\jmath} \wedge \mathrm{D}_2^{\jmath'})$ is eligible. Let us show that, $\forall \jmath \in J_1$, $\forall \jmath' \in J_2$, $(\mathcal{E}_\sigma^{\jmath,\jmath'}(L), \mathcal{D}_\sigma(L))$ is $(\mathrm{D}_1^{\jmath} \wedge \mathrm{D}_2^{\jmath'})$-Coobs. For that, we next show that $\mathcal{D}_\sigma(L) = \mathcal{D}_\sigma(L_1 \cap L_2) \subseteq \mathcal{D}_\sigma(L_1) \cup \mathcal{D}_\sigma(L_2)$. Indeed, by prefix-closure of L_1 and L_2, we have, $s \in \mathcal{D}_\sigma(L_1 \cap L_2)$ implies that $s \in \overline{L_1 \cap L_2}$ and $s\sigma \notin \overline{L_1 \cap L_2}$. That is, $s \in \overline{L_1} \cap \overline{L_2}$, and $s\sigma \notin \overline{L_1}$ or $s\sigma \notin \overline{L_2}$. Therefore, $s \in \mathcal{D}_\sigma(L_1) \cup \mathcal{D}_\sigma(L_2)$.

Hence, the subsets $D_1 = \mathcal{D}_\sigma(L) \cap \mathcal{D}_\sigma(L_1) \subseteq \mathcal{D}_\sigma(L)$ and $D_2 = \mathcal{D}_\sigma(L) \cap \mathcal{D}_\sigma(L_2) \subseteq \mathcal{D}_\sigma(L)$ form a decomposition of $\mathcal{D}_\sigma(L)$. Let $E = \mathcal{E}_\sigma^{\jmath,\jmath'}(L) = \mathcal{E}_\sigma^{\jmath}(L_1) \cap \mathcal{E}_\sigma^{\jmath'}(L_2)$, in order to show that $(E, \mathcal{D}_\sigma(L))$ is $(\mathrm{D}_1^{\jmath} \wedge \mathrm{D}_2^{\jmath'})$-Coobs, we will show that (E, D_1) is $\mathrm{D}_1^{\jmath}$-Coobs and (E, D_2) is $\mathrm{D}_2^{\jmath'}$-Coobs. For that, suppose for contradiction that either (E, D_1) is not $\mathrm{D}_1^{\jmath}$-Coobs or (E, D_2) is not $\mathrm{D}_2^{\jmath'}$-Coobs. Consider the case where (E, D_1) is not $\mathrm{D}_1^{\jmath}$-Coobs. Thus, by ELC-a, $Sup_1^{\jmath}$ is not consistent w.r.t. (E, D_1), for every $\mathrm{D}_1^{\jmath}$-supervisor $Sup_1^{\jmath}$. The two following cases can be presented :

(a) $\exists s \in E$, and then $s \in \mathcal{E}_\sigma^{\jmath}(L_1)$, such that $Sup_1^{\jmath}(s,\sigma) \neq 1$. It follows that $Sup_1^{\jmath}$ is not consistent w.r.t. $(\mathcal{E}_\sigma^{\jmath}(L_1), \mathcal{D}_\sigma(L_1))$.

(b) $\exists t \in D_1$, and then $t \in \mathcal{D}_\sigma(L_1)$, such that $Sup_1^{\jmath}(t,\sigma) \neq 0$. It follows that $Sup_1^{\jmath}$ is not consistent w.r.t. $(\mathcal{E}_\sigma^{\jmath}(L_1), \mathcal{D}_\sigma(L_1))$.

The assertions (a) or (b) imply that $Sup_1^{\jmath}$ is not consistent w.r.t. $(\mathcal{E}_\sigma^{\jmath}(L_1), \mathcal{D}_\sigma(L_1))$, for every $\mathrm{D}_1^{\jmath}$-supervisor $Sup_1^{\jmath}$, which is a contradiction with the fact that $(\mathcal{E}_\sigma^{\jmath}(L_1), \mathcal{D}_\sigma(L_1))$ is $\mathrm{D}_1^{\jmath}$-Coobs (from ELC-b). The case (E, D_2) is not $\mathrm{D}_2^{\jmath'}$-Coobs can be treated similarly, which leads to the contradiction with the fact that $(\mathcal{E}_\sigma^{\jmath'}(L_2), \mathcal{D}_\sigma(L_2))$ is $\mathrm{D}_2^{\jmath'}$-Coobs. Hence, (E, D_1) is $\mathrm{D}_1^{\jmath}$-Coobs and (E, D_2) is $\mathrm{D}_2^{\jmath'}$-Coobs, and then $(\mathcal{E}_\sigma^{\jmath,\jmath'}(L), \mathcal{D}_\sigma(L))$ is $(\mathrm{D}_1^{\jmath} \wedge \mathrm{D}_2^{\jmath'})$-Coobs, which completes the proof.

A.4 Proofs of Section 3.7

A.4.1 Proof of Theorem 3.7.1

Consider a FSA $\mathcal{A}_{\mathcal{E}_\sigma}$ accepting $\mathcal{E}_\sigma$, and let $\{x_1, \ldots, x_p\}$ be the set of marked states of $\mathcal{A}_{\mathcal{E}_\sigma}$.

"Only If" : Assume that, $\exists x \in \mathcal{A}_{\mathcal{E}_\sigma}$ s.t., for every eligible architecture D :

1. $(\mathcal{E}_\sigma^x, \mathcal{D}_\sigma)$ is not D-Coobs. Hence, from ELC-a, there is no D-supervisor Sup which is consistent w.r.t. $(\mathcal{E}_\sigma^x, \mathcal{D}_\sigma)$. Thus, from Def. 3.3.2, for every D-supervisor Sup, there exists $s \in \mathcal{E}_\sigma^x$ s.t. $Sup(s.\sigma) \neq 1$.
2. For any decomposition $\{\mathcal{E}_\sigma^1, \ldots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ satisfying **A1** (w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$), $\exists \jmath \in J$, such that $\mathcal{E}_\sigma^x \subseteq \mathcal{E}_\sigma^{\jmath}$.

From 1) and 2), for every D-supervisor Sup, there exists $s \in \mathcal{E}_\sigma^j$, such that $Sup(s.\sigma) \neq 1$, which implies that Sup is not consistent w.r.t. $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$. From the latter and ELC-b, we have that $(\mathcal{E}_\sigma^j, \mathcal{D}_\sigma)$ is not D-COOBS. Therefore, from Definition 3.4.6, there is no eligible architectures $(\mathrm{D}^j)_{j=1,\ldots,p}$ for which $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS. Hence, from Definition 3.4.7, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not $\vee$-ELA$^{\geq 1}$-COOBS w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$.

"If" : Assume that for every marked state x_j of $\mathcal{A}_{\mathcal{E}_\sigma}$, $(\mathcal{E}_\sigma^{x_j}, \mathcal{D}_\sigma)$ is D^j-COOBS for some eligible architecture D^j. From Definition 3.7.1, and the fact that the specific partition $\{\mathcal{E}_\sigma^{x_1}, \ldots, \mathcal{E}_\sigma^{x_p}\}$ of $\mathcal{E}_\sigma$ satisfies **A1**, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-$(\mathrm{D}^1, \ldots, \mathrm{D}^p)$-COOBS w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$ for some eligible architectures $(\mathrm{D}^j)_{1 \leq j \leq p}$. That is, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is $\vee$-ELA$^{\geq 1}$-COOBS w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$.

ANNEXE B

Preuves du chapitre 4

B.1 Proofs of Section 4.5

B.1.1 Proof of Lemma 4.5.1

Lemma 4.5.1 can be deduced from the following known result :

Result 1 : Consider n nondeterministic FSA $A_1, \cdots, A_n$ over an alphabet of cardinality e. Let q_i be the number of states of A_i, for $i = 1 \cdots n$, and $q = q_1 \times \cdots \times q_n$. Let A be the synchronized product of all the $A_i, i = 1 \cdots n$. The number of states of A is in $O(q)$ and its number of transitions is in $O(q.(q.e)) = O(q^2.e)$.

Let us apply Result 1 to $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ which is the synchronized product of $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ with $P_i^{-1}P_i(\mathcal{A}_{\mathcal{D}_\sigma[k]})$, $i = 1 \cdots n_\sigma$. Each $P_i^{-1}P_i(\mathcal{A}_{\mathcal{D}_\sigma[k]})$ is computed by a projection P_i of $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ without determinization and then by adding self-loops. Therefore the number of states of each $P_i^{-1}P_i(\mathcal{A}_{\mathcal{D}_\sigma[k]})$ is in $O(|Y[k]|)$. Using Result 1, we obtain :

the number of states $|X[k+1]|$ of $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ is in $O(|X[k]| \cdot |Y[k]|^{n_\sigma})$,
the number of transitions $|\alpha[k+1]|$ of $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ is in $O(|X[k+1]^2| \cdot |\Sigma|) = O(|X[k]|^2 \cdot |Y[k]|^{2n_\sigma} \cdot |\Sigma|)$.

With the same approach, if we apply Result 1 to $A_{\mathcal{D}_\sigma[k+1]}$ which is the synchronized product of $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ with $P_i^{-1}P_i(\mathcal{A}_{\mathcal{E}_\sigma[k]})$, $i = 1 \cdots n_\sigma$, we obtain :

the number of states $|Y[k+1]|$ of $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ is in $O(|Y[k]| \cdot |X[k]|^{n_\sigma})$,
the number of transitions $|\beta[k+1]|$ of $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ is in $O(|Y[k+1]^2| \cdot |\Sigma|) = O(|Y[k]|^2 \cdot |X[k]|^{2n_\sigma} \cdot |\Sigma|)$.

B.1.2 Proof of Lemma 4.5.2

Lemma 4.5.2 can be deduced from the following known result :

Result 2 : The computational complexity for constructing an automaton is in the order of its number of transitions.

Let us apply Result 2 to $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ whose number of transitions is $|\alpha[k+1]| = O(|X[k]|^2 \cdot |Y[k]|^{2n_\sigma} \cdot |\Sigma|)$ (from Lemma 4.5.1). We obtain that the computational complexity for constructing $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ is in $O(|\alpha[k+1]|) = O(|X[k]|^2 \cdot |Y[k]|^{2n_\sigma} \cdot |\Sigma|)$.

Let us apply Result 2 to $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ whose number of transitions is $|\beta[k+1]| = O(|Y[k]|^2 \cdot |X[k]|^{2n_\sigma} \cdot |\Sigma|)$ (from Lemma 4.5.1). We obtain that the computational complexity for constructing $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ is in $O(|\beta[k+1]|) = O(|Y[k]|^2 \cdot |X[k]|^{2n_\sigma} \cdot |\Sigma|)$.

B.1.3 Proof of Proposition 4.5.1

Computing $\mathcal{A}_{\mathcal{E}_\sigma[1]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma}$ and $\mathcal{A}_{\mathcal{D}_\sigma}$ is the most costly part (in terms of computation) in the procedure that checks if $\mathcal{A}_{\mathcal{E}_\sigma[1]} = \emptyset$ or $\mathcal{A}_{\mathcal{D}_\sigma[1]} = \emptyset$.

From Lemma 4.5.2 :

The complexity of computing $\mathcal{A}_{\mathcal{E}_\sigma[1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma}$ and $\mathcal{A}_{\mathcal{D}_\sigma}$ is in $O(|\alpha[1]|) = O(|X|^2 \cdot |Y|^{2n_\sigma} \cdot |\Sigma|)$.
The complexity of computing $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma}$ and $\mathcal{A}_{\mathcal{D}_\sigma}$ is in $O(|\beta[1]|) = O(|Y|^2 \cdot |X|^{2n_\sigma} \cdot |\Sigma|)$.

Therefore, the complexity of computing $\mathcal{A}_{\mathcal{E}_\sigma[1]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma}$ and $\mathcal{A}_{\mathcal{D}_\sigma}$ is the sum of the two above complexities : $O((|X|^2 \cdot |Y|^{2n_\sigma} + |Y|^2 \cdot |X|^{2n_\sigma}) \cdot |\Sigma|)$.

B.1.4 Proof of Proposition 4.5.2

Consider a decomposition $\{X_m^1, \cdots, X_m^p\}$ of X_m. By considering the decomposition $\{\mathcal{E}_\sigma^1, \cdots, \mathcal{E}_\sigma^p\}$ of $\mathcal{E}_\sigma$ such that each $\mathcal{E}_\sigma^j$ is given by (4.11), we show by induction on the inference steps k, that : $\mathcal{E}_\sigma^j[k] = \{s \in \Sigma^* | \alpha[k](x_0[k], s) \in X_m[k]|_{X_m^j}\}$ and $\mathcal{D}_\sigma^j[k] = \{s \in \Sigma^* | \beta[k](y_0[k], s) \in Y_m[k]|_{X_m^j}\}$.

Basis : Let us show that $\mathcal{E}_\sigma^j[1] = \{s \in \Sigma^* | \alpha[1](x_0[1], s) \in X_m[1]|_{X_m^j}\}$,

$$\begin{aligned} s \in \mathcal{E}_\sigma^j[1] &\Leftrightarrow [s \in \mathcal{E}_\sigma^j] \wedge [\forall i \in I_\sigma, s \in P_i^{-1}P_i(\mathcal{D}_\sigma)] \\ &\Leftrightarrow [s \in \mathcal{E}_\sigma^j] \wedge [\forall i \in I_\sigma, \exists t_i \in \mathcal{D}_\sigma \text{ s.t. } P_i(s) = P_i(t_i)] \end{aligned}$$

From (4.11), $s \in \mathcal{E}_\sigma^j$ is equivalent to $\alpha(x_0, s) \in X_m^j$, and hence,

$$s \in \mathcal{E}_\sigma^j[1] \Leftrightarrow [\alpha(x_0, s) = x \in X_m^j] \wedge [\forall i \in I_\sigma, \exists t_i \in \mathcal{D}_\sigma \text{ s.t. } P_i(s) = P_i(t_i)].$$

Since $\mathcal{A}_{\mathcal{D}_\sigma}$ is an acceptor of $\mathcal{D}_\sigma$, we have, $\forall i \in I_\sigma$, $\beta(y_0, t_i) = v_i$ for some $v_i \in Y_m$. Thus, from the synchronous composition of the FSAs $\mathcal{A}_{\mathcal{E}_\sigma}$ and $(P_i^{-1}P_i(\mathcal{A}_{\mathcal{D}_\sigma}))_{i \in I_\sigma}$ [Hopcroft et Ullman, 1979], we have,

$$s \in \mathcal{E}_\sigma^j[1] \Leftrightarrow \alpha[1](x_0[1], s) = u = (u_1, \cdots, u_n, x) \in X_m[1] \text{ s.t., } \forall i \in I_\sigma, v_i \in u_i.$$

Since x is X_m^j-marked, we have u is X_m^j-marked, and hence

$$s \in \mathcal{E}_\sigma^j[1] \Leftrightarrow \alpha[1](x_0[1], s) \in X_m[1]|_{X_m^j}.$$

Let us show that $\mathcal{D}_\sigma^j[1] = \{s \in \Sigma^* | \beta[1](y_0[1], s) \in Y_m[1]|_{X_m^j}\}$,

$$\begin{aligned} s \in \mathcal{D}_\sigma^j[1] &\Leftrightarrow [s \in \mathcal{D}_\sigma] \wedge [\forall i \in I_\sigma, s \in P_i^{-1}P_i(\mathcal{E}_\sigma^j)] \\ &\Leftrightarrow [s \in \mathcal{D}_\sigma] \wedge [\forall i \in I_\sigma, \exists t_i \in \mathcal{E}_\sigma^j \text{ s.t. } P_i(s) = P_i(t_i)]. \end{aligned}$$

From the latter equivalence and the fact that $\mathcal{A}_{\mathcal{D}_\sigma}$ is an acceptor of $\mathcal{D}_\sigma$, we deduce,

$$s \in \mathcal{D}_\sigma^J[1] \Leftrightarrow [\exists y \in Y_m \text{ s t. } \beta(y_0, s) = y] \wedge [\forall \imath \in I_\sigma, \exists t_\imath \in \mathcal{E}_\sigma^J \text{ s.t. } P_\imath(s) = P_\imath(t_\imath)].$$

From (4.11), we have, $\forall \imath \in I_\sigma$, $\alpha(x_0, t_\imath) = u_\imath$ for some $u_\imath \in X_m^J$. Thus, from the synchronous composition of the FSAs $\mathcal{A}_{\mathcal{D}_\sigma}$ and $(P_\imath^{-1}P_\imath(\mathcal{A}_{\mathcal{E}_\sigma}))_{\imath \in I_\sigma}$ [Hopcroft et Ullman, 1979], we have,

$$s \in \mathcal{D}_\sigma^J[1] \Leftrightarrow \beta[1](y_0[1], s) = v = (v_1, \cdots, v_n, y) \in Y_m[1] \text{ s.t., } \forall \imath \in I_\sigma, u_\imath \in v_\imath.$$

Since, $\forall \imath \in I_\sigma$, $u_\imath$ is X_m^J-marked, we have v is X_m^J-marked, and hence,

$$s \in \mathcal{D}_\sigma^J[1] \Leftrightarrow \beta[1](y_0[1], s) \in Y_m[1]|_{X_m^J}.$$

Induction step : Assume that $\mathcal{E}_\sigma^J[k] = \{s \in \Sigma^* | \, \alpha[k](x_0[k], s) \in X_m[k]|_{X_m^J}\}$ and $\mathcal{D}_\sigma^J[k] = \{s \in \Sigma^* | \, \beta[k](y_0[k], s) \in Y_m[k]|_{X_m^J}\}$. And let us show that $\mathcal{E}_\sigma^J[k+1] = \{s \in \Sigma^* | \, \alpha[k+1](x_0[k+1], s) \in X_m[k+1]|_{X_m^J}\}$ and $\mathcal{D}_\sigma^J[k+1] = \{s \in \Sigma^* | \, \beta[k+1](y_0[k+1], s) \in Y_m[k+1]|_{X_m^J}\}$.

$$\begin{aligned} s \in \mathcal{E}_\sigma^J[k+1] &\Leftrightarrow [s \in \mathcal{E}_\sigma^J[k]] \wedge [\forall \imath \in I_\sigma, s \in P_\imath^{-1}P_\imath(\mathcal{D}_\sigma^J[k])] \\ &\Leftrightarrow [s \in \mathcal{E}_\sigma^J[k]] \wedge [\forall \imath \in I_\sigma, \exists t_\imath \in \mathcal{D}_\sigma^J[k] \text{ s.t. } P_\imath(s) = P_\imath(t_\imath)]. \end{aligned}$$

From the induction hypothesis, we have :

$$\begin{aligned} s \in \mathcal{E}_\sigma^J[k] &\Leftrightarrow \alpha[k](x_0[k], s) = x \in X_m[k] \text{ for some } X_m^J\text{-marked state } x \text{ in } \mathcal{A}_{\mathcal{E}_\sigma[k]}, \\ t_\imath \in \mathcal{D}_\sigma^J[k] &\Leftrightarrow \beta[k](y_0[k], t_\imath) = v_\imath \in Y_m[k] \text{ for some } X_m^J\text{-marked state } v_\imath \text{ in } \mathcal{A}_{\mathcal{D}_\sigma[k]} \end{aligned}$$

Thus, from the synchronous composition of the FSAs $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $(P_\imath^{-1}P_\imath(\mathcal{A}_{\mathcal{D}_\sigma[k]}))_{\imath \in I_\sigma}$ [Hopcroft et Ullman, 1979], s reaches a state $(u_1, \cdots, u_n, x)$ in $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ s.t. $\forall \imath \in I_\sigma$, $v_\imath \in u_\imath$, hence,

$$s \in \mathcal{E}_\sigma^J[k+1] \Leftrightarrow \alpha[k+1](x_0[k+1], s) = (u_1, \cdots, u_n, x) \in X_m[k+1] \text{ s.t., } \forall \imath \in I_\sigma, v_\imath \in u_\imath.$$

Therefore, from the definition of the multi-marking, we have,

$$s \in \mathcal{E}_\sigma^J[k+1] \Leftrightarrow \alpha[k+1](x_0[k+1], s) \in X_m[k+1]|_{X_m^J}.$$

Therefore, $\mathcal{E}_\sigma^J[k+1] = \{s \in \Sigma^* | \, \alpha[k+1](x_0[k+1], s) \in X_m[k+1]|_{X_m^J}\}$. The same procedure can be followed to show that $\mathcal{D}_\sigma^J[k+1] = \{s \in \Sigma^* | \, \beta[k+1](y_0[k+1], s) \in Y_m[k+1]|_{X_m^J}\}$.

B.1.5 Proof of Lemma 4.5.3

Consider a decomposition $D = \{X_m^1, \cdots, X_m^p\}$ of X_m s.t. D satisfies (4 13). Let us prove Lemma 4.5.3 by induction on the inference steps $k \geq 1$.

Basis $(k = 1)$ **:** Let $v = (v_1, \cdots, v_n, v_{n_\sigma+1}) \in Y_m[1]$ be a X_m^J-marked state for some $X_m^J \in D$. Then, since X_m^J satisfies Cond. (4.13), we have, $\forall (a_1, \cdots, a_{n_\sigma}) \in (v_1 \cap X_m^J) \times$

$\cdots \times (v_{n_\sigma} \cap X_m^J)$, $|\bigcup_{\imath \in \mathbb{I}_\sigma} \{a_\imath\}| = 1$, i.e., $\forall \imath \in \mathbb{I}_\sigma$, $v_\imath \cap X_m^J = \{x\}$, for some unique $x \in X_m^J$. This means that v is $\{x\}$-marked and not $\mathcal{X}$-marked, $\forall \mathcal{X} \subseteq X_m^J$ s.t. $\mathcal{X} \neq \{x\}$. From the definition of the multi-marking (Subsection 4.5.3), for every $v \in X_m[1]$ such that v is X_m^J-marked for some $X_m^J \in D$, there exists a unique $x \in X_m^J$ such that v is $\{x\}$-marked.

Induction Step : Given $k \geq 1$, assume that for every marked state $u \in X_m[k]|_{X_m^J} \cup Y_m[k]|_{X_m^J}$ there exists $x \in X_m^J$ such that u is $\{x\}$-marked and not $\mathcal{X}$-marked, $\forall \mathcal{X} \subseteq X_m^J$ s.t. $\mathcal{X} \neq \{x\}$. Let us show that for every marked state $v \in X_m[k+1]|_{X_m^J}$ (the case $v \in Y_m[k+1]|_{X_m^J}$ can be treated similarly) there exists $z \in X_m^J$ such that v is $\{z\}$-marked and not $\mathcal{X}$-marked, $\forall \mathcal{X} \subseteq X_m^J$ s.t. $\mathcal{X} \neq \{z\}$. For that, let $v^{k+1} = (v_1^{k+1}, \cdots, v_{n_\sigma}^{k+1}, v_{n_\sigma+1}^{k+1})$ be a X_m^J-marked state in $X_m[k+1]$ (the superscript "k+1" stands for the inference step), which implies that $w^k = v_{n_\sigma+1}^{k+1}$ is X_m^J-marked in $X_m[k]$. Hence, from the induction hypothesis, there exists $x \in X_m^J$ such that $w^k \in X_m[k]$ is $\{x\}$-marked and not $\mathcal{X}$-marked in $X_m[k]$, $\forall \mathcal{X} \subseteq X_m^J$ s.t. $\mathcal{X} \neq \{x\}$.

Assume for contradiction that v^{k+1} is $\{x, y\}$-marked, where $y \in X_m^J$, such that, $\exists l \in \mathbb{I}_\sigma$, satisfying $v_l^{k+1} \cap Y_m[k]|_{\{y\}} \neq \emptyset$, i.e., v_l^{k+1} is $\{y\}$-marked. Hence, $\forall \imath \in \mathbb{I}_\sigma \backslash \{l\}$, there exists $u_\imath^k \in v_\imath^{k+1} \cap Y_m[k]$ which is $\{x, y\}$-marked, and there exists $u_l^k \in v_l^{k+1} \cap Y_m[k]$ which is $\{y\}$-marked. Moreover, and from the definition of the natural projection of FSA [Hopcroft et Ullman, 1979], there exit $s \in \mathcal{E}_\sigma[k]$ reaching the state $w^k \in X_m[k]$ and traces $t_\imath \in \mathcal{D}_\sigma[k]$ reaching the states $u_\imath^k \in Y_m[k]$ such that $P_\imath(s) = P_\imath(t_\imath)$, $\imath \in \mathbb{I}_\sigma$. Since, w^k is $\{x\}$-marked, $(u_\imath^k)_{\imath \in \mathbb{I}_\sigma \backslash \{l\}}$ are $\{x, y\}$-marked and u_l^k is $\{y\}$-marked, s reaches a $\{x\}$-marked state $w^{k-1} \in X_m[k-1]$, $t_\imath$ reaches a $\{x, y\}$-marked state $u_\imath^{k-1} \in Y_m[k-1]$, for $\imath \in \mathbb{I}_\sigma \backslash \{l\}$, and t_l reaches a $\{y\}$-marked state $u_l^{k-1} \in Y_m[k-1]$. Therefore, from the fact that, $\forall \imath \in \mathbb{I}_\sigma$, $P_\imath(s) = P_\imath(t_\imath)$, s reaches a state $\lambda = (\lambda_1, \cdots, \lambda_{n_\sigma}, w^{k-1}) \in X_m[k]$ such that, $\forall \imath \in \mathbb{I}_\sigma$, $u_\imath^{k-1} \in \lambda_\imath$. It follows that λ is $\{x, y\}$-marked and $\lambda_l \cap Y_m[k]|_{\{y\}} \neq \emptyset$, i.e., λ_l is $\{y\}$-marked, which is a contradiction with the induction hypothesis. Thus, v^{k+1} is $\{x\}$-marked and not $\mathcal{X}$-marked in $X_m[k+1]$, $\forall \mathcal{X} \subseteq X_m^J$ s.t. $\mathcal{X} \neq \{x\}$.

B.1.6 Proof of Theorem 4.5.1

Consider a FSA $\mathcal{A}_{\mathcal{E}_\sigma}$ and a decomposition $\{X_m^1, \cdots, X_m^p\}$ of X_m satisfying (4.13). Assume that $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not $\vee$-$Inf_{\leq N+1}^{\geq 1}$-COOBS w.r.t. $\{\mathcal{E}_\sigma^1, \cdots, \mathcal{E}_\sigma^p\}$, where $\mathcal{E}_\sigma^J = \{s \in \Sigma^* | \, \alpha(x_0, s) \in X_m^J\}$. Then, $\exists j \in J$ such that $(\mathcal{E}_\sigma^J, \mathcal{D}_\sigma)$ is not Inf_{N+1}-COOBS. Hence, $\mathcal{E}_\sigma^J[N+2] \neq \emptyset$ and $\mathcal{D}_\sigma^J[N+2] \neq \emptyset$. Thus, from Proposition 4.5.2, there exist states $u \in X_m[N+2]$ and $v \in Y_m[N+2]$ that are X_m^J-marked. Since $\mathcal{E}_\sigma^J$ satisfies (4.13), this implies, from Lemma 4.5.3, that there exist $x, y \in X_m^J$ for which u is $\{x\}$-marked and v is $\{y\}$-marked.

For every decomposition $D = (\mathcal{X}^1, \cdots, \mathcal{X}^q)$ of X_m, two cases can be presented :

1. $\exists \mathcal{X}^l, \mathcal{X}^{l'} \in D$ such that $x \in \mathcal{X}^l$ and $y \in \mathcal{X}^{l'}$, and thus u is $\mathcal{X}^l$-marked and v is $\mathcal{X}^{l'}$-marked. Let $\mathcal{E}_\sigma^{\mathcal{X}}[k]$ and $\mathcal{D}_\sigma^{\mathcal{X}}[k]$ denote the set of traces reaching the $\mathcal{X}$-marked states of $X_m[k]$ and $Y_m[k]$, respectively. Therefore, $\mathcal{E}_\sigma^{\mathcal{X}^l}[N+2] \neq \emptyset$ and $\mathcal{D}_\sigma^{\mathcal{X}^{l'}}[N+2] \neq \emptyset$. It follows that $\mathcal{E}_\sigma^{\mathcal{X}^l}[N+1] \neq \emptyset$ and $\mathcal{D}_\sigma^{\mathcal{X}^l}[N+1] \neq \emptyset$, and $\mathcal{E}_\sigma^{\mathcal{X}^{l'}}[N+1] \neq \emptyset$ and $\mathcal{D}_\sigma^{\mathcal{X}^{l'}}[N+1] \neq \emptyset$. That is, $(\mathcal{E}_\sigma^{\mathcal{X}^l}, \mathcal{D}_\sigma)$ is not Inf_N-COOBS and $(\mathcal{E}_\sigma^{\mathcal{X}^{l'}}, \mathcal{D}_\sigma)$ is not Inf_N-COOBS.

2. $\exists \mathcal{X}^l \in D$ such that $x, y \in \mathcal{X}^l$, and thus u is $\mathcal{X}^l$-marked and v is $\mathcal{X}^l$-marked. Therefore, $\mathcal{E}_\sigma^{\mathcal{X}^l}[N+2] \neq \emptyset$ and $\mathcal{D}_\sigma^{\mathcal{X}^l}[N+2] \neq \emptyset$. It follows that $\mathcal{E}_\sigma^{\mathcal{X}^l}[N+1] \neq \emptyset$ and $\mathcal{D}_\sigma^{\mathcal{X}^l}[N+1] \neq \emptyset$. That is, $(\mathcal{E}_\sigma^{\mathcal{X}^l}, \mathcal{D}_\sigma)$ is not Inf_N-COOBS.

Therefore, from (1) and (2), we have $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not $\vee$-$Inf_{\leq N}^{\geq 1}$-COOBS w.r.t. $\mathcal{A}_{\mathcal{E}_\sigma}$, as required.

B.1.7 Proof of Proposition 4.5.3

The proof of this proposition is adapted from the proof of Prop. 4.5.1.

For each $j \in J$, computing $\mathcal{A}_{\mathcal{E}_\sigma^j[k+1]}$ and $\mathcal{A}_{\mathcal{D}_\sigma^j[k+1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma^j[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma^j[k+1]}$ is the most costly part (in terms of computation) in the procedure that checks if $\mathcal{E}_\sigma^j[k+1] = \emptyset$ or $\mathcal{D}_\sigma^j[k+1] = \emptyset$.

From Lemma 4.5.2 :

1. The complexity of computing $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma}[k]$ and $\mathcal{A}_{\mathcal{D}_\sigma}[k]$ is in $O(|\alpha[k+1]|) = O(|X[k]|^2 \cdot |Y[k]|^{2n_\sigma} \cdot |\Sigma|)$. The complexity of computing $\mathcal{A}_{\mathcal{E}_\sigma^j[k+1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma^j[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma^j[k]}$ is in the same order because $\mathcal{A}_{\mathcal{E}_\sigma^j[k+1]}$ is obtained from $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ by : 1) keeping marked only the states of $\mathcal{A}_{\mathcal{E}_\sigma[k+1]}$ that are X_m^j-marked, and 2) removing all states which are not reachable from the X_m^j-marked states. This procedure is in the worst case in $O(|\alpha[k+1]|)$.
2. The complexity of computing $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma}[k]$ and $\mathcal{A}_{\mathcal{D}_\sigma}[k]$ is in $O(|\beta[k+1]|) = O(|Y[k]|^2 \cdot |X[k]|^{2n_\sigma} \cdot |\Sigma|)$. The complexity of computing $\mathcal{A}_{\mathcal{D}_\sigma^j[k+1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma^j[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma^j[k]}$ is in the same order because $\mathcal{A}_{\mathcal{D}_\sigma^j[k+1]}$ is obtained from $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ by : 1) keeping marked only the states of $\mathcal{A}_{\mathcal{D}_\sigma[k+1]}$ that are X_m^j-marked, and 2) removing all states which are not reachable from the X_m^j-marked states. This procedure is in the worst case in $O(|\beta[k+1]|)$.

Therefore, the complexity of computing $\mathcal{A}_{\mathcal{E}_\sigma^j[k+1]}$ and $\mathcal{A}_{\mathcal{D}_\sigma^j[k+1]}$ from $\mathcal{A}_{\mathcal{E}_\sigma^j[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma^j[k]}$ is the sum of the two above complexities : $O((|X[k]|^2 \cdot |Y[k]|^{2n_\sigma} + |Y[k]|^2 \cdot |X[k]|^{2n_\sigma}) \cdot |\Sigma|)$.

B.1.8 Proof of Lemma 4.5.4

Given two subsets $\mathcal{W}, \mathcal{Z} \subseteq X_m$, we have :

$$\begin{aligned}\mathrm{ID}_v(\mathcal{W} \cup \mathcal{Z}) &= \{i \in \mathbb{I}_\sigma \mid v_i \cap (\mathcal{W} \cup \mathcal{Z}) = \emptyset\} = \{i \in \mathbb{I}_\sigma \mid (v_i \cap \mathcal{W} = \emptyset) \wedge (v_i \cap \mathcal{Z} = \emptyset)\} \\ &= \{i \in \mathbb{I}_\sigma \mid v_i \cap \mathcal{W} = \emptyset\} \cap \{i \in \mathbb{I}_\sigma \mid v_i \cap \mathcal{Z} = \emptyset\} = \mathrm{ID}_v(\mathcal{W}) \cap \mathrm{ID}_v(\mathcal{Z}).\end{aligned}$$

B.1.9 Proof of Lemma 4.5.5

$\mathcal{X} \subseteq X_m$ satisfies (4.13) means (by definition) :

1. $\forall v \in Y_m[1]$ s.t. $v_i \cap \mathcal{X} \neq \emptyset$, $\forall i \in \mathbb{I}_\sigma$,
2. $\forall (a_1, \cdots, a_{n_\sigma}) \in (v_1 \cap \mathcal{X}) \times \cdots \times (v_{n_\sigma} \cap \mathcal{X})$, we have $|\bigcup_i \{a_i\}| = 1$.

The above Item 1 can be rewritten : $\forall v \in Y_m[1]$ s.t $\mathrm{ID}_v(\mathcal{X}) = \emptyset$.

The above item 2 is equivalent to : $|\bigcup_{i\in\mathbb{I}_\sigma}(\mathcal{X} \cap v_i)| = 1$, which can be rewritten : $|\mathcal{X} \cap (\bigcup_{i\in\mathbb{I}_\sigma} v_i)| = 1$.

To recapitulate : $\mathcal{X}$ satisfies (4.13) is equivalent to : $\forall v \in Y_m[1]$ s.t. $\mathrm{ID}_v(\mathcal{X}) = \emptyset$, $|\mathcal{X} \cap \bigcup_{i\in\mathbb{I}_\sigma} v_i| = 1$.

B.1.10 Proof of Proposition 4.5.4

Consider $x, y \in X_m$. Let us prove that $y \in \mathrm{Elig}(x)$ iff $y \in X_m \setminus [\{x\} \cup \bigcup_{\substack{v\in Y_m[1]\\ \mathrm{ID}_v(x)=\emptyset}} \bigcup_{i\in\mathbb{I}_\sigma} v_i \cup \bigcup_{\substack{v\in Y_m[1]\\ \mathrm{ID}_v(x)\neq\mathbb{I}_\sigma}} \bigcap_{i\in\mathrm{ID}_v(x)} v_i]$. By definition, we have,

$$y \in \mathrm{Elig}(x) \Leftrightarrow [\{x, y\} \text{ satisfies (4.13)}] \wedge [y \in X_m \setminus \{x\}]$$

The set of states $A \subseteq X_m$ s.t., $\forall y \in A$, $\{x, y\}$ satisfies (4.13), can be obtained by computing the set of states $B \subseteq X_m$ s.t. $\forall z \in A$, $\{x, z\}$ does not satisfy (4.13). And then we have $A = B^c$, where B^c is the complementary of B. For every $z \in B$, we have,

$$\{x, z\} \text{ does not satisfy (4.13)} \tag{B.1}$$

$$(B.1) \Leftrightarrow \exists v \in Y_m[1] \text{ s.t. } \mathrm{ID}_v(\{x, z\}) = \emptyset \text{ and } |\{x, z\} \cap \bigcup_{i\in\mathbb{I}_\sigma} v_i| > 1$$

$$(B.1) \Leftrightarrow \exists v \in Y_m[1] \text{ s.t. } \mathrm{ID}_v(\{x, z\}) = \emptyset \text{ and } \{x, z\} \cap \bigcup_{i\in\mathbb{I}_\sigma} v_i = \{x, z\}$$

$\{x, z\} \cap \bigcup_{i\in\mathbb{I}_\sigma} v_i = \{x, z\}$ is equivalent to $\{x, z\} \subseteq \bigcup_{i\in\mathbb{I}_\sigma} v_i$ which is equivalent to say that $\mathrm{ID}_v(x) \neq \mathbb{I}_\sigma$ and $\mathrm{ID}_v(z) \neq \mathbb{I}_\sigma$. So, we have

$$(B.1) \Leftrightarrow \exists v \in Y_m[1] \text{ s.t. } \mathrm{ID}_v(\{x, z\}) = \emptyset, \mathrm{ID}_v(x) \neq \mathbb{I}_\sigma \text{ and } \mathrm{ID}_v(z) \neq \mathbb{I}_\sigma.$$

For every $v \in Y_m[1]$ s.t. $\mathrm{ID}_v(\{x, z\}) = \emptyset$, $\mathrm{ID}_v(x) \neq \mathbb{I}_\sigma$ and $\mathrm{ID}_v(z) \neq \mathbb{I}_\sigma$, two cases can be presented :

1. $\mathrm{ID}_v(x) = \emptyset$: in this case for every $z \in \bigcup_{i\in\mathbb{I}_\sigma} v_i$, $\{x, z\}$ does not satisfy (4.13).
2. $\mathrm{ID}_v(x) \neq \emptyset$: since $\mathrm{ID}_v(\{x, z\}) = \emptyset$, $\forall i \in \mathbb{I}_\sigma$, $\{x, z\} \cap v_i \neq \emptyset$. Hence, $\forall l \in \mathbb{I}_\sigma$ s.t. $x \notin v_l$, i.e., $\forall l \in \mathrm{ID}_v(x)$, we have $z \in v_l$. It follows that $z \in \bigcap_{l\in\mathrm{ID}_v(x)} v_l$. Therefore, for every $z \in \bigcap_{l\in\mathrm{ID}_v(x)} v_l$, $\{x, z\}$ does not satisfy (4.13).

Since, either $z \in \bigcup_{i\in\mathbb{I}_\sigma} v_i$ (in the case $\mathrm{ID}_v(x) = \emptyset$) or $z \in \bigcap_{l\in\mathrm{ID}_v(x)} v_l$ (in the case $\mathrm{ID}_v(x) \neq \emptyset$) imply that $\mathrm{ID}_v(z) \neq \mathbb{I}_\sigma$, we have,

$(B.1) \Leftrightarrow \exists v \in Y_m[1]$ s.t.,

$$[\mathrm{ID}_v(x) \neq \mathbb{I}_\sigma \wedge \mathrm{ID}_v(x) = \emptyset \wedge z \in \bigcup_{i\in\mathbb{I}_\sigma} v_i] \vee [\mathrm{ID}_v(x) \neq \mathbb{I}_\sigma \wedge \mathrm{ID}_v(x) \neq \emptyset \wedge z \in \bigcap_{l\in\mathrm{ID}_v(x)} v_l]$$

Since $\mathrm{ID}_v(x) = \emptyset$ implies $\mathrm{ID}_v(x) \neq \mathbb{I}_\sigma$, and $\mathrm{ID}_v(x) = \emptyset$ implies $\bigcap_{l \in \mathrm{ID}_v(x)} v_l = \emptyset$, we have,

$$(B.1) \Leftrightarrow \exists v \in Y_m[1] \text{ s.t., } [\mathrm{ID}_v(x) = \emptyset \wedge z \in \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota] \vee [\mathrm{ID}_v(x) \neq \mathbb{I}_\sigma \wedge z \in \bigcap_{l \in \mathrm{ID}_v(x)} v_l]$$

$$\Leftrightarrow z \in \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) = \emptyset}} \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_\iota(x)} v_\iota$$

Therefore, $B = \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) = \emptyset}} \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_\iota(x) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_\iota(x)} v_\iota$. Thus, we have,

$$y \in \mathrm{Elig}(x) \Leftrightarrow [y \notin B] \wedge [y \in X_m \setminus \{x\}]$$
$$\Leftrightarrow y \in X_m \setminus [\{x\} \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) = \emptyset}} \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_v(x)} v_\iota]$$

B.1.11 Proof of Proposition 4.5.5

Consider $\mathcal{X} \subseteq X_m$, $x \in \mathrm{Elig}(\mathcal{X})$. We have to prove the equality $S = T$, where $S = \mathrm{Elig}(\mathcal{X} \cup \{x\})$ and $T = (\mathrm{Elig}(\mathcal{X}) \cap \mathrm{Elig}(x)) \setminus [\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_\iota]$. Let us prove "$S \subseteq T$" and "$T \subseteq S$".

Proof of $S \subseteq T$: Consider $y \in \mathrm{Elig}(\mathcal{X} \cup \{x\})$, which means that $y \in X_m \setminus (\mathcal{X} \cup \{x\})$ and $\mathcal{X} \cup \{y, x\}$ satisfies (4.13). Hence, $y \in X_m \setminus \mathcal{X}$, $y \in X_m \setminus \{x\}$, $\mathcal{X} \cup \{y\}$ satisfies (4.13) and $\{y, x\}$ satisfies (4.13). Therefore, $y \in \mathrm{Elig}(\mathcal{X}) \cap \mathrm{Elig}(x)$. We now prove ad adsurbum $y \notin \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_\iota$.

Assume that $y \in \bigcup_{\substack{v \in Y_m[1]: \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_\iota$. Hence, $\exists v = (v_1, \cdots, v_{n+1}) \in Y_m[1]$ s.t. $\mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq \mathbb{I}_\sigma$ and, $\forall i \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})$, $y \in v_\iota$. From the latter, we deduce that $\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) = \emptyset$. From $\mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq \mathbb{I}_\sigma$, we deduce $(\bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota) \cap (\mathcal{X} \cup \{x\}) \neq \emptyset$. The latter expression and the facts that $y \notin \mathcal{X} \cup \{x\}$ and $y \in \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota$ imply that $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota| > 1$. $\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) = \emptyset$ and $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota| > 1$ imply that $\mathcal{X} \cup \{x, y\}$ does not satisfy (4.13) (Lemma 4.5.5), which contradicts the hypothesis that $\mathcal{X} \cup \{x, y\}$ satisfies (4.13).

Proof of $T \subseteq S$: Consider $y \in (\mathrm{Elig}(\mathcal{X}) \cap \mathrm{Elig}(x)) \setminus [\bigcup_{\substack{v \in Y_m[1]: \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq \mathbb{I}_\sigma}} \bigcap_{\iota \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_\iota]$. Hence, since $y \in \mathrm{Elig}(\mathcal{X})$, we have $y \in X_m \setminus \mathcal{X}$ and $\mathcal{X} \cup \{y\}$ satisfies (4.13). And since $y \in \mathrm{Elig}(x)$, we have $y \in X_m \setminus \{x\}$ and $\{x, y\}$ satisfies (4.13). From $y \in X_m \setminus \mathcal{X}$ and $y \in X_m \setminus \{x\}$, we deduce $y \in X_m \setminus (\mathcal{X} \cup \{x\})$.

In the following we will show that $\mathcal{X} \cup \{x, y\}$ satisfies (4.13). For that, $\forall v \in Y_m[1]$, we have to show that : if $\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) = \emptyset$ then $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota| = 1$. Two cases can be considered :

$\mathrm{ID}_v(\mathcal{X} \cup \{x\}) = \emptyset$: this implies that $\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) = \emptyset$. $x \in \mathrm{Elig}(\mathcal{X})$ implies $\mathcal{X} \cup \{x\}$ satisfies (4.13). The latter and $\mathrm{ID}_v(\mathcal{X} \cup \{x\}) = \emptyset$ implies $|(\mathcal{X} \cup \{x\}) \cap (\bigcup_{\iota \in \mathbb{I}_\sigma} v_\iota)| = 1$

(Lemma 4.5.5). This means that either a unique $z \in \mathcal{X}$ is contained in all v_i or x is contained in all v_i. This equivalent to consider the two following :

$\mathrm{ID}_v(\mathcal{X}) = \emptyset$ and $\mathrm{ID}_v(x) = \mathbb{I}_\sigma$: since $\mathrm{ID}_v(\mathcal{X}) = \emptyset$, we have $\mathrm{ID}_v(\mathcal{X} \cup \{y\}) = \emptyset$. And since $y \in \mathrm{Elig}(\mathcal{X})$, $|(\mathcal{X} \cup \{y\}) \cap (\bigcup_{i \in \mathbb{I}_\sigma} v_i)| = 1$. Thus, since, $\forall i \in \mathbb{I}_\sigma$, $z \in v_i$, it follows that $y \notin \bigcup_{i \in \mathbb{I}_\sigma} v_i$. Since $\mathrm{ID}_v(x) = \mathbb{I}_\sigma$ (i.e., $x \notin \bigcup_{i \in \mathbb{I}_\sigma} v_i$), we deduce that $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{i \in \mathbb{I}_\sigma} v_i| = 1$.

$\mathrm{ID}_v(x) = \emptyset$ and $\mathrm{ID}_v(\mathcal{X}) = \mathbb{I}_\sigma$: since $\mathrm{ID}_v(x) = \emptyset$, we have $\mathrm{ID}_v(\{x, y\}) = \emptyset$. And since $x \in \mathrm{Elig}(x)$, $|(\{x, y\}) \cap (\bigcup_{i \in \mathbb{I}_\sigma} v_i)| = 1$. Thus, since, $\forall i \in \mathbb{I}_\sigma$, $x \in v_i$, it follows that $y \notin \bigcup_{i \in \mathbb{I}_\sigma} v_i$. Since $\mathrm{ID}_v(\mathcal{X}) = \mathbb{I}_\sigma$ (i.e., $\mathcal{X} \cap \bigcup_{i \in \mathbb{I}_\sigma} v_i = \emptyset$), we conclude that $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{i \in \mathbb{I}_\sigma} v_i| = 1$.

$\mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq \emptyset$: two cases can be considered :

$\mathrm{ID}_v(\mathcal{X} \cup \{x\}) = \mathbb{I}_\sigma$: $\forall i \in \mathbb{I}_\sigma$, $v_i \cap (\mathcal{X} \cup \{x\}) = \emptyset$. If $\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) = \emptyset$, we have $\forall i \in \mathbb{I}_\sigma$, $y \in v_i$. It follows that $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{i \in \mathbb{I}_\sigma} v_i| = |\{y\}| = 1$.

$\mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq \mathbb{I}_\sigma$: since $y \notin \bigcup_{\substack{v \in Y_m[1] \cdot \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq \mathbb{I}_\sigma}} \bigcap_{i \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_i$, then there exists $i \in \mathbb{I}_\sigma$ s.t. $y \notin v_i$ and $v_i \cap (\mathcal{X} \cup \{x\}) = \emptyset$, thus, $v_i \cap (\mathcal{X} \cup \{x, y\}) = \emptyset$. Therefore, $\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) \neq \emptyset$, and thus from Lemma 4.5.5, we deduce $\mathcal{X} \cup \{x, y\}$ satisfies (4.13) w.r.t. v.

From $y \in (\mathrm{Elig}(\mathcal{X}) \cap \mathrm{Elig}(x)) \setminus [\bigcup_{\substack{v \in Y_m[1] \cdot \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq \mathbb{I}_\sigma}} \bigcap_{i \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_i]$, we have deduced $y \in X_m \setminus (\mathcal{X} \cup \{x, y\})$ and $\mathcal{X} \cup \{x, y\}$ satisfies (4.13), which means $y \in \mathrm{Elig}(\mathcal{X} \cup \{x\})$.

B.1.12 Proof of Lemme 4.5.6

Given $v \in Y_m[1]$ and $x \in X_m$, the most costly operation for computing $\mathrm{ID}_v(x)$ is checking whether $v_i \cap \{x\} = \emptyset$, $\forall i \in \mathbb{I}_\sigma$. Checking whether $v_i \cap \{x\} = \emptyset$ is in $O(|X|)$ because $v_i \subseteq X$. The complexity of checking for all v_i of v is obtained by multiplying the above complexity by n_σ, that is, $O(n_\sigma \cdot |X|)$.

B.1.13 Proof of Lemme 4.5.7

Given $x \in X_m$ and $v \in Y_m[1]$, the complexity for computing $v_i \cup v_{i+1}$ and $v_i \cap v_{i+1}$ is in $O(|X|^2)$, and the complexity of computing $\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) = \emptyset}} \bigcup_{i \in \mathbb{I}_\sigma} v_i$ and $\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) \neq \mathbb{I}_\sigma}} \bigcap_{i \in \mathrm{ID}_v(x)} v_i$ is obtained by multiplying the above complexity by n_σ, that is, $O(n_\sigma \cdot |X|^2)$.

If we consider all $v \in Y_m[1]$, the complexity of computing $\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) = \emptyset}} \bigcup_{i \in \mathbb{I}_\sigma} v_i$ and $\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_i(x) \neq \mathbb{I}_\sigma}} \bigcap_{i \in \mathrm{ID}_v(x)} v_i$ is obtained by multiplying the above complexity by $|Y_m[1]|$ that is, $O(n_\sigma \cdot |Y_m[1]| \cdot |X|^2)$. The union with $\{x\}$ is in $O(|X|)$. The subtraction $X_m \setminus \cdots$ is in $O(|X| \cdot |X_m|) \leq O(|X|^2)$. Therefore the complexity of computing $\mathrm{Elig}(x)$ is in $O(n_\sigma \cdot |Y_m[1]| \cdot |X|^2)$.

B.1.14 Proof of Lemma 4.5.8

We have seen in the proof of Lemma 4.5.7 that the complexity for computing $\bigcup_{\substack{v\in Y_m[1]\\ \mathrm{ID}_v(x)\neq\mathbb{I}_\sigma}} \bigcap_{i\in\mathrm{ID}_v(x)} v_i$ is $O(n_\sigma\cdot|Y_m[1]|\cdot|X|^2)$. The complexity of the intersection $\mathrm{Elig}(X)\cap\mathrm{Elig}(x)$ is in $O(|X_m|^2)\leq O(|X|^2)$. The complexity of the subtraction $[\mathrm{Elig}(X)\cap Elig(x)]\setminus\cdots$ is in $O(|X_m|\cdot|X|)\leq O(|X|^2)$. Therefore, the total complexity is $O(n_\sigma\cdot|Y_m[1]|\cdot|X|^2)$.

B.1.15 Proof of Proposition 4.5.6

Let us consider the 2 steps of the procedure.

1. Initializations : $Z_m\leftarrow X_m$ is in $O(|X_m|)$, $X_m^1\leftarrow\emptyset$ is in $O(1)$. Therefore Step 1 is in $O(|X_m|)$,
2. Compute $\mathrm{Elig}(X_m^1)$ knowing $\mathrm{Elig}(x)$ and $\mathrm{Elig}(X_m^1\setminus\{x\})$:
   Compute $\mathrm{Elig}(x)$ is in $O(n_\sigma\cdot|Y_m[1]|\cdot|X|^2)$ (Lemma 4.5.7).
   Compute $\mathrm{Elig}(X_m^1)$ from $\mathrm{Elig}(x)$ and $\mathrm{Elig}(X_m^1\setminus\{x\})$ is in $O(n_\sigma\cdot|Y_m[1]|\cdot|X|^2)$.

Then from $\mathrm{Elig}(X_m^1)$ and Z_m : computing $\mathrm{Elig}(X_m^1)\cap Z_m$ is in $O(|X_m|^2)\leq O(|X|^2)$; checking if $\mathrm{Elig}(X_m^1)\cap Z_m$ is empty is in $O(|X_m|^2)\leq O(|X|^2)$; selecting randomly x in $\mathrm{Elig}(X_m^1)\cap Z_m$ is in $O(1)$; and moving the selected x from Z_m to X_m^1 is in $O(1)$. To construct the whole partition, the while-loop is repeated at most $|X_m|$ times. Therefore, the total complexity is in $O(n_\sigma\cdot|Y_m[1]|\cdot|X|^2\cdot|X_m|)\leq O(n_\sigma\cdot|Y_m[1]|\cdot|X|^3)$.

B.2 Proofs of Section 4.6

B.2.1 Proof of Theorem 4.6.1

The algorithm computes $\mathcal{A}_{\mathcal{E}_\sigma}$, $\mathcal{A}_{\mathcal{D}_\sigma}$, $\mathcal{A}_{\mathcal{E}_\sigma[1]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[1]}$ as indicated in Subsection 4.5.1. If $Y_m[1]=0$ or $X_m[1]=0$ (thus, $\mathcal{E}_\sigma[1]=0$ or $\mathcal{D}_\sigma[1]=0$), then from Def. 4.4.2, $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is Inf_0-COOBS. This the only situation where the algorithm generates the output "$(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is Inf_0-COOBS". Therefore, the latter output is generated if and only if it is true.

The algorithm computes every $\mathrm{ID}_v(x)$, using its definition given by (4.14), and $\mathrm{Elig}(x)$, using (4.16) of Proposition 4.5.4. Then the algorithm constructs iteratively sets $X_m^1,\cdots,X_m^p$ that constitute a partition of X_m, such that each X_m^j satisfies (4.13). The construction of each X_m^j is based on : $\mathrm{ID}_v(X_m^j)$, which is computed using Lemma 4.5.4; and $\mathrm{Elig}(X_m^j)$, which is computed using 4.17 of Proposition 4.5.5.

Then, the algorithm searches the smallest $N_1,\cdots,N_p\leq N$ such that $Y_m[N_j+1]|_{X_m^j}=\emptyset$ or $X_m[N_j+1]|_{X_m^j}=\emptyset$, (thus, $\mathcal{E}_\sigma^j[N_j+1]=0$ or $\mathcal{D}_\sigma^j[N_j+1]=0$), for every $j=1,\cdots,p$. For that purpose, the algorithm computes $\mathcal{A}_{\mathcal{E}_\sigma[k]}$ and $\mathcal{A}_{\mathcal{D}_\sigma[k]}$ for $k=1,\cdots,\max(N_1,\cdots,N_p)$ as indicated in Subsection 4.5.1.

If such $(N_j)_{j=1,\cdots,p}$ exists (and thus, is found by the algorithm), then from Def. 4.4.2, every $(\mathcal{E}_\sigma^j,\mathcal{D}_\sigma)$ is Inf_{N_j}-COOBS. And from Def. 4.4.3, $(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is $\vee$-$(Inf_{N_1},\cdots,Inf_{N_p})$-COOBS. This the only situation where the algorithm generates the output "$(\mathcal{E}_\sigma,\mathcal{D}_\sigma)$ is

$\vee$-($Inf_{N_1}, \cdots, Inf_{N_p}$)-COOBS". Therefore, the latter output is generated if and only if it is true.

If such $(N_j)_{j=1,\cdots,p}$ does not exist (and thus, is not found by the algorithm), then from Theorem 4.5.1, $(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not $\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS. This is the only situation where the algorithm generates the output "$(\mathcal{E}_\sigma, \mathcal{D}_\sigma)$ is not $\vee$-$Inf^{\geq 1}_{\leq N}$-COOBS". Therefore, the latter output is generated if and only if it is true.

ANNEXE C

Preuves du chapitre 5

C.1 Proofs of Section 5.4

C.1.1 Proof of Proposition 5.4.1

"Only If" : Consider p D^j-diagnosers $(Diag^j)_{j\in J}$, whose global diagnoses are fused conjunctively. Assume that the $\wedge$-$(\mathrm{D}^1,\ldots,\mathrm{D}^p)$-diagnoser $Diag = ((Diag^j)_{j\in J},\wedge)$ satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F},\mathcal{H})$. Let us consider a set $\{\mathcal{H}^1,\ldots,\mathcal{H}^p\}$ of subsets of $\mathcal{H}$ defined as follows :

$$\forall j \in J, \mathcal{H}^j = \mathcal{H} \cap \{s|\, Diag^j(s) \neq 1\}. \tag{C.1}$$

Eq (C.1) implies that $s \notin \mathcal{H}^j$ if $s \notin \mathcal{H}$ or $Diag^j(s) = 1$, $\forall j \in J$. Let us now show that $\{\mathcal{H}^1,\ldots,\mathcal{H}^p\}$ is a decomposition of $\mathcal{H}$, i.e., $\bigcup_{j\in J}\mathcal{H}^j = \mathcal{H}$. Eq. (C.1) implies that $\mathcal{H}^j \subseteq \mathcal{H}$, $\forall j \in J$. Therefore, $\bigcup_{j\in J}\mathcal{H}^j \subseteq \mathcal{H}$. It remains to show that $\mathcal{H} \subseteq \bigcup_{j\in J}\mathcal{H}^j$, as follows :

$$\begin{aligned}
s \in \mathcal{H} &\Rightarrow Diag(s) \neq 1 && \text{(from } Diag \text{ satisfies Cond. (5.3) w.r.t. } \mathcal{H})\\
&\Rightarrow \exists j \in J \text{ s.t. } Diag^j(s) \neq 1 && \text{(from Eq. (5.7))}\\
&\Rightarrow \exists j \in J \text{ s.t. } s \in \mathcal{H}^j && \text{(from Eq. (C.1) and } s \in \mathcal{H})\\
&\Leftrightarrow s \in \bigcup_{j\in J}\mathcal{H}^j,
\end{aligned}$$

which means that $\mathcal{H} \subseteq \bigcup_{j\in J}\mathcal{H}^j$.

Cond. (5.3) : Let us show that, $\forall j \in J$, $Diag^j$ satisfies Cond. (5.3) w.r.t. $\mathcal{H}^j$. By the construction of the decomposition $\{\mathcal{H}^1,\ldots,\mathcal{H}^p\}$ of $\mathcal{H}$ defined by Eq. (C.1), we have, $\forall j \in J$, $Diag^j$ satisfies Cond. (5.3) w.r.t. $\mathcal{H}^j$.

Cond. (5.2) : Assume that $Diag$ satisfies Cond. (5.2) w.r.t $\mathcal{F}$, thus we have,

$$\begin{aligned}
\forall s \in \mathcal{F} : Diag(s) \neq 0 &\Rightarrow \forall j \in J, \forall s \in \mathcal{F} : Diag^j(s) \neq 0, && \text{(from Eq. (5.7))}\\
&\Leftrightarrow \forall j \in J, Diag^j \text{ satisfies Cond. (5.2) w.r.t. } \mathcal{F}.
\end{aligned}$$

Cond. (5.1) : Assume that $Diag$ satisfies Cond. (5.1) w.r.t $\mathcal{F}$, thus we have,

$$\begin{aligned}
\exists l \in \mathbb{Z}^+, \forall s \in \mathcal{F}_l : Diag(s) = 1 &\Rightarrow \exists l \in \mathbb{Z}^+, \forall s \in \mathcal{F}_l, \forall j \in J : Diag^j(s) = 1, && \text{(from Eq. (5.7))}\\
&\Leftrightarrow \forall j \in J, \exists l \in \mathbb{Z}^+, \forall s \in \mathcal{F}_l : Diag^j(s) = 1,\\
&\Leftrightarrow \forall j \in J, Diag^j \text{ satisfies Cond. (5.1) w.r.t. } \mathcal{F}.
\end{aligned}$$

"If" : Consider languages $\mathcal{F}$ and $\mathcal{H}$, and D^j-diagnosers $(Diag^j)_{j\in J}$. Assume that there exists a decomposition $\{\mathcal{H}^1,\ldots,\mathcal{H}^p\}$ of $\mathcal{H}$ such that, $\forall j \in J$, the D^j-diagnoser $Diag^j$ satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F},\mathcal{H}^j)$.

Cond. (5.1) : Since $Diag^j$ satisfies Cond. (5.1) w.r.t. $\mathcal{F}$, we have,

$$\forall j \in J, \exists l_j \in \mathbb{Z}^+, \forall s \in \mathcal{F}_{l_j} : Diag^j(s) = 1. \tag{C.2}$$

Consider $l_1,\ldots,l_p$ corresponding to Eq. C.2, and let $l = \max_{j\in J} l_j$. Then, $\forall j \in J$, if $s \in \mathcal{F}_l$ then $s \in \mathcal{F}_{l_j}$ (because $l_j \leq l$, and thus, $\mathcal{F}_l \subseteq \mathcal{F}_{l_j}$). Hence, from Eq. (C.2), $\forall s \in \mathcal{F}_l$, $\forall j \in J$, $Diag^j(s) = 1$. It follows from Eq. (5.7) that $Diag(s) = 1$. Therefore, $Diag$ satisfies Cond. (5.1) w.r.t. $\mathcal{F}$.

Cond. (5.2) : Since, $\forall j \in J$, the D^j-diagnoser $Diag^j$ satisfies Cond. (5.2) w.r.t. $\mathcal{F}$, we have,

$$\forall j \in J, \forall s \in \mathcal{F} : Diag^j(s) \neq 0. \tag{C.3}$$

Consider $s \in \mathcal{F}$, and thus, $\forall j \in j$, $Diag^j(s) \neq 0$ (from Eq. C.3). Hence, from Eq. (5.7), $Diag(s) \neq 0$. Therefore, $Diag$ satisfies Cond. (5.2) w.r.t. $\mathcal{F}$.

Cond. (5.3) : Since, $\forall j \in J$, the D^j-diagnoser $Diag^j$ satisfies Cond. (5.3) w.r.t. $\mathcal{H}^j$, we have,

$$\forall j \in J, \forall s \in \mathcal{H}^j : Diag^j(s) \neq 1. \tag{C.4}$$

Consider $s \in \mathcal{H}$, and thus, there exists $j \in J$ s.t. $s \in \mathcal{H}^j$ (because $\mathcal{H} = \bigcup_{j\in J} \mathcal{H}^j$), and then from Eq. (C.4), $Diag^j(s) \neq 1$. Hence, from Eq. (5.7), $Diag(s) \neq 1$. Therefore, $Diag$ satisfies Cond. (5.3) w.r.t. $\mathcal{H}$.

C.1.2 Proof of Proposition 5.4.2

"Only If" : Consider p D^j-diagnosers $(Diag^j)_{j\in J}$, whose global diagnoses are fused disjunctively. Assume that the $\vee$-$(\mathrm{D}^1,\ldots,\mathrm{D}^p)$-diagnoser $Diag = ((Diag^j)_{j\in J}, \vee)$ satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F},\mathcal{H})$. We consider an integer l such that $Diag(s) = 1$ for every $s \in \mathcal{F}_l$. Such l exists from the satisfaction of Cond. (5.1). Let us consider a set $\{\mathcal{F}^1,\ldots,\mathcal{F}^p\}$ of subsets of $\mathcal{F}$ defined as follows :

$$\forall j \in J, \mathcal{F}^j = \{s \in \mathcal{F}_l |\, Diag^j(s) = 1\} \cup \{s \in \mathcal{F}\backslash\mathcal{F}_l |\, Diag^j(s) \neq 0\}. \tag{C.5}$$

Let us now show that $\{\mathcal{F}^1,\ldots,\mathcal{F}^p\}$ is a decomposition of $\mathcal{F}$, i.e., $\bigcup_{j\in J} \mathcal{F}^j = \mathcal{F}$. Eq. (C.5) implies that $\mathcal{F}^j \subseteq \mathcal{F}$, $\forall j \in J$. Therefore, $\bigcup_{j\in J} \mathcal{F}^j \subseteq \mathcal{F}$. It remains to show that

$\mathcal{F} \subseteq \bigcup_{j \in J} \mathcal{F}^j$, as follows :

$$\begin{aligned}
s \in \mathcal{F} &\Rightarrow Diag(s) \neq 0, \qquad (Diag \text{ satisfies Cond. (5.2) w.r.t. } \mathcal{F}) \\
&\Leftrightarrow [s \in \mathcal{F}_l \wedge Diag(s) \neq 0] \vee [s \in \mathcal{F} \backslash \mathcal{F}_l \wedge Diag(s) \neq 0], \\
&\Leftrightarrow [s \in \mathcal{F}_l \wedge Diag(s) = 1] \vee [s \in \mathcal{F} \backslash \mathcal{F}_l \wedge Diag(s) \neq 0], \\
&\qquad (Diag \text{ satisfies Cond. (5.1) w.r.t. } \mathcal{F}) \\
&\Rightarrow [\exists j \in J : s \in \mathcal{F}_l \wedge Diag^j(s) = 1] \vee [\exists j \in J : s \in \mathcal{F} \backslash \mathcal{F}_l \wedge Diag^j(s) \neq 0], \\
&\qquad (\text{from Eq. (5.8)}) \\
&\Rightarrow \exists j \in J : s \in \mathcal{F}^j, \qquad (\text{from Eq. (C.5)}) \\
&\Leftrightarrow s \in \bigcup_{j \in J} \mathcal{F}^j,
\end{aligned}$$

which means that $\mathcal{F} \subseteq \bigcup_{j \in J} \mathcal{F}^j$.

Cond. (5.3) : Since $Diag$ satisfies Cond. (5.3) w.r.t. $\mathcal{H}$. Thus we have,

$$\begin{aligned}
\forall s \in \mathcal{H} : Diag(s) \neq 1 &\Rightarrow \forall j \in J, \forall s \in \mathcal{H} : Diag^j(s) \neq 1, \qquad (\text{from Eq. (5.8)}) \\
&\Leftrightarrow \forall j \in J, Diag^j \text{ satisfies Cond. (5.3) w.r.t. } \mathcal{H}.
\end{aligned}$$

Cond. (5.2) : Let us show that, $\forall j \in J$, $Diag^j$ satisfies Cond. (5.2) w.r.t. $\mathcal{F}^j$. By the construction of the decomposition $\{\mathcal{F}^1, \ldots, \mathcal{F}^p\}$ of $\mathcal{F}$ defined by Eq. (C.5), we have, $\forall j \in J$, $Diag^j$ satisfies Cond. (5.2) w.r.t. $\mathcal{F}^j$.

Cond. (5.1) : Consider the l from which the decomposition $(\mathcal{F}^j)_{j \in J}$ has been constructed using Eq. (C.5). Consider $s \in \mathcal{F}_l^j$ for some $j \in J$.

$$\begin{aligned}
s \in \mathcal{F}_l^j &\Rightarrow s \in \mathcal{F}_l \wedge s \in \mathcal{F}^j, \\
&\Rightarrow Diag^j(s) = 1 \qquad (\text{from Eq. (C.5)}).
\end{aligned}$$

"If" : Consider languages $\mathcal{F}$ and $\mathcal{H}$, and D^j-diagnosers $(Diag^j)_{j \in J}$. Assume that there exists a decomposition $\{\mathcal{F}^1, \ldots, \mathcal{F}^n\}$ of $\mathcal{F}$ such that, $\forall j \in J$, the D^j-diagnoser $Diag^j$ satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}^j, \mathcal{H})$.

Cond. (5.1) : Since, $\forall j \in J$, the D^j-diagnoser $Diag^j$ satisfies Cond. (5.1) w.r.t. $\mathcal{F}^j$, we have,

$$\forall j \in J, \exists l_j \in \mathbb{Z}^+, \forall s \in \mathcal{F}_{l_j}^j : Diag^j(s) = 1. \tag{C.6}$$

Consider $l_1, \ldots, l_p$ corresponding to Eq. C.6, and let $l = \max_{j \in J} l_j$. Consider $s \in \mathcal{F}_l$, and thus, $\exists j \in J$ such that $s \in \mathcal{F}_l^j$ (because $\mathcal{F}_l = \bigcup_{j \in J} \mathcal{F}_l^j$). Then, since $\mathcal{F}_l^j \subseteq \mathcal{F}_{l_j}^j$ (because $l_j \leq l$), $s \in \mathcal{F}_{l_j}^j$. It follows from Eq. (C.6) that $Diag^j(s) = 1$. Hence, from Eq. (5.8), $Diag(s) = 1$. Therefore, $Diag$ satisfies Cond. (5.1) w.r.t. $\mathcal{F}$.

Cond. (5.2) : Since, $\forall j \in J$, the D^j-diagnoser $Diag^j$ satisfies Cond. (5.2) w.r.t. $\mathcal{F}^j$, we have,

$$\forall j \in J, \forall s \in \mathcal{F}^j : Diag^j(s) \neq 0. \tag{C.7}$$

Consider $s \in \mathcal{F}$, and thus, $\exists j \in J$ s.t. $s \in \mathcal{F}^j$ (because $\mathcal{F} = \bigcup_{j \in J} \mathcal{F}^j$). Hence, from Eq. (C.7), $Diag^j(s) \neq 0$. it follows from Eq. (5.8) that $Diag(s) \neq 0$. Therefore, $Diag$ satisfies Cond. (5.2) w.r.t. $\mathcal{F}$.

Cond. (5.3) : Since, $\forall j \in J$, the D^j-diagnoser $Diag^j$ satisfies Cond. (5.3) w.r.t. $\mathcal{F}^j$, we have,

$$\forall j \in J, \forall s \in \mathcal{H} : Diag^j(s) \neq 1. \tag{C.8}$$

Consider $s \in \mathcal{H}$, and thus, $\forall j \in J$, $Diag^j(s) \neq 1$ (from Eq. (C.8)). It follows from Eq. (5.8) that $Diag(s) \neq 1$. Therefore, $Diag$ satisfies Cond. (5.3) w.r.t. $\mathcal{H}$.

C.2 Proofs of Section 5.6

C.2.1 Proof of Proposition 5.6.1

Consider a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$, and the $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-diagnoser $Diag = ((Diag^j)_{j \in J}, \wedge)$ defined by Eqs. (5.9)-(5.15) and (5.7) w.r.t. $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$. I.e., every $Diag^j$ is an Inf_{N_j}-diagnoser given by Eqs. (5.9)-(5.15) w.r.t. $(\mathcal{F}, \mathcal{H}^j)$. Hence, from Lemma 5.6.1, $\forall j \in J$, $Diag^j$ satisfies Conds. (5.2) and (5.3) w.r.t. $(\mathcal{F}, \mathcal{H}^j)$. It follows, from Proposition 5.4.1, that $Diag$ satisfies Conds. (5.2) and (5.3) w.r.t. $(\mathcal{F}, \mathcal{H})$.

C.2.2 Proof of Proposition 5.6.2

Consider a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$, and the $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-diagnoser $Diag = ((Diag^j)_{j \in J}, \wedge)$ defined by Eqs. (5.9)-(5.15) and (5.7) w.r.t. $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$. Hence, every $Diag^j$ is an Inf_{N_j}-diagnoser given by Eqs. (5.9)-(5.15) w.r.t. $(\mathcal{F}, \mathcal{H}^j)$. It follows, from Lemma 5.6.2, $\forall j \in J$, $Diag^j$ satisfies Cond. (5.1) w.r.t. $\mathcal{F}$. Therefore, from Proposition 5.4.1, $Diag$ satisfies Cond. (5.1) w.r.t. $\mathcal{F}$.

C.2.3 Proof of Theorem 5.6.2

"**Only If** :" Assume that $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG. Hence, from Definition 5.6.2, there exists a decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ such that, $\forall j \in J$, $(\mathcal{F}, \mathcal{H}^j)$ is Inf_{N_j}-F-CODIAG. Consider the multi-decision decentralized diagnoser $Diag = ((Diag^j)_{j \in J}, \wedge)$ defined by Eqs. (5.9)-(5.15) w.r.t. $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$. By Propositions 5.5.1, 5.6.1 and 5.6.2, $Diag$ is a $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-diagnoser satisfying Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H})$.

"**If**" : We consider a $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-diagnoser $Diag = ((Diag^j)_{j \in J}, \wedge)$, thus, $\forall j \in J$, $Diag^j$ is a Inf_{N_j}-diagnoser. We assume that $Diag$ satisfies Conds. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H})$. We consider the decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ defined by Eq. (C.1). We have shown in the proof of Proposition 5.4.1 that, $\forall j \in J$, $Diag^j$ satisfies Cond. (5.1)-(5.3) w.r.t. $(\mathcal{F}, \mathcal{H}^j)$.

Hence, from Theorem 5.6.1, $(\mathcal{F}, \mathcal{H}^j)$ is Inf_{N_j}-F-CODIAG for every $j \in J$. From Definition 5.6.2, we deduce that $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG.

C.3 Proofs of Section 5.7

C.3.1 Proof of Lemma 5.7.1

Let us first show by induction on k that, $\forall k \geq 0$, $\mathcal{H}^{\nu_1}[k] \subseteq \mathcal{H}^{\nu_2}[k]$ and $\mathcal{F}^{\nu_1}[k] \subseteq \mathcal{F}^{\nu_2}[k]$. In the basis case $k = 0$, we have $\mathcal{H}^{\nu_1}[0] = \mathcal{H}^{\nu_1} \subseteq \mathcal{H}^{\nu_2} = \mathcal{H}^{\nu_2}[0]$ and $\mathcal{F}^{\nu_1}[0] = \mathcal{F} \subseteq \mathcal{F} = \mathcal{F}^{\nu_2}[0]$. Assume that for a given integer $k \geq 0$, $\mathcal{F}^{\nu_1}[k] \subseteq \mathcal{F}^{\nu_2}[k]$ and $\mathcal{H}^{\nu_1}[k] \subseteq \mathcal{H}^{\nu_2}[k]$ and let us show that $\mathcal{H}^{\nu_1}[k+1] \subseteq \mathcal{H}^{\nu_2}[k+1]$ and $\mathcal{F}^{\nu_1}[k+1] \subseteq \mathcal{F}^{\nu_2}[k+1]$. Since P_i and P_i^{-1} are maps and $\mathcal{H}^{\nu_1}[k] \subseteq \mathcal{H}^{\nu_2}[k]$, we have, $\forall i \in I$, $P_i^{-1}P_i(\mathcal{H}^{\nu_1}[k]) \subseteq P_i^{-1}P_i(\mathcal{H}^{\nu_2}[k])$ and then $\bigcap_{i \in I} P_i^{-1}P_i(\mathcal{H}^{\nu_1}[k]) \subseteq \bigcap_{i \in I} P_i^{-1}P_i(\mathcal{H}^{\nu_2}[k])$. From the latter inclusion and the fact that $\mathcal{F}^{\nu_1}[k] \subseteq \mathcal{F}^{\nu_2}[k]$, we deduce

$$\mathcal{F}^{\nu_1}[k+1] = \mathcal{F}^{\nu_1}[k] \cap \bigcap_{i \in I} P_i^{-1}P_i(\mathcal{H}^{\nu_1}[k]) \subseteq \mathcal{F}^{\nu_2}[k] \cap \bigcap_{i \in I} P_i^{-1}P_i(\mathcal{H}^{\nu_2}[k]) = \mathcal{F}^{\nu_2}[k+1].$$

By the same approach, we can show that $\mathcal{H}^{\nu_1}[k+1] \subseteq \mathcal{H}^{\nu_2}[k+1]$.

Now we assume that $(\mathcal{F}, \mathcal{H}^{\nu_2})$ is Inf_N-F-CODIAG, i.e., there exists $l \in \mathbb{Z}^+$ such that $\mathcal{F}^{\nu_2}[N+1] \cap \mathcal{F}_l = \emptyset$. From the latter and the fact that $\mathcal{F}^{\nu_1}[N+1] \subseteq \mathcal{F}^{\nu_2}[N+1]$, we deduce that $\mathcal{F}^{\nu_1}[N+1] \cap \mathcal{F}_l = \emptyset$, i.e., $(\mathcal{F}, \mathcal{H}^{\nu_1})$ is Inf_N-F-CODIAG.

C.3.2 Proof of Proposition 5.7.1

"Only If" : Assume that there exists a state x of $\mathcal{A}_{\mathcal{H}}$ such that $(\mathcal{F}, \mathcal{L}(\mathcal{A}_{\mathcal{H}}, x))$ is not Inf_{N_x}-F-CODIAG for any $N_x \leq N$. For any decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ satisfying Assumption **A3** (w.r.t. $\mathcal{A}_{\mathcal{H}}$), $\exists j \in J$, such that $\mathcal{L}(\mathcal{A}_{\mathcal{H}}, x) \subseteq \mathcal{H}^j$. Since $(\mathcal{F}, \mathcal{L}(\mathcal{A}_{\mathcal{H}}, x))$ is not Inf_{N_x}-F-CODIAG for any $N_x \leq N$, then from Lemma 5.7.1 we have that $(\mathcal{F}, \mathcal{H}^j)$ is not Inf_{N_x}-F-CODIAG for any $N_x \leq N$. We have shown that for any decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ satisfying Assumption **A3** (w.r.t. $\mathcal{A}_{\mathcal{H}}$), we can find a $\mathcal{H}^j$ such that $(\mathcal{F}, \mathcal{H}^j)$ is not Inf_{N_x}-F-CODIAG for any $N_x \leq N$. Hence, from Definition 5.6.5, $(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$.

"If" : Assume that for every marked state x_j of $\mathcal{A}_{\mathcal{H}}$, $(\mathcal{F}, \mathcal{L}(\mathcal{A}_{\mathcal{H}}, x_j))$ is Inf_{N_j}-F-CODIAG for some $N_j \leq N$. This means that every $(\mathcal{F}, \mathcal{H}^j)$ is Inf_{N_j}-F-CODIAG for some $N_j \leq N$, for the special partition $(\mathcal{H}^j)_{j \in J}$ defined as follows. Every $\mathcal{H}^j = \mathcal{L}(\mathcal{A}_{\mathcal{H}}, x_j)$ for some marked state x_j of $\mathcal{A}_{\mathcal{H}}$, and conversely, for every marked state x_j of $\mathcal{A}_{\mathcal{H}}$ there exists a unique $\mathcal{H}^j$ such that $\mathcal{H}^j = \mathcal{L}(\mathcal{A}_{\mathcal{H}}, x_j)$. Note that his special partition satisfies Assumption **A3**, and its corresponding p is the number of states of $\mathcal{A}_{\mathcal{H}}$. Therefore, from Definition 5.6.4, $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$ for some $N_1, \ldots, N_p \leq N$. And from Definition 5.6.5, $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$.

The same kind of proof can be made for $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG.

C.3.3 Proof of Proposition 5.7.2

Consider FSAs $\mathcal{A}_{\mathcal{H},1}$ and $\mathcal{A}_{\mathcal{H},2}$ accepting $\mathcal{H}$ such that, for every marked state x of $\mathcal{A}_{\mathcal{H},1}$, there exists a state y of $\mathcal{A}_{\mathcal{H},2}$ such that $\mathcal{L}(\mathcal{A}_{\mathcal{H},1}, x) \subseteq \mathcal{L}(\mathcal{A}_{\mathcal{H},2}, y)$. Assume that $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H},2}$. Hence, from Proposition 5.7.1, for every marked state y of $\mathcal{A}_{\mathcal{H},2}$, $(\mathcal{F}, \mathcal{L}(\mathcal{A}_{\mathcal{H},2}, y))$ is Inf_{N_y}-F-CODIAG for some $N_y \leq N$. From $\mathcal{L}(\mathcal{A}_{\mathcal{H},1}, x) \subseteq \mathcal{L}(\mathcal{A}_{\mathcal{H},2}, y)$ and $(\mathcal{F}, \mathcal{L}(\mathcal{A}_{\mathcal{H},2}, y))$ is Inf_{N_y}-F-CODIAG, we deduce from Lemma 5.7.1 that $(\mathcal{F}, \mathcal{L}(\mathcal{A}_{\mathcal{H},1}, x))$ is Inf_{N_y}-F-CODIAG. To recapitulate, for every marked state x of $\mathcal{A}_{\mathcal{H},1}$, $(\mathcal{F}, \mathcal{L}(\mathcal{A}_{\mathcal{H},1}, x))$ is Inf_{N_x}-F-CODIAG for some $N_x \leq N$. It follows, from Proposition 5.7.1, that $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H},1}$.

C.3.4 Proof of Proposition 5.7.5

Assume that $(\mathcal{F}, \mathcal{H})$ is not diagnosable. Hence, $\forall l \in \mathbb{Z}^+$, there exist $s \in \mathcal{F}_l$ and $u \in \mathcal{H}$ such that $P(s) = P(u)$. For any decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$, there exists $j \in J$ such that $u \in \mathcal{H}^j$. Since $P(s) = P(u)$ we have $P_i(s) = P_i(u)$ for any $i \in I$. Let us prove by induction that, $\forall k \geq 0$, $s \in \mathcal{F}^j[k] \neq \emptyset$ and $u \in \mathcal{H}^j[k] \neq \emptyset$.

Basis : We have $s \in \mathcal{F} = \mathcal{F}^j[0]$ and $u \in \mathcal{H}^j = \mathcal{H}^j[0]$.

Induction step : Assume that $s \in \mathcal{F}^j[k]$ and $u \in \mathcal{H}^j[k]$ for some $k \geq 0$. Let us prove that $s \in \mathcal{F}^j[k+1]$ and $u \in \mathcal{H}^j[k+1]$. Since $P_i(s) = P_i(u)$ for any $i \in I$, we have $s \in \bigcap_{i \in I} P_i^{-1} P_i(\mathcal{H}^j[k])$ and $u \in \bigcap_{i \in I} P_i^{-1} P_i(\mathcal{F}^j[k])$. And since $s \in \mathcal{F}^j[k]$ and $u \in \mathcal{H}^j[k]$, we obtain $s \in \mathcal{F}^j[k] \cap \bigcap_{i \in I} P_i^{-1} P_i(\mathcal{H}^j[k]) = \mathcal{F}^j[k+1]$ and $u \in \mathcal{H}^j[k] \cap \bigcap_{i \in I} P_i^{-1} P_i(\mathcal{F}^j[k]) = \mathcal{H}^j[k+1]$.

We have proved by induction that $s \in \mathcal{F}^j[k]$ and $u \in \mathcal{H}^j[k]$, $\forall k \geq 0$. Since $s \in \mathcal{F}_l$, it follows that, $\forall N_j \geq 0$, $s \in \mathcal{F}_l \cap \mathcal{F}^j[N_j + 1] \neq \emptyset$, i.e., $(\mathcal{F}, \mathcal{H}^j)$ is not Inf_{N_j}-F-CODIAG, and then $(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG.

C.4 Proofs of Section 5.8

C.4.1 Proof of Lemma 5.8.1

Lemma 5.8.1 can be deduced from the following known result

Result 1 : Consider n nondeterministic FSA $A_1, \ldots, A_n$ over an alphabet of cardinality e. Let q_i be the number of states of A_i, for $i = 1 \ldots n$, and $q = q_1 \times \ldots \times q_n$. Let A be the synchronized product of all the $A_i, i = 1 \ldots n$. The number of states of A is in $O(q)$ and its number of transitions is in $O(q.(q.e)) = O(q^2.e)$.

Let us apply Result 1 to $\mathcal{A}_{\mathcal{H}[k+1]}$ which is the synchronized product of $P_i^{-1} P_i(\mathcal{A}_{\mathcal{F}[k]})$, $i = 1 \ldots n$, with $\mathcal{A}_{\mathcal{H}[k]}$. Each $P_i^{-1} P_i(\mathcal{A}_{\mathcal{F}[k]})$ is computed by a projection P_i of $\mathcal{A}_{\mathcal{F}[k]}$ without determinization and then by adding self-loops. Therefore the number of states of each $P_i^{-1} P_i(\mathcal{A}_{\mathcal{F}[k]})$ is in $O(|Y[k]|)$. Using Result 1, we obtain :

the number of states $|X[k+1]|$ of $\mathcal{A}_{\mathcal{H}[k+1]}$ is in $O(|X[k]| \cdot |Y[k]|^n)$,

the number of transitions $|\alpha[k+1]|$ of $\mathcal{A}_{\mathcal{H}[k+1]}$ is in

$$O(|X[k+1]^2| \cdot |\Sigma|) = O(|X[k]|^2 \cdot |Y[k]|^{2n} \cdot |\Sigma|).$$

With the same approach, if we apply Result 1 to $A_{\mathcal{F}[k+1]}$ which is the synchronized product of $\mathcal{A}_{\mathcal{F}[k]}$ with $P_i^{-1}P_i(\mathcal{A}_{\mathcal{H}[k]})$, $i = 1 \ldots n$, we obtain :
the number of states $|Y[k+1]|$ of $\mathcal{A}_{\mathcal{F}[k+1]}$ is in $O(|Y[k]| \cdot |X[k]|^n)$,
the number of transitions $|\beta[k+1]|$ of $\mathcal{A}_{\mathcal{F}[k+1]}$ is in

$$O(|Y[k+1]|^2 \cdot |\Sigma|) = O(|Y[k]|^2 \cdot |X[k]|^{2n} \cdot |\Sigma|).$$

C.4.2 Proof of Lemma 5.8.2

Lemma 5.8.2 can be deduced from the following known result.

Result 2 : The computational complexity for constructing an automaton is in the order of its number of transitions.

By applying Result 2 to $\mathcal{A}_{\mathcal{H}[k+1]}$, we obtain that the computational complexity for constructing $\mathcal{A}_{\mathcal{H}[k+1]}$ is in $O(|\alpha[k+1]|) = O(|X[k]|^2 \cdot |Y[k]|^{2n} \cdot |\Sigma|)$ (from Lemma 5.8.1). Let us apply Result 2 to $\mathcal{A}_{\mathcal{F}[k+1]}$ whose number of transitions is $|\beta[k+1]| = O(|Y[k]|^2 \cdot |X[k]|^{2n} \cdot |\Sigma|)$ (from Lemma 5.8.1). We obtain that the computational complexity for constructing $\mathcal{A}_{\mathcal{F}[k+1]}$ is in $O(|\beta[k+1]|) = O(|Y[k]|^2 \cdot |X[k]|^{2n} \cdot |\Sigma|)$.

C.4.3 Proof of Proposition 5.8.1

From Lemma 5.8.2, the complexity for computing $\mathcal{A}_{\mathcal{F}[1]}$ from $\mathcal{A}_{\mathcal{H}}$ and $\mathcal{A}_{\mathcal{F}}$ is in $O(|Y|^2 \cdot |X|^{2n} \cdot |\Sigma|)$. And detecting cycles of faulty states in $\mathcal{A}_{\mathcal{F}[1]}$ is linear in the size of $\mathcal{A}_{\mathcal{F}[1]}$, i.e., $O(|\beta[1]|)$. Therefore, the complexity of computing $\mathcal{A}_{\mathcal{F}[1]}$ and detecting cycles of faulty states in $\mathcal{A}_{\mathcal{F}[1]}$ is in $O(|\beta[1]|) = O(|Y|^2 \cdot |X|^{2n} \cdot |\Sigma|)$.

C.4.4 Proof of Proposition 5.8.2

Consider FSAs $\mathcal{A}_{\mathcal{F}}$ and $\mathcal{A}_{\mathcal{H}}$, and a decomposition $\{X_m^1, \ldots, X_m^p\}$ of X_m. By considering the decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ of $\mathcal{H}$ such that each $\mathcal{H}^j$ is given by Eq. (5.16), we show by induction on the inference steps k, that : $\mathcal{H}^j[k] = \{s \in \Sigma^* | \, \alpha[k](x_0[k], s) \in X_m[k]|_{X_m^j}\}$ and $\mathcal{F}^j[k] = \{s \in \Sigma^* | \, \beta[k](y_0[k], s) \in Y_m[k]|_{X_m^j}\}$.

Basis : Let us show that $\mathcal{H}^j[1] = \{s \in \Sigma^* | \, \alpha[1](x_0[1], s) \in X_m[1]|_{X_m^j}\}$,

$$\begin{aligned} s \in \mathcal{H}^j[1] &\Leftrightarrow [s \in \mathcal{H}^j] \wedge [\forall i \in I, s \in P_i^{-1}P_i(\mathcal{F})] \\ &\Leftrightarrow [s \in \mathcal{H}^j] \wedge [\forall i \in I, \exists t_i \in \mathcal{F} \text{ s.t. } P_i(s) = P_i(t_i)] \end{aligned}$$

From Eq. (5.16), $s \in \mathcal{H}^j$ is equivalent to $\alpha(x_0, s) \in X_m^j$, and hence,

$$s \in \mathcal{H}^j[1] \Leftrightarrow [\alpha(x_0, s) = x \in X_m^j] \wedge [\forall i \in I, \exists t_i \in \mathcal{F} \text{ s.t. } P_i(s) = P_i(t_i)].$$

Since $\mathcal{A}_{\mathcal{F}}$ is an acceptor of $\mathcal{F}$, we have, $\forall i \in I$, $\beta(y_0, t_i) = v_i$ for some $v_i \in Y_m$. Thus, from the synchronous composition of the FSAs $\mathcal{A}_{\mathcal{H}}$ and $(P_i^{-1}P_i(\mathcal{A}_{\mathcal{F}}))_{i\in I}$ [Hopcroft et Ullman, 1979], we have,

$$s \in \mathcal{H}^J[1] \Leftrightarrow \alpha[1](x_0[1], s) = u = (u_1, \ldots, u_n, x) \in X_m[1] \text{ s.t., } \forall i \in I, v_i \in u_i,$$

Since x is X_m^J-marked, we have u is X_m^J-marked, and hence,

$$s \in \mathcal{H}^J[1] \Leftrightarrow \alpha[1](x_0[1], s) \in X_m[1]|_{X_m^J}.$$

Let us show that $\mathcal{F}^J[1] = \{s \in \Sigma^* | \beta[1](y_0[1], s) \in Y_m[1]|_{X_m^J}\}$,

$$\begin{aligned} s \in \mathcal{F}^J[1] &\Leftrightarrow [s \in \mathcal{F}] \wedge [\forall i \in I, s \in P_i^{-1}P_i(\mathcal{H}^J)] \\ &\Leftrightarrow [s \in \mathcal{F}] \wedge [\forall i \in I, \exists t_i \in \mathcal{H}^J \text{ s.t. } P_i(s) = P_i(t_i)], \end{aligned}$$

From the latter equivalence and the fact that $\mathcal{A}_{\mathcal{F}}$ is an acceptor of $\mathcal{F}$, we deduce,

$$s \in \mathcal{F}^J[1] \Leftrightarrow [\exists y \in Y_m \text{ s.t. } \beta(y_0, s) = y] \wedge [\forall i \in I, \exists t_i \in \mathcal{H}^J \text{ s.t. } P_i(s) = P_i(t_i)].$$

From Eq. (5.16), we have, $\forall i \in I$, $\alpha(x_0, t_i) = u_i$ for some $u_i \in X_m^J$. Thus, from the synchronous composition of the FSAs $\mathcal{A}_{\mathcal{F}}$ and $(P_i^{-1}P_i(\mathcal{A}_{\mathcal{H}}))_{i\in I}$ [Hopcroft et Ullman, 1979], we have,

$$s \in \mathcal{F}^J[1] \Leftrightarrow \beta[1](y_0[1], s) = v = (v_1, \ldots, v_n, y) \in Y_m[1] \text{ s.t., } \forall i \in I, u_i \in v_i,$$

Since, $\forall i \in I$, u_i is X_m^J-marked, we have v is X_m^J-marked, and hence,

$$s \in \mathcal{F}^J[1] \Leftrightarrow \beta[1](y_0[1], s) \in Y_m[1]|_{X_m^J}.$$

Induction step : Assume that $\mathcal{H}^J[k] = \{s \in \Sigma^* | \alpha[k](x_0[k], s) \in X_m[k]|_{X_m^J}\}$ and $\mathcal{F}^J[k] = \{s \in \Sigma^* | \beta[k](y_0[k], s) \in Y_m[k]|_{X_m^J}\}$. And let us show that $\mathcal{H}^J[k+1] = \{s \in \Sigma^* | \alpha[k+1](x_0[k+1], s) \in X_m[k+1]|_{X_m^J}\}$ and $\mathcal{F}^J[k+1] = \{s \in \Sigma^* | \beta[k+1](y_0[k+1], s) \in Y_m[k+1]|_{X_m^J}\}$.

$$\begin{aligned} s \in \mathcal{H}^J[k+1] &\Leftrightarrow [s \in \mathcal{H}^J[k]] \wedge [\forall i \in I, s \in P_i^{-1}P_i(\mathcal{F}^J[k])] \\ &\Leftrightarrow [s \in \mathcal{H}^J[k]] \wedge [\forall i \in I, \exists t_i \in \mathcal{F}^J[k] \text{ s.t. } P_i(s) = P_i(t_i)], \end{aligned}$$

From the induction hypothesis, we have :

$$s \in \mathcal{H}^J[k] \Leftrightarrow \alpha[k](x_0[k], s) = x \in X_m[k] \text{ for some } X_m^J\text{-marked state } x \text{ in } \mathcal{A}_{\mathcal{H}[k]},$$

$$t_i \in \mathcal{F}^J[k] \Leftrightarrow \beta[k](y_0[k], t_i) = v_i \in Y_m[k] \text{ for some } X_m^J\text{-marked state } v_i \text{ in } \mathcal{A}_{\mathcal{F}[k]},$$

Thus, from the synchronous composition of the FSAs $\mathcal{A}_{\mathcal{H}[k]}$ and $(P_i^{-1}P_i(\mathcal{A}_{\mathcal{F}[k]}))_{i\in I}$ [Hopcroft et Ullman, 1979], s reaches a state $(u_1, \ldots, u_n, x)$ in $\mathcal{A}_{\mathcal{H}[k+1]}$ s.t. $\forall i \in I$, $v_i \in u_i$,

hence,

$$s \in \mathcal{H}^j[k+1] \Leftrightarrow \alpha[k+1](x_0[k+1], s) = (u_1, \ldots, u_n, x) \in X_m[k+1] \text{ s.t., } \forall \imath \in I, v_\imath \in u_\imath,$$

Therefore, from the definition of the multi-marking, we have,

$$s \in \mathcal{H}^j[k+1] \Leftrightarrow \alpha[k+1](x_0[k+1], s) \in X_m[k+1]|_{X_m^j}.$$

Therefore, $\mathcal{H}^j[k+1] = \{s \in \Sigma^* | \, \alpha[k+1](x_0[k+1], s) \in X_m[k+1]|_{X_m^j}\}$. The same procedure can be followed to show that $\mathcal{F}^j[k+1] = \{s \in \Sigma^* | \, \beta[k+1](y_0[k+1], s) \in Y_m[k+1]|_{X_m^j}\}$.

C.4.5 Proof of Theorem 5.8.2

Consider the set of marked states $X_m = \{x^1, \ldots, x^{|X_m|}\}$, and consider the trivial partition $T = \{\mathcal{H}^1, \ldots, \mathcal{H}^{|X_m|}\}$ of $\mathcal{H}$ such that, $\forall j \in \{1, \ldots, |X_m|\}$, $\mathcal{H}^j$ is given by Eq. (5.16), i.e., $\mathcal{H}^j = \mathcal{L}(\mathcal{A}_\mathcal{H}, x^j) = \{s \in \overline{K} | \, \alpha(x_0, s) = x^j\}$. From Proposition 5.7.1, the following assertions are equivalent :

(a) $(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf_{\leq N}^{\geq 1}$-F-CODIAG w.r.t. $\mathcal{A}_\mathcal{H}$,

(b) $\exists x^j \in X_m$ s.t. $(\mathcal{F}, \mathcal{L}(\mathcal{A}_\mathcal{H}, x^j)) = (\mathcal{F}, \mathcal{H}^j)$ is not Inf_N-F-CODIAG.

From Theorem 5.8.1, we have,

$(b) \Leftrightarrow \exists \mathcal{H}^j \in T$,
there exists a cycle of faulty states in the acceptor $\mathcal{A}_{\mathcal{F}^j[N+1]}$ of $\mathcal{F}^j[N+1]$.

From Proposition 5.8.2,

$$\mathcal{F}^j[N+1] = \{s \in \Sigma^* | \, \beta[N+1](y_0[N+1], s) \in Y_m[N+1]|_{\{x^j\}}\},$$

thus,

$(b) \Leftrightarrow \exists x^j \in X_m, \exists v \in Y_m[N+1]|_{\{x^j\}}$
s.t. v is reached in $\mathcal{A}_{\mathcal{F}[N+1]}$ through a cycle of faulty states.

For every decomposition $D = \{X_m^1, \ldots, X_m^p\}$ of X_m, $\exists X_m^l \in D$ such that $x^j \in X_m^l$. Therefore, since v is $\{x^j\}$-marked, v is X_m^l-marked. And then,

$(a) \Leftrightarrow (b) \Leftrightarrow \exists x^j \in X_m, \exists v \in Y_m[N+1]|_{X_m^l}$
s.t. v is reached in $\mathcal{A}_{\mathcal{F}[N+1]}$ through a cycle of faulty states.

C.4.6 Proof of Lemma 5.8.3

Consider a decomposition $D = \{X_m^1, \ldots, X_m^p\}$ of X_m s.t. D satisfies Cond. (5.18). Let us prove Lemma 5.8.3 by induction on the inference steps $k \geq 1$.

Basis : Let $v = (v_1, \ldots, v_n, v_{n+1}) \in Y_m[k]$ be a X_m^j-marked state such that $\mathcal{C}_f(v) \neq \emptyset$, for some $X_m^j \in D$. Then, since X_m^j satisfies Cond. (5.18), we have, $\forall (a_1, \ldots, a_n) \in (v_1 \cap$

$X_m^j) \times \ldots \times (v_n \cap X_m^j)$, $|\bigcup_{i \in I_\sigma} \{a_i\}| = 1$, i.e., $\forall i \in I$, $v_i \cap X_m^j = \{x\}$, for some $x \in X_m^j$. This means that v is $\{x\}$-marked and not $\mathcal{X}$-marked, $\forall \mathcal{X} \subseteq X_m^j$ s.t. $\mathcal{X} \neq \{x\}$.

Induction Step : For some $k \geq 1$, assume that for every marked state $u \in Y_m[k]|_{X_m^j}$, for which $\mathcal{C}_f(u) \neq \emptyset$, there exists $x \in X_m^j$ such that u is $\{x\}$-marked and not $\mathcal{X}$-marked, $\forall \mathcal{X} \subseteq X_m^j$ s.t. $\mathcal{X} \neq \{x\}$. Let us show that for every marked state $v \in Y_m[k+1]|_{X_m^j}$, for which $\mathcal{C}_f(v) \neq \emptyset$, there exists $z \in X_m^j$ such that v is $\{z\}$-marked and not $\mathcal{X}$-marked, $\forall \mathcal{X} \subseteq X_m^j$ s.t. $\mathcal{X} \neq \{z\}$. For that, let $v^{k+1} = (v_1^{k+1}, \ldots, v_n^{k+1}, v_{n+1}^{k+1})$ be a X_m^j-marked state in $X_m[k+1]$ (the superscript "k+1" stands for the inference step), such that $\mathcal{C}_f(v^{k+1}) \neq \emptyset$, which implies that $v^k = v_{n+1}^{k+1} \in Y_m[k]$ is X_m^j-marked (i.e., $v^k = v_{n+1}^{k+1} \in Y_m[k]|_{X_m^j}$). Let us show that $\mathcal{C}_f(v^k) \neq \emptyset$. Since $\mathcal{C}_f(v^{k+1}) \neq \emptyset$, there exists traces $\lambda, \mu, \nu \in \Sigma^*$, such that μ is faulty and $\lambda\mu^*\nu \subseteq \mathcal{L}(\mathcal{A}_{\mathcal{F}[k+1]}, v^{k+1})$, where $\mathcal{L}(\mathcal{A}_{\mathcal{F}[k+1]}, v^{k+1})$ is the set of traces reaching v^{k+1} in $\mathcal{A}_{\mathcal{F}[k+1]}$. Since for every trace s reaching v^{k+1} in $\mathcal{A}_{\mathcal{F}[k+1]}$, i.e., $s \in \mathcal{L}(\mathcal{A}_{\mathcal{F}[k+1]}, v^{k+1})$, we have s reaches $v_{n+1}^{k+1} = v^k$ in $\mathcal{A}_{\mathcal{F}[k]}$, i.e. $s \in \mathcal{L}(\mathcal{A}_{\mathcal{F}[k]}, v^k)$. Hence, $\mathcal{L}(\mathcal{A}_{\mathcal{F}[k+1]}, v^{k+1}) \subseteq \mathcal{L}(\mathcal{A}_{\mathcal{F}[k]}, v^k)$. Thus, $\lambda\mu^*\nu \subseteq \mathcal{L}(\mathcal{A}_{\mathcal{F}[k]}, v^k)$, i.e., $\mathcal{C}_f(v^k) \neq \emptyset$. Hence, from the induction hypothesis, there exists $y \in X_m^j$ such that $v^k = (v_1^k, \ldots, v_n^k, v_{n+1}^k) \in Y_m[k]$ is $\{y\}$-marked and not $\mathcal{X}$-marked in $Y_m[k]$, $\forall \mathcal{X} \subseteq X_m^j$ s.t. $\mathcal{X} \neq \{y\}$.

Assume for contradiction that v^{k+1} is $\mathcal{Y}$-marked for some subset $\mathcal{Y} \subseteq X_m^j$ such that $y \in \mathcal{Y}$ (otherwise, i.e., $y \notin \mathcal{Y}$, v^{k+1} is not $\mathcal{Y}$-marked). Hence, $\exists l \in I$ such that v_l^{k+1} contains a state $w^k = (w_1^k, \ldots, w_n^k, w_{n+1}^k) \in X_m[k]$ which is $\mathcal{Y}$-marked, hence $w^{k-1} = w_{n+1}^k$ is $\mathcal{Y}$-marked in $X_m[k-1]$ (see the definition of multi-marking in Subsection 5.8.3).

Since $w^k \in v_l^{k+1}$, and from the definition of the natural projection of FSA [Hopcroft et Ullman, 1979], there exits $s \in \mathcal{F}[k]$ reaching the state $v^k = v_{n+1}^{k+1} \in Y_m[k]$ and $t \in \mathcal{H}[k]$ reaching the state $w^k \in X_m[k]$ such that $P_l(s) = P_l(t)$. Since s reaches $v^k = (v_1^k, \ldots, v_n^k, v_{n+1}^k)$ in $\mathcal{A}_{\mathcal{F}[k]}$, then s reaches $v^{k-1} = v_{n+1}^k \in Y_m[k-1]$ in $\mathcal{A}_{\mathcal{F}[k-1]}$. And since t reaches $w^k = (w_1^k, \ldots, w_n^k, w_{n+1}^k) \in X_m[k]$ in $\mathcal{A}_{\mathcal{F}[k]}$, then t reaches $w^{k-1} = w_{n+1}^k \in X_m[k-1]$ in $\mathcal{A}_{\mathcal{H}[k-1]}$. Since $P_l(s) = P_l(t)$ and from $\mathcal{F}[k] = \mathcal{F}[k-1] \cap \bigcap_{i \in I} P_i^{-1} P_i(\mathcal{H}[k-1])$, we have $w^{k-1} \in v_l^k$ (recall that $v^k = (v_1^k, \ldots, v_n^k, v_{n+1}^k)$). And since w^{k-1} is $\mathcal{Y}$-marked and v^k is $\{y\}$-marked, we have v^k is $(\mathcal{Y} \cup \{y\})$-marked. This contradicts the fact that v^k is $\{y\}$-marked and not $\mathcal{X}$-marked in $Y_m[k]$, $\forall \mathcal{X} \subseteq X_m^j$ s.t. $\mathcal{X} \neq \{y\}$. Thus, $\mathcal{Y} = \{y\}$, and then w^k is $\{y\}$-marked. Therefore, v^{k+1} is $\{y\}$-marked and not $\mathcal{X}$-marked in $Y_m[k+1]$, $\forall \mathcal{X} \subseteq X_m^j$ s.t. $\mathcal{X} \neq \{y\}$.

C.4.7 Proof of Theorem 5.8.3

Consider a FSA $\mathcal{A}_\mathcal{H}$ accepting $\mathcal{H}$ and a decomposition $\{X_m^1, \ldots, X_m^p\}$ of X_m satisfying Cond. (5.18). **"If" :** Assume that $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf_{\leq N}^{\geq 1}$-CODIAG w.r.t. $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$, where, $\forall j \in J$, $\mathcal{H}^j = \{s \in \Sigma^* | \, \alpha(x_0, s) \in X_m^j\}$. Since, each $\mathcal{H}^j$ contains only the traces leading to marked states of X_m^j, the decomposition $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$ satisfies Assumption **A3**. Therefore, from Definitions 5.6.4 and 5.6.5, $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$Inf_{\leq N}^{\geq 1}$-CODIAG w.r.t. $\mathcal{A}_\mathcal{H}$.

"Only If" : Assume that $(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf_{\leq N}^{\geq 1}$-CODIAG w.r.t. $\{\mathcal{H}^1, \ldots, \mathcal{H}^p\}$, where $\mathcal{H}^j = \{s \in \Sigma^* | \, \alpha(x_0, s) \in X_m^j\}$. Then, $\exists j \in J$ such that $(\mathcal{F}, \mathcal{H}^j)$ is not Inf_N-CODIAG, i.e., there exists $v \in Y_m[N+1]$ that is X_m^j-marked and reached by traces through a cycle of faulty states in $\mathcal{A}_{\mathcal{F}[N+1]}$. Since $\mathcal{H}^j$ satisfies Cond. (5.18), this implies, from Lemma 5.8.3,

that there exists $x \in X_m^J$ for which v is $\{x\}$-marked and not $\mathcal{X}$-marked, $\forall \mathcal{X} \subseteq X_m^J$ s.t. $\mathcal{X} \neq \{x\}$.

For every decomposition $D = (\mathcal{X}^1, \ldots, \mathcal{X}^q)$ of X_m, $\exists \mathcal{X}^l \in D$ such that $x \in \mathcal{X}^l$, and thus v is $\mathcal{X}^l$-marked. Since, v is reached by traces through a cycle of faulty states in $\mathcal{A}_{\mathcal{F}[N+1]}$, this implies from Theorem 5.8.2, that $(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf_{\leq N}^{\geq 1}$-CODIAG w.r.t. $\mathcal{A}_{\mathcal{H}}$, as required.

C.4.8 Proof of Proposition 5.8.3

From Lemma 5.8.2, the complexity of computing $\mathcal{A}_{\mathcal{F}[k+1]}$ from $\mathcal{A}_{\mathcal{H}}[k]$ and $\mathcal{A}_{\mathcal{F}}[k]$ is in $O(|\beta[k+1]|) = O(|Y[k]|^2 \cdot |X[k]|^{2n} \cdot |\Sigma|)$. The complexity of computing $\mathcal{A}_{\mathcal{F}^J[k+1]}$ from $\mathcal{A}_{\mathcal{H}^J[k]}$ and $\mathcal{A}_{\mathcal{F}^J[k]}$ is in the same order because $\mathcal{A}_{\mathcal{F}^J[k+1]}$ is obtained from $\mathcal{A}_{\mathcal{F}[k+1]}$ by : 1) keeping marked only the states of $\mathcal{A}_{\mathcal{F}[k+1]}$ that are X_m^J-marked, and 2) removing all states which are not reachable from the X_m^J-marked states. This procedure is in the worst case in $O(|\beta[k+1]|)$. And detecting cycles of faulty states in $\mathcal{A}_{\mathcal{F}[k+1]}$ is linear in the size of $\mathcal{A}_{\mathcal{F}[k+1]}$, i.e., $O(|\beta[k+1]|)$. Therefore, the complexity of computing $\mathcal{A}_{\mathcal{F}[k+1]}$ from $\mathcal{A}_{\mathcal{H}}[k]$ and $\mathcal{A}_{\mathcal{F}}[k]$ and detecting cycles of faulty states in $\mathcal{A}_{\mathcal{F}[k+1]}$ is in $O(|\beta[1]|) = O(|Y[k]|^2 \cdot |X[k]|^{2n} \cdot |\Sigma|)$.

C.4.9 Proof of Lemma 5.8.4

Given two subsets $\mathcal{W}, \mathcal{Z} \subseteq X_m$, we have :

$$\begin{aligned}\mathrm{ID}_v(\mathcal{W} \cup \mathcal{Z}) &= \{i \in I \mid v_i \cap (\mathcal{W} \cup \mathcal{Z}) = \emptyset\} = \{i \in I \mid (v_i \cap \mathcal{W} = \emptyset) \wedge (v_i \cap \mathcal{Z} = \emptyset)\} \\ &= \{i \in I \mid v_i \cap \mathcal{W} = \emptyset\} \cap \{i \in I \mid v_i \cap \mathcal{Z} = \emptyset\} = \mathrm{ID}_v(\mathcal{W}) \cap \mathrm{ID}_v(\mathcal{Z}).\end{aligned}$$

C.4.10 Proof of Lemma 5.8.5

$\mathcal{X} \subseteq X_m$ satisfies Cond. (5.18) means :

1. $\forall v = (v_1, \ldots, v_{n+1}) \in Y_m[1]$ s.t. v is $\mathcal{X}$-marked and $\mathcal{C}_f(v) \neq \emptyset$,
2. $\forall (a_1, \ldots, a_n) \in (v_1 \cap \mathcal{X}) \times \ldots \times (v_n \cap \mathcal{X})$, we have $|\bigcup_{i \in I} \{a_i\}| = 1$.

The above Item 1 can be rewritten : $\forall v \in Y_m[1]$ s.t. $\mathcal{C}_f(v) \neq \emptyset$ and $\mathrm{ID}_v(\mathcal{X}) = \emptyset$. The above item 2 is equivalent to : $|\bigcup_{i \in I} (\mathcal{X} \cap v_i)| = 1$, which can be rewritten : $|\mathcal{X} \cap (\bigcup_{i \in I} v_i)| = 1$.

To recapitulate, $\mathcal{X}$ satisfies Cond. (5.18) is equivalent to : $\forall v \in Y_m[1]$ s.t. $\mathcal{C}_f(v) \neq \emptyset$: if $\mathrm{ID}_v(\mathcal{X}) = \emptyset$ then $|\mathcal{X} \cap \bigcup_{i \in I} v_i| = 1$.

C.4.11 Proof of Proposition 5.8.4

Consider $x, y \in X_m$. Let us prove that $y \in \mathrm{Elig}(x)$ iff $y \in X_m \setminus [\{x\} \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathcal{C}_f(v) \neq \emptyset \\ \mathrm{ID}_v(x) = \emptyset}} \bigcup_{i \in I} v_i \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathcal{C}_f(v) \neq \emptyset \\ \mathrm{ID}_v(x) \neq I}} \bigcap_{i \in \mathrm{ID}_v(x)} v_i]$. By definition, we have,

$$y \in \mathrm{Elig}(x) \Leftrightarrow [\{x, y\} \text{ satisfies Cond. (5.18)}] \wedge [y \in X_m \setminus \{x\}]$$

The set of states $A \subseteq X_m$ s.t., $\forall y \in A$, $\{x, y\}$ satisfies Cond. (5.18), can be obtained by computing the set of states $B \subseteq X_m$ s.t. $\forall z \in A$, $\{x, z\}$ does not satisfy Cond. (5.18). And then we have $A = B^c$, where B^c is the complementary of B. For every $z \in B$, we have,

$$\{x, z\} \text{ does not satisfy Cond. (5.18)} \tag{C.9}$$

$$(C.9) \Leftrightarrow \exists v \in Y_m[1] \text{ s.t. } \mathcal{C}_f(v) \neq \emptyset, \mathrm{ID}_v(\{x, z\}) = \emptyset \text{ and } |\{x, z\} \cap \bigcup_{i \in I} v_i| > 1$$

$$(C.9) \Leftrightarrow \exists v \in Y_m[1] \text{ s.t. } \mathcal{C}_f(v) \neq \emptyset, \mathrm{ID}_v(\{x, z\}) = \emptyset \text{ and } \{x, z\} \cap \bigcup_{i \in I} v_i = \{x, z\}$$

$\{x, z\} \cap \bigcup_{i \in I} v_i = \{x, z\}$ is equivalent to $\{x, z\} \subseteq \bigcup_{i \in I} v_i$ which is equivalent to say that $\mathrm{ID}_v(x) \neq I$ and $\mathrm{ID}_v(z) \neq I$. So, we have

$$(C.9) \Leftrightarrow \exists v \in Y_m[1] \text{ s.t. } \mathcal{C}_f(v) \neq \emptyset, \mathrm{ID}_v(\{x, z\}) = \emptyset, \mathrm{ID}_v(x) \neq I \text{ and } \mathrm{ID}_v(z) \neq I.$$

For every $v \in Y_m[1]$ s.t. $\mathcal{C}_f(v) \neq \emptyset$, $\mathrm{ID}_v(\{x, z\}) = \emptyset$, $\mathrm{ID}_v(x) \neq I$ and $\mathrm{ID}_v(z) \neq I$, two cases can be presented :

1. $\mathrm{ID}_v(x) = \emptyset$: in this case for every $z \in \bigcup_{i \in I} v_i$, $\{x, z\}$ does not satisfy Cond. (5.18).

2. $\mathrm{ID}_v(x) \neq \emptyset$: since $\mathrm{ID}_v(\{x, z\}) = \emptyset$, $\forall i \in I$, $\{x, z\} \cap v_i \neq \emptyset$. Hence, $\forall l \in I$ s.t. $x \notin v_l$, i.e., $\forall l \in \mathrm{ID}_v(x)$, we have $z \in v_l$. It follows that $z \in \bigcap_{l \in \mathrm{ID}_v(x)} v_l$. Therefore, for every $z \in \bigcap_{l \in \mathrm{ID}_v(x)} v_l$, $\{x, z\}$ does not satisfy Cond. (5.18)

Since, either $z \in \bigcup_{i \in I} v_i$ (in the case $\mathrm{ID}_v(x) = \emptyset$) or $z \in \bigcap_{l \in \mathrm{ID}_v(x)} v_l$ (in the case $\mathrm{ID}_v(x) \neq \emptyset$) imply that $\mathrm{ID}_v(z) \neq I$, we have,

$$\begin{aligned}(C.9) \Leftrightarrow\ & \exists v \in Y_m[1] \text{ s.t. } \mathcal{C}_f(v) \neq \emptyset : \\ & [\mathrm{ID}_v(x) \neq I \wedge \mathrm{ID}_v(x) = \emptyset \wedge z \in \bigcup_{i \in I} v_i] \vee [\mathrm{ID}_v(x) \neq I \wedge \mathrm{ID}_v(x) \neq \emptyset \wedge z \in \bigcap_{l \in \mathrm{ID}_v(x)} v_l]\end{aligned}$$

Since $\mathrm{ID}_v(x) = \emptyset$ implies $\mathrm{ID}_v(x) \neq I$, and $\mathrm{ID}_v(x) = \emptyset$ implies $\bigcap_{l \in \mathrm{ID}_v(x)} v_l = \emptyset$, we have,

$$\begin{aligned}(C.9) \Leftrightarrow\ & \exists v \in Y_m[1] \text{ s.t. } \mathcal{C}_f(v) \neq \emptyset : \\ & [\mathrm{ID}_v(x) = \emptyset \wedge z \in \bigcup_{i \in I} v_i] \vee [\mathrm{ID}_v(x) \neq I \wedge z \in \bigcap_{l \in \mathrm{ID}_v(x)} v_l] \\ \Leftrightarrow\ & z \in \bigcup_{\substack{v \in Y_m[1] \\ \mathcal{C}_f(v) \neq \emptyset \\ \mathrm{ID}_v(x) = \emptyset}} \bigcup_{i \in I} v_i \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathcal{C}_f(v) \neq \emptyset \\ \mathrm{ID}_v(x) \neq I}} \bigcap_{i \in \mathrm{ID}_v(x)} v_i\end{aligned}$$

Therefore, $B = \bigcup_{\substack{v \in Y_m[1] \\ \mathcal{C}_f(v) \neq \emptyset \\ \mathrm{ID}_v(x) = \emptyset}} \bigcup_{i \in I} v_i \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathcal{C}_f(v) \neq \emptyset \\ \mathrm{ID}_v(x) \neq I}} \bigcap_{i \in \mathrm{ID}_v(x)} v_i$. Thus, we have,

$$\begin{aligned} y \in \mathrm{Elig}(x) &\Leftrightarrow [y \notin B] \wedge [y \in X_m \setminus \{x\}] \\ &\Leftrightarrow y \in X_m \setminus [\{x\} \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathcal{C}_f(v) \neq \emptyset \\ \mathrm{ID}_v(x) = \emptyset}} \bigcup_{i \in I} v_i \cup \bigcup_{\substack{v \in Y_m[1] \\ \mathcal{C}_f(v) \neq \emptyset \\ \mathrm{ID}_v(x) \neq I}} \bigcap_{i \in \mathrm{ID}_v(x)} v_i] \end{aligned}$$

C.4.12 Proof of Proposition 5.8.5

Consider $\mathcal{X} \subseteq X_m$, $x \in \mathrm{Elig}(\mathcal{X})$. We have to prove the equality $S = T$, where $S = \mathrm{Elig}(\mathcal{X} \cup \{x\})$ and $T = (\mathrm{Elig}(\mathcal{X}) \cap \mathrm{Elig}(x)) \setminus [\bigcup_{\substack{v \in Y_m[1]: \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq I \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcap_{i \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_i]$. Let us prove "$S \subseteq T$" and "$T \subseteq S$". Hereafter, we consider only the states $v \in Y_m[1]$ satisfying $\mathcal{C}_f(v) \neq \emptyset$.

Proof of $S \subseteq T$: Consider $y \in \mathrm{Elig}(\mathcal{X} \cup \{x\})$, which means that $y \in X_m \setminus (\mathcal{X} \cup \{x\})$ and $\mathcal{X} \cup \{y, x\}$ satisfies Cond. (5.18). Hence, $y \in X_m \setminus \mathcal{X}$, $y \in X_m \setminus \{x\}$, $\mathcal{X} \cup \{y\}$ satisfies Cond. (5.18) and $\{y, x\}$ satisfies Cond. (5.18). Therefore, $y \in \mathrm{Elig}(\mathcal{X}) \cap \mathrm{Elig}(x)$.

We now prove ad adsurbum $y \notin \bigcup_{\substack{v \in Y_m[1]: \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq I \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcap_{i \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_i$. Assume that $y \in \bigcup_{\substack{v \in Y_m[1]: \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq I \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcap_{i \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_i$. Hence, $\exists v = (v_1, \ldots, v_{n+1}) \in Y_m[1]$ s.t. $\mathcal{C}_f(v) \neq \emptyset$, $\mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq I$ and, $\forall i \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})$, $y \in v_i$. From the latter, we deduce that $\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) = \emptyset$.

From $\mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq I$, we deduce $(\bigcup_{i \in I} v_i) \cap (\mathcal{X} \cup \{x\}) \neq \emptyset$. The latter expression and the facts that $y \notin \mathcal{X} \cup \{x\}$ and $y \in \bigcup_{i \in I} v_i$ imply that $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{i \in I} v_i| > 1$.

$\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) = \emptyset$ and $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{i \in I} v_i| > 1$ imply that $\mathcal{X} \cup \{x, y\}$ does not satisfy Cond. (5.18) (Lemma 5.8.5), which contradicts the hypothesis that $\mathcal{X} \cup \{x, y\}$ satisfies Cond. (5.18).

Proof of $T \subseteq S$: Consider $y \in (\mathrm{Elig}(\mathcal{X}) \cap \mathrm{Elig}(x)) \setminus [\bigcup_{\substack{v \in Y_m[1]: \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq I \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcap_{i \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_i]$. Hence, since $y \in \mathrm{Elig}(\mathcal{X})$, we have $y \in X_m \setminus \mathcal{X}$ and $\mathcal{X} \cup \{y\}$ satisfies Cond. (5.18). And since $y \in \mathrm{Elig}(x)$, we have $y \in X_m \setminus \{x\}$ and $\{x, y\}$ satisfies Cond. (5.18). From $y \in X_m \setminus \mathcal{X}$ and $y \in X_m \setminus \{x\}$, we deduce $y \in X_m \setminus (\mathcal{X} \cup \{x\})$.

In the following we will show that $\mathcal{X} \cup \{x, y\}$ satisfies Cond. (5.18). For that, $\forall v \in Y_m[1]$ s.t. $\mathcal{C}_f(v) \neq \emptyset$, we have to show that : if $\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) = \emptyset$ then $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{i \in I} v_i| = 1$. Two cases can be considered :

$\mathrm{ID}_v(\mathcal{X} \cup \{x\}) = \emptyset$: this implies that $\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) = \emptyset$. $x \in \mathrm{Elig}(\mathcal{X})$ implies $\mathcal{X} \cup \{x\}$ satisfies Cond. (5.18). The latter and $\mathrm{ID}_v(\mathcal{X} \cup \{x\}) = \emptyset$ implies $|(\mathcal{X} \cup \{x\}) \cap (\bigcup_{i \in I} v_i)| = 1$ (Lemma 5.8.5). This means that either a unique $z \in \mathcal{X}$ is contained in all v_i or x is contained in all v_i. This equivalent to consider the two following :

$\mathrm{ID}_v(\mathcal{X}) = \emptyset$ and $\mathrm{ID}_v(x) = I$: since $\mathrm{ID}_v(\mathcal{X}) = \emptyset$, we have $\mathrm{ID}_v(\mathcal{X} \cup \{y\}) = \emptyset$. And since $y \in \mathrm{Elig}(\mathcal{X})$, $|(\mathcal{X} \cup \{y\}) \cap (\bigcup_{i \in I} v_i)| = 1$. Thus, since, $\forall i \in I$, $z \in v_i$, it follows that $y \notin \bigcup_{i \in I} v_i$. Since $\mathrm{ID}_v(x) = I$ (i.e., $x \notin \bigcup_{i \in I} v_i$), we deduce that $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{i \in I} v_i| = 1$.

$\mathrm{ID}_v(x) = \emptyset$ and $\mathrm{ID}_v(\mathcal{X}) = I$: since $\mathrm{ID}_v(x) = \emptyset$, we have $\mathrm{ID}_v(\{x, y\}) = \emptyset$. And since $x \in \mathrm{Elig}(x)$, $|(\{x, y\}) \cap (\bigcup_{i \in I} v_i)| = 1$. Thus, since, $\forall i \in I$, $x \in v_i$, it follows that $y \notin \bigcup_{i \in I} v_i$. Since $\mathrm{ID}_v(\mathcal{X}) = I$ (i.e., $\mathcal{X} \cap \bigcup_{i \in I} v_i = \emptyset$), we conclude that $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{i \in I} v_i| = 1$.

$\mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq \emptyset$: two cases can be considered :

$\mathrm{ID}_v(\mathcal{X} \cup \{x\}) = I$: $\forall i \in I$, $v_i \cap (\mathcal{X} \cup \{x\}) = \emptyset$. If $\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) = \emptyset$, we have $\forall i \in I$, $y \in v_i$. It follows that $|(\mathcal{X} \cup \{x, y\}) \cap \bigcup_{i \in I} v_i| = |\{y\}| = 1$.

$\mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq I$: since $y \notin \bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq I \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcap_{i \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_i$, then there exists $i \in I$ s.t. $y \notin v_i$ and $v_i \cap (\mathcal{X} \cup \{x\}) = \emptyset$, thus, $v_i \cap (\mathcal{X} \cup \{x, y\}) = \emptyset$. Therefore, $\mathrm{ID}_v(\mathcal{X} \cup \{x, y\}) \neq \emptyset$, and thus from Lemma 5.8.5, we deduce $\mathcal{X} \cup \{x, y\}$ satisfies Cond. (5.18) w.r.t. v.

From $y \in (\mathrm{Elig}(\mathcal{X}) \cap \mathrm{Elig}(x)) \setminus [\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(\mathcal{X} \cup \{x\}) \neq I \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcap_{i \in \mathrm{ID}_v(\mathcal{X} \cup \{x\})} v_i]$, we have deduced $y \in X_m \setminus (\mathcal{X} \cup \{x, y\})$ and $\mathcal{X} \cup \{x, y\}$ satisfies Cond. (5.18), which means $y \in \mathrm{Elig}(\mathcal{X} \cup \{x\})$.

C.4.13 Proof of Lemme 5.8.6

Given $v \in Y_m[1]$ s.t. $\mathcal{C}_f(v) \neq \emptyset$, and $x \in X_m$, the most costly operation for computing $\mathrm{ID}_v(x)$ is checking whether $v_i \cap \{x\} = \emptyset$, $\forall i \in I$. Checking whether $v_i \cap \{x\} = \emptyset$ is in $O(|X|)$ because $v_i \subseteq X$. The complexity of checking for all v_i of v is obtained by multiplying the above complexity by n, that is, $O(n \cdot |X|)$.

C.4.14 Proof of Lemme 5.8.7

Given $x \in X_m$ and $v \in Y_m[1]$ such that $\mathcal{C}_f(v) \neq \emptyset$, the complexity for computing $v_i \cup v_{i+1}$ and $v_i \cap v_{i+1}$ is in $O(|X|^2)$, and the complexity of computing $\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) = \emptyset \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcup_{i \in I} v_i$ and $\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) \neq I \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcap_{i \in \mathrm{ID}_v(x)} v_i$ is bounded by multiplying the above complexity by n, that is, $O(n \cdot |X|^2)$.

The complexity of computing $\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) = \emptyset \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcup_{i \in I} v_i$ and $\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) \neq I \\ \mathcal{C}_f(v) \neq \emptyset}} \bigcap_{i \in \mathrm{ID}_v(x)} v_i$ is bounded by multiplying the above complexity by $|Y_m[1]|$ that is, $O(n \cdot |Y_m[1]| \cdot |X|^2)$. The union with $\{x\}$ is in $O(|X|)$. The subtraction $X_m \setminus \ldots$ is in $O(|X| \cdot |X_m|) \leq O(|X|^2)$. Therefore the complexity of computing $\mathrm{Elig}(x)$ is in $O(n \cdot |Y_m[1]| \cdot |X|^2)$.

C.4.15 Proof of Lemma 5.8.8

The complexity for computing $\bigcup_{\substack{v \in Y_m[1] \\ \mathrm{ID}_v(x) \neq I \\ C_f(v) \neq \emptyset}} \bigcap_{v_i \in \mathrm{ID}_v(x)} v_i$ is $O(n \cdot |Y_m[1]| \cdot |X|^2)$ (from Lemma 5.8.7). The complexity of the intersection $\mathrm{Elig}(X) \cap \mathrm{Elig}(x)$ is in $O(|X_m|^2) \leq O(|X|^2)$. The complexity of the subtraction $[\mathrm{Elig}(X) \cap Elig(x)] \setminus \ldots$ is in $O(|X_m| \cdot |X|) \leq O(|X|^2)$. Therefore, the total complexity of computing $\mathrm{Elig}(X)$ is $O(n \cdot |Y_m[1]| \cdot |X|^2)$.

C.4.16 Proof of Proposition 5.8.6

Let us consider the 2 steps of the procedure.

1. Initializations : $Z_m \leftarrow X_m$ is in $O(|X_m|)$, $X_m^1 \leftarrow \emptyset$ is in $O(1)$. Therefore Step 1 is in $O(|X_m|)$,
2. Compute $\mathrm{Elig}(X_m^1)$ knowing $\mathrm{Elig}(x)$ and $\mathrm{Elig}(X_m^1 \setminus \{x\})$:
   Compute $\mathrm{Elig}(x)$ is in $O(n \cdot |Y_m[1]| \cdot |X|^2)$ (Lemma 5.8.7).
   Compute $\mathrm{Elig}(X_m^1)$ from $\mathrm{Elig}(x)$ and $\mathrm{Elig}(X_m^1 \setminus \{x\})$ is in $O(n \cdot |Y_m[1]| \cdot |X|^2)$ (Lemma 5.8.8).

Then from $\mathrm{Elig}(X_m^1))$ and Z_m : computing $\mathrm{Elig}(X_m^1)) \cap Z_m$ is in $O(|X_m|^2) = O(|X|^2)$; checking if $\mathrm{Elig}(X_m^1)) \cap Z_m$ is empty is in $O(|X_m|^2) \leq O(|X|^2)$; selecting randomly x in $\mathrm{Elig}(X_m^1) \cap Z_m$ is in $O(1)$; and moving the selected x from Z_m to X_m^1 is in $O(1)$. To construct the whole partition, the while-loop is repeated at most $|X_m|$ times. Therefore, the total complexity is in $O(n \cdot |Y_m[1]| \cdot |X|^2 \cdot |X_m|) \leq O(n \cdot |Y_m[1]| \cdot |X|^3)$.

C.5 Proofs of Section 5.9

C.5.1 Proof of Theorem 5.9.1

The algorithm computes $\mathcal{A}_{\mathcal{H}}$, $\mathcal{A}_{\mathcal{F}}$, $\mathcal{A}_{\mathcal{H}[1]}$ and $\mathcal{A}_{\mathcal{F}[1]}$ as indicated in Subsection 5.8.1. If there is no cycle of faulty states in $Y_m[1]$, then from Def. 5.6.1, $(\mathcal{F}, \mathcal{H})$ is Inf_0-F-CODIAG. This the only situation where the algorithm generates the output "$(\mathcal{F}, \mathcal{H})$ is Inf_0-F-CODIAG". Therefore, the latter output is generated if and only if it is true.

The algorithm computes every $\mathrm{ID}_v(x)$, using its definition given by Eq. (5.19), and $\mathrm{Elig}(x)$, using Eq. (5.21) of Proposition 5.8.4. Then the algorithm constructs iteratively sets $X_m^1, \ldots, X_m^p$ that constitute a partition of X_m, such that each X_m^j satisfies Cond. (5.18). The construction of each X_m^j is based on : $\mathrm{ID}_v(X_m^j)$, which is computed using Lemma 5.8.4; and $\mathrm{Elig}(X_m^j)$, which is computed using Eq. (5.22) of Proposition 5.8.5.

Then, the algorithm searches the smallest $N_1, \ldots, N_p \leq N$ such that there is no cycle of faulty states in $\mathcal{A}_{\mathcal{F}[N_j+1]}$, for every $j = 1, \ldots, p$. For that purpose, the algorithm computes $\mathcal{A}_{\mathcal{H}[k]}$ and $\mathcal{A}_{\mathcal{F}[k]}$ for $k = 1, \ldots, \max(N_1, \ldots, N_p)$ as indicated in subsection 5.8.1.

If such $(N_j)_{j=1,\ldots,p}$ exists (and thus, is found by the algorithm), then from Def. 5.6.1, every $(\mathcal{F}, \mathcal{H}^j)$ is Inf_{N_j}-F-CODIAG. And from Def. 5.6.2, $(\mathcal{F}, \mathcal{H})$ is $\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG. This the only situation where the algorithm generates the output "$(\mathcal{F}, \mathcal{H})$ is

$\wedge$-$(Inf_{N_1}, \ldots, Inf_{N_p})$-F-CODIAG". Therefore, the latter output is generated if and only if it is true.

If such $(N_j)_{j=1, \ldots, p}$ does not exist (and thus, is not found by the algorithm), then from Theorem 5.8.3, $(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG. This is the only situation where the algorithm generates the output "$(\mathcal{F}, \mathcal{H})$ is not $\wedge$-$Inf^{\geq 1}_{\leq N}$-F-CODIAG". Therefore, the latter output is generated if and only if it is true.

LISTE DES RÉFÉRENCES

Abdelwahed, A. (2002). *Interacting discrete event systems : modeling, verification an supervisory control.* Thèse de doctorat, University of Toronto, Toronto, Ontario, Canada.

Abdelwahed, S. et Wonham, W. (2002). Supervisory control of interacting discrete event systems. Dans *Conference on Decision and Control (CDC).* IEEE, Las Vegas, NV, USA, p. 1175 1180.

Aghasaryaiu, A., Fabre, E., Benveniste, A., Boubour, R. et Jard, C. (1997). A Petri net approach to fault detection and diagnosis in distributed systems. Dans *Conference on Decision and Control (CDC).* IEEE, San Diego, USA, p. 720 725.

Akesson, K., Flordal, H. et Fabian, M. (2002). Exploiting modularity for synthesis and verification of supervisors. Dans *15th Triennial World Congress of the International Federation of Automatic Control.* IFAC, Barcelona, Spain.

Baroni, P., Lamperti, G., Poglianob, P. et Zanella, M. (1999). Diagnosis of large active systems. *Artificial Intelligence*, volume 110, numéro 1, p. 135 183.

Barrett, W. et Couch, J. (1979). *Compiler Construction : Theory and Practice.* Science Research Associates Inc.

Bavishi, S. et Chon, E. (1994). Automated fault diagnosis using a discrete event systems framework. Dans *International Symposium on Intelligent Control.* IEEE, Columbus, Ohio, USA, p. 213 218.

Boel, R. K. et van Schuppen, J. H. (2002). Decentralized failure diagnosis for discrete event systems with constrained communication between diagnosers. Dans *International Workshop on Discrete Event Systems (WODES).* IEEE, Zaragoza, Spain, p. 175 181.

Brandt, R. D., Garg, V. K., Kumar, R., Lin, F., Marcus, S. I. et Wonham, W. M. (1990). Formulas for calculating supremal and normal sublanguages. *Systems and Control Letters*, volume 15, numéro 8, p. 111 117.

Cardoso, J., Kunzle, L. et Valett, R. (1995). Petri net based reasoning for the diagnosis of dynamic discrete event systems. Dans *6th International Fuzzy Systems Association World Congress.* IFSA, Sao Paulo, Brazil, p. 333 336.

Cassandras, G. et Lafortune, S. (1999). *Introduction to Discrete Event Systems.* Kluwer Academic Publishers.

Chakib, H. et Khoumsi, A. (2008a). Multi-decision C&P∨D&A architecture for the decentralized control of discrete event systems. Dans *Conference on Automation Science and Engineering (CASE).* IEEE, Washington, D.C., USA, p. 187 193.

Chakib, H. et Khoumsi, A. (2008b). Multi-decision decentralized control of discrete event systems : Application to the C&P architecture. Dans *International Workshop on Discrete Event Systems (WODES).* IEEE, Göteborg, Sweden, p. 480 485.

Chakib, H. et Khoumsi, A. (2009). Multi-decision diagnosis : Parallel decentralized architectures cooperating for diagnosing discrete event systems. Dans *European Control Conference (ECC)*. IFAC, Budapest, Hungry

Chakib, H. et Khoumsi, A. (2011a). Multi-decision diagnosis : Decentralized architectures cooperating for diagnosing the presence of faults in discrete event systems. *Submitted to Journal of Discrete Event Dynamic Systems : Theory & Applications.*

Chakib, H. et Khoumsi, A. (2011b). Multi-decision supervisory control : Parallel decentralized architectures cooperating for controlling discrete event systems. *Accepted for publication in IEEE Transactions on Automatic Control.*

Chakib, H. et Khoumsi, A. (2011c). Verification of coobservability in the context of multi-decision supervisory control of discrete event systems. *Submitted to IEEE Transactions on Automatic Control.*

Cieslak, R., Desclaux, C., Fawaz, A. et Varaiya, P. (1988). Supervisory control of discrete event processes with partial observations. *IEEE Transactions on Automatic Control*, volume 33, numéro 3, p. 249 260.

Cook, J A., Kolmanovsky, I. V., McNamara, D., Nelson, E. C. et Prasad, K. V. (2007). Control, computing and communications : Technologies for the twenty-first century model. *Proceedings of the IEEE*, volume 95, numéro 2, p. 334 355.

Darwiche, A. et Provan, G. (1996). Exploiting system structure in model-based diagnosis of discrete event systems. Dans *Proceedings of the Seventh International Workshop on the Principles of Diagnosis (DX-96)*. p. 95 105.

de Queiroz, M H. et Cury, J. E. R. (2000a). Modular control of composed systems. *American Control Conference (ACC)*, p. 4051 4055.

de Queiroz, M. H. et Cury, J. E. R. (2000b). Modular supervisory control of large scale discrete-event systems. *Discrete Event Systems : Analysis and Control*, p. 103 110.

de Queiroz, M. H., Cury, J. E. R. et Wonham, W. M. (2005). Multitasking supervisory control of discrete-event systems. *Discrete Event Dynamic Systems : Theory & Applications*, volume 15, p. 375 395.

Debouk, R., Lafortune, S. et Teneketzis, D. (2000). Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamical Systems : Theory & Applications*, volume 10, numéro 1-2, p. 33 86.

Debouk, R., Lafortune, S. et Teneketzis, D. (2003). On the effect of communication delays in failure diagnosis of decentralized discrete event systems. *Discrete Event Dynamic Systems : Theory & Applications*, volume 13, numéro 3, p. 263 289.

Dorf, R. C. et Bishop, R. H. (2001). *Modern Control Systems.* Prentice-Hall, Englewood Cliffs.

Dubuisson, B. (2001). *Automatique et statistiques pour le diagnostic.* Traité IC2 Information, commande, communication, Hermès Sciences.

Fabre, E., Benveniste, A. et Jard, C (2002). Distributed diagnosis for large discrete event dynamic systems. Dans *15th IFAC World Congress*. IFAC, Barcelona, Spain.

Frank, P. (1990). Fault diagnosis in dynamic systems using analytical and knowledge based redundancy. *Automatica*, volume 26, p. 459 474.

Frank, P. M. (1996). Analytical and qualitative model-based fault diagnosis - a survey and some new results. *European Journal of Control*, volume 2, p. 6 28.

Franklin, G. F., Powell, J. D. et Emani-Naeini, A. (1994). *Feedback Control of Dynamics Systems*. Addison Wesley, Reading, MA.

Gaudin, B. et Marchand, H. (2004). Modular supervisory control of a class of concurrent discrete event systems. Dans *Workshop on Discrete Event Systems (WODES)*. IEEE, Reims, France, p. 181 186.

Gertler, J. (1998). *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker.

Hamscher, W., Console, L. et de Kleer, J. (1992a). *Readings in model-based diagnosis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Hamscher, W., Console, L. et de Kleer, J. (1992b). *Readings in Model-based Diagnosis*. Morgan Kaufmann Publishers, CA, USA.

Hoare, C. A. R. (1985). *Communicating Sequential Processes*. Prentice-Hall.

Hopcroft, J. E. et Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

Jiang, S., Huang, Z., Chandra, V. et Kumar, R. (2001). A polynomial time algorithm for diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, volume 46, numéro 8, p. 1318 1321.

Jiang, S. et Kumar, R. (2000). Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, volume 30, numéro 5, p. 653 660.

Khoumsi, A (2005). Coordination of components in a distributed discrete-event system. Dans *International Symposium on Parallel and Distributed Computing (ISPDC)* IEEE Computer Society, Washington, DC, USA, p. 299 306.

Khoumsi, A. et Chakib, H. (2007). Decentralized supervisory control of discrete event systems : involving the fusion system in the decision-making. Dans *International Conference on Intelligent Systems and Control (ISC)*. ACTA, Cambridge, Massachussets, USA, p. 44 49.

Khoumsi, A. et Chakib, H. (2008a). A multi-decision approach for decentralized diagnosis of the presence and absence of faults in discrete event systems Dans *Mediterranean Conference on Control and Automation (MED)*. IEEE, Ajaccio, France, p. 406 412.

Khoumsi, A. et Chakib, H. (2008b). A new architecture for decentralized control of discrete event systems : Decidability and synthesis issues. Dans *Conférence francophone de modélisation et simultation*. MOSIM, Paris, France.

Khoumsi, A. et Chakib, H. (2009). Multi-decision decentralized prognosis of failures in discrete event systems. Dans *Americain Control Conference (ACC)*. IEEE, St. Louis, Missouri, USA, p. 4974 4981.

Kumar, R. et Garg, V. K. (1995). *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, Boston, MA.

Kumar, R., Garg, V. K. et Marcus, S. I. (1991). On controllability and normality of discrete event systems. *Systems and Control Letters*, volume 17, numéro 3, p. 157 168.

Kumar, R. et Takai, S. (2005). Inference-based ambiguity management in decentralized decision-making : Decentralized control of discrete event systems. Dans *Conference on Decision and Control (CDC)*. IEEE, Seville, Spain, p. 3480 3485.

Kumar, R. et Takai, S. (2006). Inference-based ambiguity management in decentralized diagnosis-making : Decentralized diagnosis of discrete event systems. Dans *American Control Conference (ACC)*. IEEE, Minneapolis, USA, p. 6069 6074.

Kumar, R. et Takai, S. (2007). Inference-based ambiguity management in decentralized decision-making : Decentralized control of discrete event systems. *IEEE Transactions on Automatic Control*, volume 52, numéro 10, p. 1783 1794.

Kumar, R. et Takai, S. (2008). Decentralized prognosis of failures in discrete event systems. Dans *International Workshop on Discrete Event Systems (WODES)*. IEEE, Goteborg, Sweden, p. 376 381.

Kumar, R. et Takai, S. (2009). Inference-based ambiguity management in decentralized decision-making : Decentralized diagnosis of discrete-event systems. *IEEE transactions on automation science and engineering*, volume 6, numéro 3, p. 479 491.

Lafortune, S. (2007). *On decentralized and distributed control of partially-observed discrete event systems*, chapitre Advances in Control Theory and Applications, *volume 353*. Springer Berlin, Heidelberg, p. 171 184.

Lamperti, G. et Zanella, M. (2002). Diagnosis of discrete-event systems from uncertain temporal observations. *Artificial Intelligence*, volume 137, numéro 1-2, p. 91 163.

Lee, W., Grosh, D., Tillman, F. et Lie, C. (1985). Fault tree analysis, methods, and applications - a review. *IEEE Transactions on Reliability*, volume 34, numéro 3, p. 194 203.

Lin, F. (1994). Diagnosability of discrete-event systems and its applications. *Discrete Event Dynamic Systems : Theory and Applications*, volume 4, numéro 2, p. 197 212.

Lin, F. et Wonham, W. M. (1988a). Decentralized supervisory control of discrete event systems. *Information Sciences*, volume 44, p. 199 224.

Lin, F. et Wonham, W M. (1988b). On observability of discrete event systems. *Information Sciences*, volume 44, numéro 3, p. 173 198.

Lin, F. et Wonham, W. M. (1990). Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, volume 35, numéro 12, p. 1330 1337.

Milner, R. (1980). *A calculus for communicating systems*, Lecture Notes in Computer Science, *volume 92*. Springer-Verlag.

Minhas, R. S. et Wonham, W. M. (2003). Online supervision of discrete event systems. Dans *American Control Conference (ACC)*. IEEE, Denver, Colorado, USA, p 1685 1690.

Overkamp, A. et van Schuppen, J. H. (2001). Maximal solutions in decentralized supervisory control. *SIAM Journal on Control and Optimization*, volume 39, numéro 2, p. 492 511.

Perrow, C. (1984). *Normal Accidents : Living with High Risk Technologies*. Basis Books, Inc., New York.

Pouliezos, A D. et Stavrakakis, G. S. (1994). *Real time fault monitoring of industrial processes*. Kluwer Academic Publisher

Prosser, J. H., Kam, M. et Kwatny, H. G. (1997). Decision fusion and supervisor synthesis in decentralized discrete-event systems. Dans *American Control Conference (ACC)*. IEEE, Albuquerque, New Mexico, USA, p. 2251 2255.

Qiu, W. et Kumar, R. (2004). Decentralized failure diagnosis of discrete event systems Dans *International Workshop on Discrete Event Systems (WODES)*. IEEE, Reims, France.

Qiu, W. et Kumar, R (2005). Distributed failure diagnosis under bounded-delay communication of immediately forwarded local observations. Dans *American Control Conference (ACC)*. IEEE, Portland, OR, USA, p. 1027 1032.

Qiu, W. et Kumar, R. (2006). Decentralized failure diagnosis of discrete event systems. *IEEE Transactions on Systems, Man & Cybernetics Part A*, volume 36, numéro 2, p. 384 395.

Ramadge, P. J. et Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, volume 25, numéro 1, p. 206 230.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, volume 32, p. 57 95.

Rich, S et Venkatasubramanian, V. (1987). Model-based reasoning in diagnostic expert systems for chemical process plants *Computers and Chemical Engineering*, volume 11, numéro 2, p. 111 122.

Ricker, S. et Rudie, K. (2000). Know means no : Incorporating knowledge into discrete-event control systems. *IEEE Transactions on Automatic Control*, volume 45, numéro 9, p. 1656 1668.

Ricker, S. L. et Rudie, K. (2003). Knowledge is a terrible thing to waste : using inference in discrete-event control problems. Dans *American Control Conference (ACC)*. IEEE, Denver, Colorado, USA, p. 2246 2251.

Rudie, K. et Willems, J. C. (1995). The computational complexity of decentralized discrete event control problems. *IEEE Transactions on Automatic Control*, volume 40, numéro 7, p. 1313 1319.

Rudie, K. et Wonham, W. M. (1992). Think globally, act locally : decentralized supervisory control. *IEEE Transactions on Automatic Control*, volume 31, numéro 11, p. 1692 1708.

Sampath, M., Lafortune, S. et Teneketzis, D. (1998). Active diagnosis of discrete event systems. *IEEE Transactions on Automatic Control*, volume 43, numéro 7, p. 908 929.

Sampath, M., Sengupta, R., Lafortune, S., Sinaamohideen, K. et Teneketzis, D. (1995). Diagnosability of discrete event systems. *IEEE Transactions on Autommatic Control*, volume 40, numéro 9, p. 1555 1575.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K. et Teneketzis, D. (1996). Failure diagnosis using discrete-event models. *IEEE Transactions Control Systems Technology*, volume 4, numéro 2, p. 105 124.

Sangiovanni-Vincentelli, A. (2007). Quo vadis, SLD ? reasoning about the trends and challenges of system level design. *Proceedings of the IEEE*, volume 95, numéro 3, p. 467 506.

Sengupta, R. (1998). Diagnosis and communication in distributed systems. Dans *International Workshop on Discrete Event Systems (WODES)*. IEEE, Cagliari, Italy, p. 144 151.

Sengupta, R. et Tripakis, S. (2002). Decentralized diagnosability of regular languages is undecidable. Dans *Conference on Decision and Control (CDC)*. IEEE, Las Vegas, USA, p. 423 428

Su, R. et Wonham, W. (2000). Decentralized fault diagnosis for discrete event systems. Dans *Proc. 2000 CISS*. p. 1 6.

Su, R., Wonham, W. M., Kurien, J. et Koutsoukos, X. (2002). Distributed diagnosis for qualitative systems. Dans *International Workshop on Discrete Event Systems (WODES)*. IEEE, Zaragoza, Spain, p. 169 174.

Takai, S. et Kumar, R. (2006). Decentralized diagnosis for nonfailures of discrete event systems using inference-based ambiguity management. Dans *Int. Workshop on Discrete Event Systems (WODES)*. IEEE, Ann Arbor, USA.

Takai, S. et Kumar, R. (2008). Inference-based decentralized prognosis in discrete event systems. Dans *Conference on Decision and Control (CDC)*. IEEE, Cancun, Mexico, p. 871 876.

Takai, S. et Kumar, R. (2010). Decentralized diagnosis for nonfailures of discrete event systems using inference-based ambiguity management. *IEEE Transactions on Automatic Control*, volume 40, numéro 2, p. 406 412.

Takai, S. et Ushio, T. (2005). Decentralized supervisory control of discrete event systems using dynamic default control. *IEICE Transactions on Fundamentals*, volume E88-A, numéro 11, p. 2982 2988.

Tripakis, S. (2004). Undecidable problems in decentralized observation and control for regular languages. *Information Processing Letters*, volume 90, numéro 1, p. 21 28.

Viswanadham, N. et Narahari, Y. (1992). *Performance Modeling of Automated Manufacturing Systems*. Prentice-Hall Inc.

Wang, Y., Yoo, T.-S. et Lafortune, S. (2004). New results on decentralized diagnosis of discrete event systems. Dans *42nd Annual Allerton Conference on Communication, Control, and Computing*. Allerton Conference, Allerton, IL, USA.

Wang, Y., Yoo, T.-S. et Lafortune, S. (2005). Decentralized diagnosis of discrete event systems using unconditional and conditional decisions. Dans *Conferences on Decision and Control (CDC)*. IEEE, Seville, Spain, p. 6298 6304.

Wang, Y., Yoo, T.-S. et Lafortune, S. (2007). Diagnosis of discrete event systems using decentralized architectures. *Discrete Event Dynamic Systems : Theory & Applications*, volume 17, numéro 2, p. 233 263.

Willner, Y. et Heymann, M. (1991). Supervisory control of concurrent discrete-event systems. *International Journal of Control*, volume 54, numéro 5, p. 1143 1169.

Willsky, A. S. (1976). A survey of design methods for failure detection in dynamic systems. *Automatica*, volume 12, p. 601 611.

Wonham, W. M. (2008). *Supervisory control of discrete-event systems* (Rapport technique ECE 1636F/1637S). Department of Electrical and Computer Engineering, University of Toronto.

Wonham, W. M. et Ramadge, P. J. (1987). On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, volume 23, numéro 3, p. 637 659.

Yoo, T.-S. et Lafortune, S. (2002a). A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems : Theory and Applications*, volume 12, numéro 3, p. 335 377.

Yoo, T.-S. et Lafortune, S. (2002b). Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Transactions on Automatic Control*, volume 47, numéro 9, p. 1491 1495.

Yoo, T.-S. et Lafortune, S. (2004). Decentralized supervisory control with conditional decisions : Supervisor existence. *IEEE Transactions on Automatic Control*, volume 49, numéro 11, p. 1886 1904.

Zad, S. H., Kwong, R. et Wonham, W. (1998). Fault diagnosis in discrete-event systems. Dans *Conference on Decision and Control (CDC)*. IEEE, Tampa, Florida, USA, p. 3769 3774.

Zwingelstein, G. (1995). *Diagnostic des défaillances*. Traité des Nouvelles Technologies, Hermès.