Thèse

Université Paris Sud 11

présentée pour obtenir le grade de

Docteur de l'Université Paris Sud

Spécialité : Informatique

Par

Vincent Armant

Diagnostic distribué de systèmes respectant la confidentialité

(Importance des structures arborescentes)

M.	Philippe DAGUE, Directeur de Thèse, Université Paris Sud	Directeur de thèse
M.	Laurent SIMON, Maître de conférence, Université Paris Sud	Co-directeur de thèse
M.	Philippe JÉGOU, Professeur, Université Paul Cézanne	Rapporteur
M.	Pierre MARQUIS, Professeur, Université d'Artois	Rapporteur
Mme	Marie-Odile CORDIER, Professeur, Université Rennes 1	Examinatrice
M.	Alain DENISE, Professeur, Université Paris Sud	Examinateur

Laboratoire de Recherche en Informatique, U.M.R. CNRS 8623,
Bat, 650, Université Paris-Sud, 91405 Orsay Cedex, France

Remerciements

Je souhaiterais commencer par cette citation d'Aimé Césaire :
"C'est quoi une vie d'homme ? C'est le combat de l'ombre et de la lumière... C'est une lutte entre l'espoir et le désespoir, entre la lucidité et la ferveur... Je suis du côté de l'espérance, mais d'une espérance conquise, lucide, hors de toute naïveté."

Tout comme les hommes, les arbres mènent ce combat vers la lumière afin de produire le fruit de nouveaux arbres. Je tiens à remercier à travers cet arbre tous ceux qui ont participé aux fruits de cette thèse.

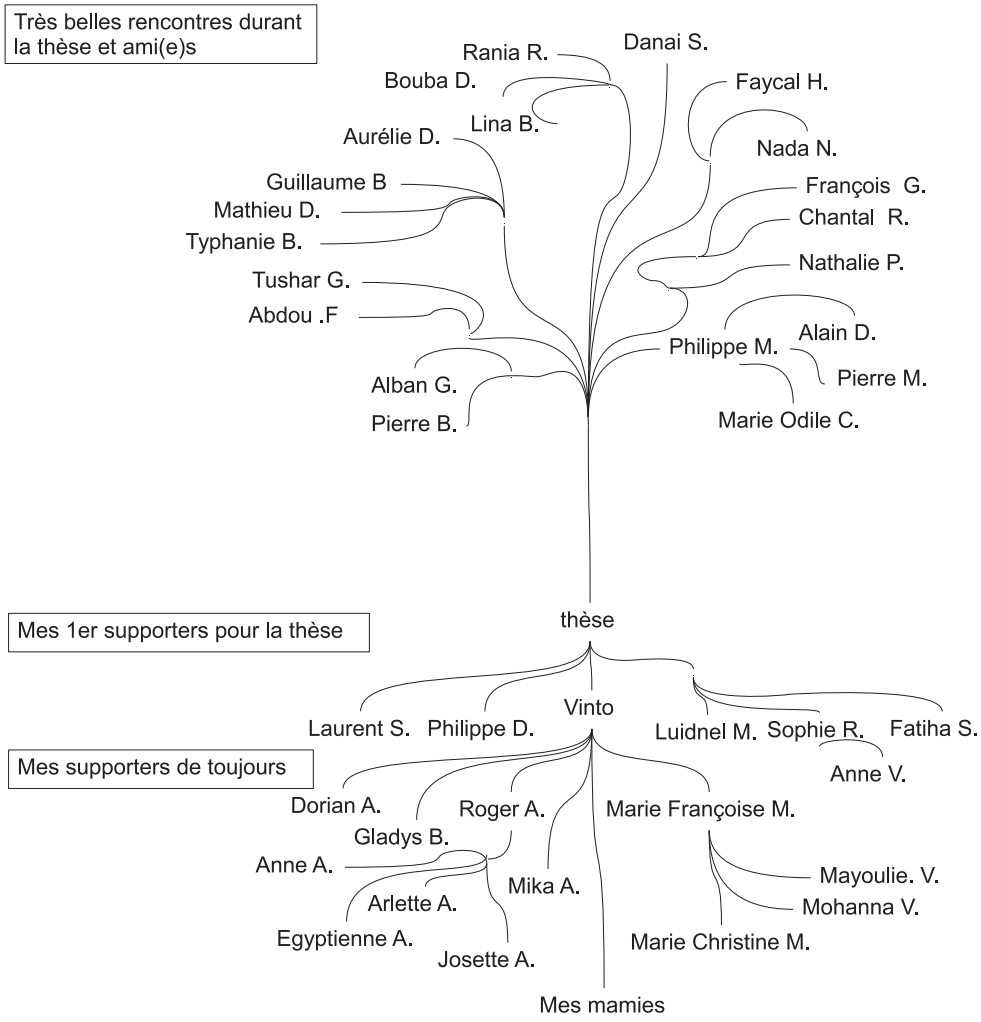


Figure 1 – Arbre de remerciements

Je dédie cette thèse
à mes parents, ma famille, mes proches
pour tout l'amour dont ils font preuve jour après jour.

Table des matières

Table des définitions, théorèmes, propriétés

Table des figures

Chapitre 1

Introduction

Cette thèse s'inscrit dans le cadre du diagnostic à base de modèles. L'originalité de notre approche réside principalement dans la volonté de pouvoir traiter des problèmes intrinsèquement répartis où certaines données resteront privées et ne pourront absolument pas être divulguées sur le réseau. Le diagnostic à base de modèles a été introduit dans les années 80 par Reiter et de Kleer [Rei87, dKW87] et a depuis été largement diffusé dans nombre de domaines connexes. La problématique de diagnostic a su rapidement trouver des applications fructueuses dans l'analyse des circuits électroniques [SWP88] et, aujourd'hui encore, le cadre du diagnostic est très utilisé pour résoudre un grand nombres de problèmes, comme la surveillance de systèmes embarqués automobiles [WR03], la configuration de systèmes [FFJS04], le débogage de programmes [SVAV05] ou encore plus récemment dans l'aide à la décision pour des problèmes liés à l'environnement [Wot11].

Ce cadre est assez général pour pouvoir accepter différentes modélisations de systèmes. Dans tous les cas, cependant, le comportement attendu du système doit être modélisé. Il peut se modéliser par le biais de formalismes logiques [FPvG08] (c'est le choix que l'on a fait dans ce manuscrit), d'automates [PC05], de réseaux de Petri [BHFJ03] ou encore grâce à des algèbres de processus [CPR02]. Cette modélisation va permettre de confronter ce que l'on attend du système avec ce qu'il calcule effectivement. Ainsi, à partir d'observations mesurées sur le système, des hypothèses (correspondant au *diagnostic*) sont émises afin d'expliquer un écart entre le comportement attendu du système, tel qu'il a été modélisé, et son comportement observé, tel qu'il a été mesuré.

Lorsque la description du système est centralisée, c-à-d connue d'un unique pair (au sens des systèmes pair-à-pair), la littérature fournit déjà un ensemble d'algorithmes et de *structures* relativement efficaces pour traiter la problématique de diagnostic. Par le terme structure, nous entendons un ensemble de propriétés graphiques respectées par un problème (éventuellement transformé) permettant de garantir de bonnes performances pour énumérer ou calculer les diagnostics pour une ou tout ensemble d'observations. Lorsque la description du système est distribuée, jusqu'à présent le calcul des diagnostics se fait par l'intermédiaire d'un superviseur de telle sorte que les approches envisagées dans le contexte centralisé s'appliquent aussi dans le cadre du diagnostic de système distribué. Il existe ainsi, dans ce contexte, des langages très expressifs tels que les réseaux de Petri, les automates, et plus généralement les systèmes à évènements discrets [CL99] servant à décrire le comportement d'un système distribué. Dans cette thèse, nous avons choisi de représenter les comportements attendus et observés des différents composants du système à diagnostiquer par le langage de la logique propositionnelle. Dans ce cadre, nous réutilisons le formalisme décrit par de Kleer, Mackworth et Reiter dans [dKMR92].

Même si la logique propositionnelle est moins expressive que les systèmes à évènements discrets, en pratique, beaucoup de travaux, y compris ceux du diagnostic de systèmes [GARK07], ont montré que plusieurs problèmes complexes non traitables par les algorithmes de l'état de l'art fondés pourtant sur des formalismes plus expressifs, devenaient traitables par la recherche de modèles des solveurs SAT, bénéficiant ainsi des progrès fulgurants obtenus en pratique dans ce domaine ces dernières années. Parce que le diagnostic à base de cohérence décrit par De Kleer se résout aussi par la recherche de modèles, nous considérons que les formules propositionnelles en entrée de ce formalisme pourront aussi bien provenir, dans un but d'efficacité, de la traduction dans un langage propositionnel de descriptions de comportements distribués exprimés dans des langages plus expressifs.

Dans ce manuscrit, nous nous intéresserons à diagnostiquer des systèmes intrinsèquement distribués, comme les systèmes pairs-à-pairs, dans lesquels chaque pair ne pourra communiquer qu'avec un nombre limité de voisins (ses accointances) par le biais de messages. Les pairs du réseau ne pourront communiquer avec l'extérieur que par l'intermédiaire de leurs accointances et, détail important, ne pourront pas créer de nouveaux liens pendant le calcul. De plus, en raison d'une politique d'accès, chaque pair ne pourra avoir accès qu'à la connaissance d'une sous-partie de la description du système global. Ainsi, aucun pair n'aura la connaissance de la description globale du système distribué. De plus, il se pourra qu'aucun pair ni superviseur ne soit en mesure d'expliquer le comportement global du système distribué, si la politique d'accès est trop restrictive. Ce nouveau contexte nous permet d'énoncer le défi du diagnostic distribué en ces termes :

L'objectif du diagnostic distribué avec contrôle d'accès consiste à pouvoir expliquer le comportement global d'un système distribué par un ensemble de pairs, ayant chacun une vision limitée de la description globale du système tout comme l'aurait fait un unique pair diagnostiqueur ayant, lui, une vision globale du système.

Tout d'abord, après un rappel du diagnostic à base de cohérence, nous introduisons plus précisément, chapitre 2, notre problématique du diagnostic distribué par rapport aux travaux de l'état de l'art du diagnostic distribué [Pro02, RtTW03, BNF08], des systèmes d'inférence distribués [ACG$^+$06], ou encore des problèmes de satisfaction de contraintes (CSP) [PF05] ou d'optimisation de contraintes (COP)[LF11] distribués. Par exemple, nous distinguons certains systèmes d'inférence distribués qui peuvent sembler très proches de notre approche comme Somewhere [ACG$^+$06] ou plus récemment Deca [AGR09]. En effet, par leur recherche d'impliqués, ces systèmes seraient en mesure de calculer des diagnostics abductifs du système distribué global. Intuitivement, les diagnostics abductifs sont des explications impliquant les observations, donc directement impliquées par la négation de celles-ci. Ils représentent un sous-ensemble des diagnostics cohérents auxquels nous nous intéressons. De plus, si les travaux autour de Somewhere s'intéressent aussi au respect des accointances des pairs ainsi qu'à la préservation de la confidentialité des variables locales, comme c'est fréquemment le cas des systèmes supervisés, on peut considérer que l'ensemble des réponses est retourné à un pair, ce qui pourrait être assimilé à une connaissance globale du problème initial.

Nous nous intéressons à pouvoir expliquer le comportement global d'un système distribué par un ensemble de pairs quand bien même les diagnostics locaux calculés par chaque pair doivent rester confidentiels à ce pair.

En effet, on peut très bien imaginer qu'un service web d'une banque, diagnostiqué défaillant par la banque, puisse être réparé localement sans que la banque n'ait particulièrement envie de le faire savoir. Bien entendu, il faut s'attendre à ce que les réponses soient réparties à travers le système. Notons que nous serons tout de même capable de calculer les meilleurs diagnostics globaux, lorsque les politiques d'accès le permettront. Il faudra faire au mieux étant donnée

la politique d'accès, et repenser les approches envisagées dans un cadre centralisé ou même supervisé afin de pouvoir prendre en compte cette politique d'accès.

Dans le chapitre 3 nous formalisons la problématique de diagnostic distribué respectant une politique de contrôle d'accès par l'intermédiaire d'un système de pairs diagnostiqueurs. Un système de pairs diagnostiqueurs se décrit par un réseau d'accointances définissant le voisinage local de chaque pair, une politique d'accès respectée par chacun des pairs lors de ses échanges de messages et enfin un ensemble de descriptions locales traduisant, pour chaque pair, l'écart entre le comportement attendu et observé du sous-système supervisé. À partir des descriptions initiales, le but du diagnostic distribué est de pouvoir faire évoluer, par un raisonnement distribué et des échanges de messages, les descriptions locales de chacun des pairs vers des descriptions dites *accessibles* (c-à-d respectant la politique de contrôle d'accès de chacun des pairs). À partir des descriptions accessibles locales, chaque pair devra élaborer des diagnostics locaux faisant partie d'un diagnostic global du système. Cette propriété sera centrale dans notre manuscrit. Nous dirons d'un système qu'il garantit un diagnostic distribué *correct* ssi tout diagnostic local à un pair pourra se prolonger par un diagnostic global. De plus, nous dirons d'un système garantissant un diagnostic distribué correct qu'il est *compilé pour les diagnostics minimaux* ssi il est possible de vérifier (efficacement) que tout diagnostic local appartient à un diagnostic global minimal du système.

Une fois notre problématique ainsi formellement définie, nous cherchons à caractériser, chapitre 4, les propriétés structurelles des systèmes garantissant un diagnostic distribué correct. Dans ce contexte nous montrons que la propriété d'*intersection courante* aussi connue sous le nom de *running intersection*, bien connue en CSP, joue un rôle majeur. Intuitivement, la propriété de running intersection vérifie que tout ensemble de pairs contenant une même variable forme un graphe connecté. Ainsi, nous montrons que tout système appelé *structuré* (défini comme un système contenant un arbre couvrant respectant la running intersection) suffit à garantir un diagnostic distribué correct. Inversement, nous montrons aussi que dans un grand nombre de cas, correspondant aux modélisations graphiques des systèmes, seuls les systèmes structurés sont capables de garantir un diagnostic distribué correct.

Dans le chapitre 5, essentiellement basé sur l'article [ASD12], nous cherchons à décomposer un système initial vers un système structuré respectant la politique de confidentialité des des variables locales. La portée de ce chapitre est plus générale que la seule résolution de la problématique de diagnostic. Les méthodes et protocoles de décomposition arborescente que nous mettons en avant s'inscrivent dans le cadre de techniques "diviser pour mieux régner" et peuvent s'appliquer à l'optimisation de nombreux problèmes t.q. la théorie des graphes [RS86], la Bioinformatique [RPBD12], l'optimisation de contraintes [KDLD05, Dec06, JNT07], la planification [GG02], les bases de données [GLS00, DGG$^+$08], ou encore la représentation des connaissances [HD07, SBH07, FdGJ09]. La qualité d'une décomposition arborescente est donnée par la largeur arborescente du plus grand de ses clusters. Cette mesure est très importante en pratique et en théorie, car elle permet de borner la complexité maximale en temps ou en espace de toute une famille de raisonnements sur le problème décomposé. Toutefois, nous constatons que les méthodes et définitions de décomposition arborescente envisagées dans un contexte centralisé ne s'appliquent pas directement à notre contexte distribué où il est nécessaire de respecter à la fois la structure du réseau initial ainsi que la confidentialité des variables locales. Afin de respecter ces spécificités nous introduisons la notion de *décomposition arborescente distribuée*. Nous présentons tout d'abord des algorithmes de décomposition arborescente envisagés dans un contexte distribué qui se fondent sur un parcours DFS (en profondeur d'abord), BFS (en largeur d'abord) ou encore Flooding (par inondation) du réseau. Ensuite, dans le but de diminuer la largeur arborescente des décompositions arborescentes obtenues et d'accélérer le calcul des diagnostics

distribués dans le réseau, nous nous inspirons des méthodes de décomposition arborescente se fondant sur une élimination successive de variables [Dec99, KDLD05]. Nous présentons notre algorithme de décomposition arborescente distribuée, appelé *Token Elimination* dans ce même chapitre. Cet algorithme surpasse les méthodes de décomposition arborescente distribuée tout en respectant à la fois les accointances des pairs et la politique de contrôle d'accès. Le résultat de cet algorithme est un système structuré à partir duquel les pairs seront en mesure d'expliquer le comportement global d'un système distribué tout en respectant la confidentialité de leurs diagnostics locaux.

Le dernier chapitre de ce manuscrit, chapitre 6, fondé en grande partie sur l'article [ADS08], permet de relier différents résultats obtenus dans les chapitres précédents. Ainsi, dans le chapitre 4, nous avons vu que les systèmes structurés garantissent un diagnostic distribué correct. Puis, chapitre 5, nous avons présenté un algorithme distribué permettant de transformer un système distribué en un système structuré de manière performante. Dans ce dernier chapitre, nous présentons tout d'abord un algorithme permettant de garantir un diagnostic distribué compilé à partir d'un réseau structuré de DNFs. Pour cela, nous montrons tout d'abord que toute DNF contient les diagnostic minimaux d'un système. Ensuite, nous montrons qu'à partir d'un réseau de DNFs structuré, tout pair est capable de vérifier que ses diagnostics locaux font partie d'un diagnostic global. Nous montrons enfin comment, à partir de tout système structuré, on peut obtenir efficacement tous les diagnostics ou seulement une partie d'entre eux. Ce dernier résultat permet, si besoin est, de "récolter" les diagnostics stockés localement, à l'issue de l'exécution de la Token Elimination par exemple.

Enfin, nous conclurons ce manuscrit par quelques perspectives ouvertes par ce travail.

Chapitre 2

Du diagnostic au diagnostic distribué : algorithmes, structure et challenge

Dans ce manuscrit nous nous intéressons à la problématique de diagnostic de système distribué. Nous prenons un exemple jouet dans toute la thèse qui traduit un scénario anormal produit lors d'une validation d'une commande par internet. Nous présentons les structures et algorithmes envisagés pour le diagnostic à base de cohérence de systèmes dans un cadre centralisé avant d'introduire les travaux et challenges liés au diagnostic distribué.

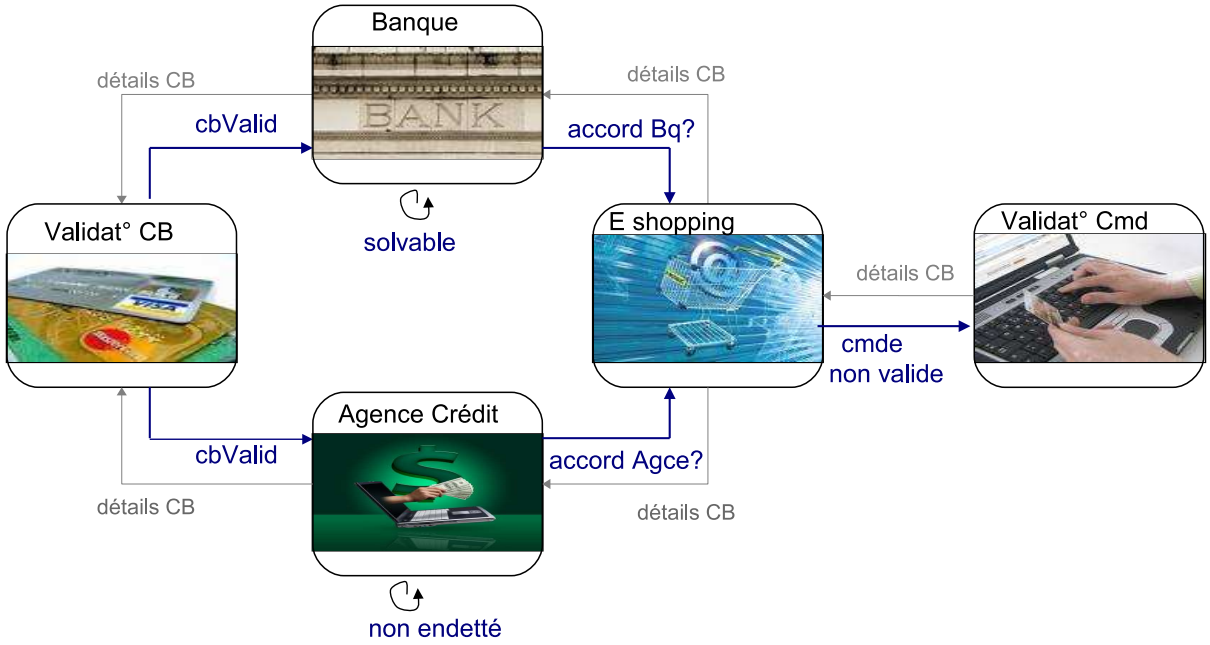


FIGURE 2.1 – Validation d'une commande par internet (exemple jouet)

Nous introduisons la problématique de diagnostic et de diagnostic distribué par le comportement de quatre services intervenant lors d'une validation d'une commande par internet (cf figure 2.1).

Exemple. 2.1 (Validation d'une commande par internet) *Un client souhaite valider une commande par internet, il envoie les détails de sa carte bancaire au service d'achat en ligne (e-shopping).*

- *Le service d'e-shopping reçoit les détails bancaires du client ainsi que sa commande et valide cette dernière si la banque du client ou une agence de crédit accepte la transaction.*
- *Le service de la banque du client reçoit une demande de transaction d'un service e-shopping et donne son accord lorsque le client est solvable et que les détails de sa carte bancaire sont valides.*
- *Le service de l'agence de crédit reçoit aussi une demande de transaction d'un service e-shopping et donne son accord lorsque le client n'est pas endetté et que les détails de sa carte bancaire sont valides.*
- *Un organisme de contrôle vérifie si oui ou non la carte bancaire est valide.*

Le scénario que l'on considère est le suivant : un client voit sa commande invalidée alors qu'il est solvable, qu'il n'est pas endetté et que sa carte bancaire est valide. Ce scénario traduit un comportement anormal des services intervenant dans la validation de commandes. Compte tenu des observations de ce scénario, le but du diagnostic est de pouvoir expliquer le ou lesquels des services n'ont pas suivi leurs comportements attendus.

Dans l'exemple précédent, une explication cohérente pourrait être : le service d'e-shopping suit un comportement anormal alors que les autres services fonctionnent normalement. En effet, si l'on suppose que le service d'e-shopping invalide toutes les commandes qu'il reçoit cela suffit à expliquer pourquoi la transaction du client a été invalidée. Dans cette thèse nous modélisons à la fois les comportements attendus et observés d'un système par la logique propositionnelle. Bien que la description d'un comportement par la logique propositionnelle peut paraître limitée, grâce aux performances sans cesse renouvelées des SAT solveurs [AS09], bon nombre de problèmes trouvent une traduction réussie (terme de rapidité de résolution) vers ce langage. Nous rappelons les principes du langage de la logique des propositions dans la section suivante. Ensuite nous présentons le cadre général du diagnostic à base de cohérence introduit par de Kleer et al [dKMR92]. Bien que ce cadre ait été largement utilisé pour la surveillance de circuits électroniques, nous verrons qu'il peut aussi s'appliquer au diagnostic de systèmes distribués. D'autre part, parmi les différents types possibles de diagnostics à base de cohérence, nous motivons notre choix de calcul des diagnostics minimaux à la fin de la section 2.1.3. Pour mieux comprendre, revenons à l'exemple. Nous remarquerons que pour expliquer le comportement anormal du système, on aurait aussi pu supposer que tous les services aient suivi un mode de fonctionnement anormal. Cette explication bien que cohérente est moins informative que celle où on suppose juste qu'un seul des service est anormal. Lorsque qu'un système est en panne, il est plus fréquent de constater que seul un petit ensemble de composants dysfonctionnent en même temps et plus rare d'observer que tous les composants suivent un comportement anormal. C'est cette préférence que traduit le calcul de diagnostic minimaux.

2.1 Le langage propositionnel pour le diagnostic à base de cohérence

2.1.1 La logique propositionnelle

Dans cette thèse nous supposons que les descriptions des comportements des différents sous-systèmes peuvent se traduire par une formule dans le langage de la logique propositionnelle. Bien que la logique propositionnelle soit un langage simple elle a la capacité de modéliser un

large éventail d'applications du monde réel. Le langage de la logique des propositions se décrit ainsi :

Définition 2.1 (Formule) *Soit un alphabet constitué de :*
- *un ensemble fini V dont les éléments sont appelés variables propositionnelles,*
- *deux symboles particuliers $\bot$ et $\top$,*
- *un ensemble C de connecteurs $\{\neg$(non), $\wedge$ (et), $\vee$ (ou), $\rightarrow$ implique), $\equiv$ (équivalent)$\}$, $\wedge$ est aussi appelé conjonction et $\vee$ est aussi appelé disjonction,*
- *deux caractères auxiliaires (et).*

Le langage des propositions $\mathcal{LP}(V, C)$ est l'ensemble des formules bien formées finies ou formules propositionnelles construites sur V et C se définit inductivement par :

1. *Une variable propositionnelle, $\bot$, $\top$ sont des formules bien formées,*
2. *Si Φ est une formule bien formée alors $\neg\Phi$ est une formule bien formée,*
3. *Si Φ et Φ' sont des formules bien formées et c un connecteur différent de $\neg$ alors $(\Phi c \Phi')$ est une formule bien formée,*
4. *Toute formule bien formée s'obtient par l'application des règles précédentes un nombre fini de fois.*

Définition 2.2 (Littéral, littéral positif, littéral négatif, littéral complémentaire) *Pour une variable propositionnelle v, v et $\neg v$ sont des littéraux. v est appelé littéral positif et $\neg v$ littéral négatif. On dira que v est le complémentaire de $\neg v$ et réciproquement.*

Définition 2.3 (Clause, monôme) *Une clause est une disjonction de littéraux, un monôme est une conjonction de littéraux.*

Pour simplifier les notations et lorsqu'il n'y aura pas d'ambiguïtés, au lieu de considérer une clause comme une disjonction de littéraux, nous la représenterons par un ensemble de littéraux E. De même nous désignerons un monôme par un ensemble de littéraux I.

Définition 2.4 (FNC, FND) *Une Forme Normale Conjonctive Σ est une conjonction de clauses. Une Forme Normale Disjonctive Π est une disjonction de monômes.*

Lorsqu'un système se décrit naturellement par un ensemble de sous-parties, c'est la forme normale conjonctive (FNC) qui est privilégiée pour modéliser le système. Dans ce cas une ou plusieurs clauses de la FNC traduit une propriété du système.

Définition 2.5 (Sémantique des connecteurs) *$\neg$ est un connecteur unaire et les autres connecteurs sont des connecteurs binaires. À $\neg$ est associé $s_\neg$ une application de $\{Vrai, Faux\}$ dans $\{Vrai, Faux\}$ et à chaque connecteur binaire c est associé s_c une application de $\{Vrai, Faux\}^2$ dans $\{Vrai, Faux\}$. $Vrai$ et $Faux$ désignent les valeurs de vérité. Le tableau suivant décrit la sémantique usuelle de chacune des fonctions s*

x	y	$s_\vee(x,y)$	$s_\wedge(x,y)$	$s_\rightarrow(x,y)$	$s_\equiv(x,y)$	$s_\neg(y)$
Vrai	*Vrai*	*Vrai*	*Vrai*	*Vrai*	*Vrai*	*Faux*
Vrai	*Faux*	*Vrai*	*Faux*	*Faux*	*Faux*	*Vrai*
Faux	*Vrai*	*Vrai*	*Faux*	*Vrai*	*Faux*	
Faux	*Faux*	*Faux*	*Faux*	*Vrai*	*Vrai*	

Définition 2.6 (Interprétation) *Une interprétation i d'une formule Φ formée sur V est une application de V dans $\{Vrai, Faux\}$. L'ensemble de toutes les interprétations sera noté $\mathcal{I}(V)$.*

Définition 2.7 (Valeur de vérité) *La fonction de valeur de vérité $\mathcal{V}$ est une fonction de $\mathcal{I}(V) \times \mathcal{LP}(V, \mathcal{C})$ dans $\{Vrai, Faux\}$ définie par :*

- $\mathcal{V}(i, \bot) = Faux$
- $\mathcal{V}(i, \top) = Vrai$
- *pour un symbole propositionnel v , $\mathcal{V}(i, v) = i(v)$*
- *pour une formule Φ, $\mathcal{V}(i, \neg\Phi) = s_\neg(\mathcal{V}(i, \Phi))$*
- *pour les formules Φ, Φ', et un connecteur c, $\mathcal{V}(i, \Phi c \Phi') = s_c(\mathcal{V}(i, \Phi), \mathcal{V}(i, \Phi'))$*

Définition 2.8 (Modèle, formule satisfaite, satisfiable, tautologie) *Pour une formule Φ et une interprétation i*

- *Φ est dite satisfaite par i si $\mathcal{V}(i, \Phi) = Vrai$, i sera appelé modèle de Φ et noté m, nous notons $\Gamma(\Phi)$ l'ensemble des modèles m de Φ.*
- *un modèle est dit partiel s'il représente une application de $V' \subseteq V$ dans $\{Vrai, Faux\}$ pouvant être prolongée par un modèle.*
- *Φ est dite falsifiée par i si $\mathcal{V}(i, \Phi) = Faux$, i est appelé contre modèle de Φ,*
- *Φ est dite satisfiable ssi elle admet au moins un modèle, elle est dite insatisfiable sinon.*
- *Φ est une tautologie ssi toutes ses interprétations sont des modèles.*

Définition 2.9 (Conséquence logique) *Soient deux formules Φ et Φ', Φ a pour conséquence logique Φ' noté $\Phi \models \Phi'$ ssi tout modèle de Φ est un modèle de Φ'*

Définition 2.10 (Impliquant, impliquant premier) *Soit Φ une formule propositionnelle et I un monôme ne contenant pas de littéraux complémentaires, I est un impliquant de Φ ssi $I \models \Phi$. I est dit impliquant premier ssi pour tout impliquant I' de Φ si $I \models I'$ alors $I \equiv I'$.*

Définition 2.11 (Impliqué, impliqué premier) *Soit Φ une formule propositionnelle et E une clause ne contenant pas de littéraux complémentaires, E est un impliqué de Φ ssi $\Phi \models E$. E est dit impliqué premier ssi pour tout impliqué E' de Φ si $E' \models E$ alors $E' \equiv E$.*

Les définitions ci-dessus sont importantes dans la mesure où la problématique de diagnostic peut se ramener à la recherche de modèles, d'impliquants ou d'impliqués d'une formule propositionnelle.

2.1.2 Décrire et analyser un système de validation de commande

Nous avons introduit la problématique de diagnostic à travers un scénario mettant en évidence un défaut de fonctionnement d'un processus de validation de commande internet. Nous modélisons tout d'abord le comportement attendu du système de validation de commande par une formule propositionnelle. Les travaux de Reiter [Rei87] et de de Kleer, Mackworth, Reiter [dKMR92] sur le diagnostic à base de modèles distinguent parmi les variables V d'un système, les variables observables V_{Obs} et les variables de modes V_{Mod}.

Les variables observables traduisent le comportement du système tel qu'il a été mesuré. Les variables de modes, représentées par okC_i, traduisent d'éventuelles pannes ou comportements critiques permettant d'identifier un comportement anormal du composant C_i. Ces variables ne sont pas observables et doivent être inférées lors du raisonnement. La description DS du comportement attendu du système est encodée par une formule propositionnelle. C'est ce formalisme que nous allons suivre pour encoder le comportement d'un système distribué. Nous donnons une définition formelle d'un système (cf : Définition 2.12).

Exemple. 2.2 (Description du système de validation de commandes) *La description du comportement attendu du processus de validation de commande par internet : DS_{Vci}, est obtenue par l'union des descriptions des services de la banque DS_{Bq}, de l'agence de crédit DS_{Ag}, d'e-shopping DS_{Es}, de validation de la carte bleu DS_{Vcb} t.q. $DS_{Vci} = DS_{Bq} \cup DS_{Ag} \cup DS_{Es} \cup DS_{Vcb}$. Dans chaque description, est présente une variable de mode okX permettant d'identifier si le service X a éventuellement suivi un comportement anormal.*

Le service de la banque *Le service de la banque est dans un état normal (okBq) lorsqu'il donne son accord (b) pour une transaction d'un client solvable (s) ayant sa carte bleue valide (v). La description propositionnelle du service de la banque se traduit par :*

$$DS_{Bq} : okBq \Rightarrow ((s \wedge v) \Leftrightarrow b)$$

L'agence de crédit *Le service de l'agence de crédit se comporte dans un état normal (okAg) lorsqu'il donne son accord (a) pour un client qui n'est pas endetté ($\neg e$) dont la carte bleue est valide (v). La description propositionnelle du service de l'agence de crédit se traduit par :*

$$DS_{Ag} : okAg \Rightarrow ((v \wedge \neg e) \Leftrightarrow a)$$

Le service d'e-shopping *De façon identique, le service d'e-shopping est dans un état normal (okEs) s'il valide une commande (c) lorsqu'il a reçu l'accord de la banque (b) ou celui de l'agence de crédit (a). La description propositionnelle du service d' e-shopping se traduit par :*

$$DS_{Es} : okEs \Rightarrow ((b \vee a) \Leftrightarrow c$$

La validation de la carte bleue *Par souci de simplicité, nous supposons que ce service ne tombe pas en panne et qu'il répond simplement si oui ou non les détails d'une carte bleue sont valides(v).*

$$DS_{Vcb} : v \vee \neg v$$

Nous venons de voir la modélisation du comportement attendu du processus de validation de commande en logique propositionnelle. Le comportement observé d'un système se traduit lui aussi par une formule. Chaque observation est traduite par l'assignation de variables. L'ensemble des observations est traduit par une conjonction de littéraux. Nous donnons l'exemple d'un comportement observé du processus de validation de commande par le scénario 1.

Exemple. 2.3 (scénario 1 commande invalidée) *Considérons le scénario 1 : un client voit sa commande invalidée alors qu'il est solvable, il n'est pas endetté et sa carte bancaire est valide. Les variables observables V_{OBS_1} sont telles que $V_{OBS_1} = \{s, e, v, c\}$. Le scénario 1 se traduit par des observations prenant la forme d'assignations $OBS_1 = \{s = Vrai,\ e = False,\ v = Vrai,\ c = False\}$. De même, ces observations satisfont le monôme (conjonction de littéraux) : $s \wedge \neg e \wedge s \wedge \neg c$.*

On notera que dans l'Exemple 2.2 la description DS_{Bq} est semblable à celle d'une simple porte logique ET. Les variables client solvable (s) et carte bleue valide (v) sont en entrées de la porte ET, l'accord de la banque (b) représente la sortie. De façon similaire, on peut associer la description du service de l'agence de crédit à une porte ET et celui d'e-shopping à une porte OU. Nous schématisons sous forme de circuit électronique la description du système de validation de commande internet Figure 2.2. Les observations OBS_1 du scénario 1 sont représentées par des

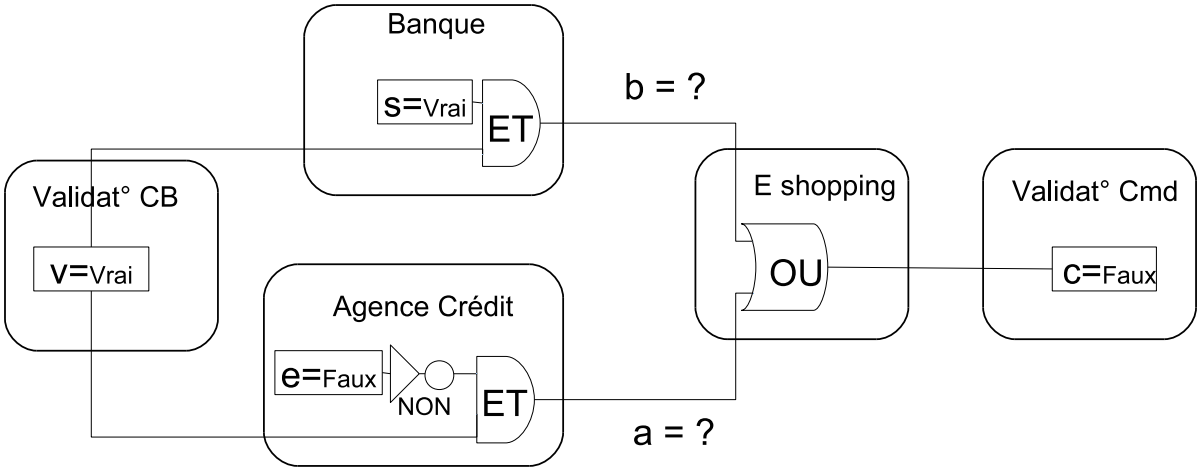


FIGURE 2.2 – Circuit électronique modélisant le processus de validation par internet

assignation de valeurs aux variables observables. Les variables non observées sont assignées de "?".

Étant donnée la description du processus de validation de commande, si l'on suppose que tous les services fonctionnent normalement ($okBq = Vrai$, $okEs = Vrai$, $okAg = Vrai$), on s'attend à ce qu'un client solvable, non endetté et ayant une carte bleue valide puisse valider sa commande. Or, par le scénario 1, un client solvable, non endetté et ayant une carte bleue valide a vu sa commande invalidée. Ainsi, supposer que tous les services suivent leurs comportement attendu n'est pas une hypothèse correcte. En logique propositionnelle, cela se traduit par une incohérence entre les hypothèses de fonctionnement des services et la description du système observé. Dans notre cas on a : $okBq \wedge okEs \wedge okAg \wedge DS_{Vci} \wedge OBS_1 \models \bot$. Intuitivement, le but du diagnostic à base de cohérence est de trouver une une explication (c-à-d une hypothèse de fonctionnement des services), cohérente avec le comportement du système observé. Nous remarquerons que pour les hypothèses de mode de fonctionnement *ok* ne figure pas dans le circuit, elles sont ajoutées à la description du système afin de détecter les comportements anormaux.

2.1.3 Le diagnostic à base de cohérence

Les travaux pionniers de Mackworth et al [dKMR92] définissent différents types de diagnostics à base de cohérence. Dans notre étude nous nous intéressons plus particulièrement au diagnostics minimaux. Ici nous présentons quelques types de diagnostics les plus utilisés et motivons notre choix.

Bien que la portée du diagnostic à base cohérence s'étend à différents domaines, les travaux de de Kleer et al ont été essentiellement appliqués à la détection de pannes dans les circuits électroniques.

Définition 2.12 (Système observé [dKMR92]) *Un système observé est alors défini comme un triplet $\langle DS,COMPS,OBS\rangle$ où*

- *DS est une formule de la logique du premier ordre décrivant le comportement du système*
- *OBS est une formule décrivant les observations (revenant dans la plupart des cas à une assignation de valeurs aux variables observables)*
- *$COMPS$ est l'ensemble des composants surveillés. Chacun de ces composants apparaît en tant qu'argument de $\neg ok()$ dans DS (ex : $\neg ok(C_i)$ signifie que C_i est anormal).*

Nous notons par Mod^- l'ensemble des fautes $\neg ok(C_i)$. Un diagnostic à base de cohérence est une explication, par des hypothèses de fonctionnement de composants, d'un écart entre le comportement attendu et le comportement observé d'un système. En logique propositionnelle, cette explication prend la forme d'une conjonction de littéraux de modes cohérente avec la description du système observé. Elle traduit une hypothèse sur le mode de fonctionnement de tous les composants du système.

Définition 2.13 (Diagnostic cohérent [dKMR92]) *Soit un système observé* $\langle DS, COMPS, OBS\rangle$ *et* $Mod^- = \{\neg ok(C_i) : C_i \in COMPS\}$ *les propositions représentant les comportements anormaux. Soit* $\Delta \in Mod^-$,

$(\Delta, \overline{\{Mod^- \setminus \Delta\}})$ *est un diagnostic cohérent ssi* $\Delta \wedge \overline{\{Mod^- \setminus \Delta\}} \wedge DS \wedge OBS \not\models \bot$

Un diagnostic cohérent est une explication cohérente du comportement attendu d'un système, avec ses observations. Nous donnons l'exemple d'un raisonnement conduisant à un diagnostic cohérent expliquant l'écart entre le comportement attendu du processus de validation de commandes par internet et le scénario 1. Nous illustrons ce diagnostic par la Figure : 2.3

Exemple. 2.4 (diagnostic cohérent du processus de validation de commande) *Nous avons vu précédemment que l'hypothèse supposant que tous les services se comportaient normalement n'expliquait pas le comportement observé du service de validation de commande. Supposons les modes de fonctionnement suivants : le service de la banque et de l'agence fonctionnent normalement (okBq, okAg) alors que le service d' e-shopping lui est anormal (¬okEs).*

1. *Si le service de la banque fonctionne normalement (okBq), étant comparable à une porte ET et ayant observé en entrée : la carte bleue est valide ($v = Vrai$) et le client est solvable ($s = Vrai$) on s'attend à ce qu'il donne son accord ($b = Vrai$).*
2. *De même si le comportement de l'agence de crédit fonctionne normalement (okAg), étant comparable à la composition d'une porte NON et d'une porte ET, ayant observé que le client n'est pas endetté ($e = Faux$) et sa carte bleue est valide ($v = Vrai$) on s'attend à ce que le service donne son accord (a).*
3. *Par contre si l'on suppose que le service d'e-shopping est anormal ¬okEs, on ne s'attend plus à ce qu'il se comporte comme une porte OU comme il était prévu. Il peut donc invalider la commande ($c = Faux$) alors qu'il a peut être reçu des avis positifs de la banque (b) et de l'agence de crédit (a).*

Ainsi les hypothèses ({¬okEs}, {okBq, okAg}) forment un diagnostic cohérent avec la description du système de validation de commande et le scénario 1.

Nous remarquons que si le système observé est insatisfiable, il n'y a pas de diagnostics. Ainsi, la satisfiabilité du système observé est une condition nécessaire à la problématique de diagnostic. Cette remarque est très utile pour la génération d'instances de systèmes à diagnostiquer. D'autre part, pour simplifier les définitions suivantes, nous appelons diagnostic les hypothèses de fonctionnement anormales Δ uniquement constituées de littéraux de mode négatifs. Pour revenir à la définition originale, il suffira de calculer $\overline{Mod^- \setminus \Delta}$ en fonction de Δ.

Étant donné un système observé satisfiable, il peut y avoir au maximum $2^{|V_{Mod}|}$ diagnostics possibles. Dans un souci d'une représentation succincte on cherche soit à représenter tous les diagnostics de façon compacte, soit à conserver les plus intéressants d'entre eux. Par exemple supposer que tous les composants d'un système sont défaillants en même temps n'est pas une hypothèse réaliste. Inversement supposer que seuls quelques composants d'un système sont en panne en même temps est plus réaliste. Ainsi nous nous intéresserons aux diagnostics faisant l'hypothèse d'un nombre ou d'un ensemble minimal de composants défaillants.

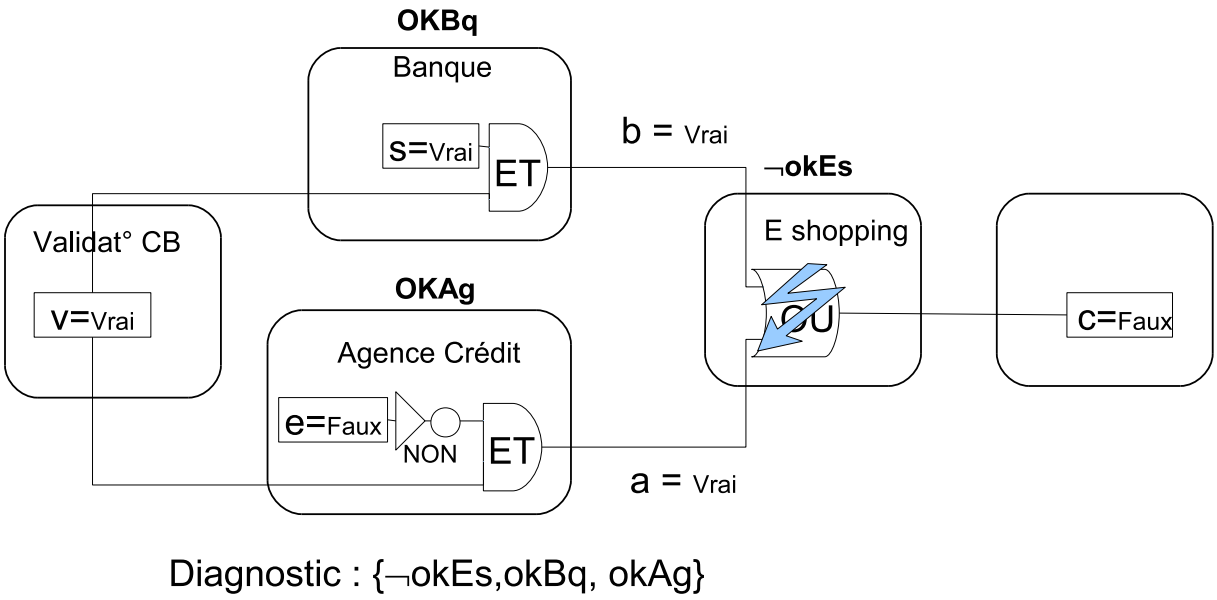


FIGURE 2.3 – Diagnostic du processus de validation d'une commande

Définition 2.14 (Diagnostic minimal) *Étant donné un système observé satisfiable* $\langle DS, COMPS, OBS\rangle$ *et un ensemble de littéraux de mode négatifs :* $Mod^- = \{\neg okC_i : C_i \in COMPS\}$*, un diagnostic cohérent* $\Delta \subseteq Mod^-$ *est un diagnostic cohérent minimal ssi aucun sous-ensemble propre* Δ' *t.q.* $\Delta' \subset \Delta$*, est un diagnostic cohérent.*

Exemple. 2.5 (Diagnostics minimaux du processus de validation de commande) *Dans un premier temps, nous avons vu que l'on ne pouvait pas supposer que tous les services se comportent normalement, ainsi* $\Delta = \{\}$ *n'est pas un diagnostic. D'autre part nous avons vu que* $\{\neg okEs\}$ *était un diagnostic. Puisqu'il n'existe pas de sous-ensemble de* $\{\neg okEs\}$ *qui soit un diagnostic,* $\{\neg okEs\}$ *est un diagnostic minimal. De même on peut vérifier que l'autre diagnostic minimal est* $\{\neg okBq, \neg okAg\}$*.*

La façon classique de calculer les diagnostics minimaux consiste, dans un premier temps, à détecter les conflits. Un conflit peut être vu comme un symptôme déclencheur de la recherche de diagnostic. Intuitivement, un conflit est un ensemble d'hypothèses de bon fonctionnement du système qui ne peuvent être toutes supposées vraies à la fois. Plus formellement :

Définition 2.15 (Conflit, conflit minimal, conflit positif) *Étant donné un système observé satisfiable* $\langle DS, COMPS, OBS\rangle$ *et un ensemble de variables de mode* $Mod = \{okC_i : C_i \in COMPS\}$*,*

- *un conflit est une clause* E *formée sur les variables de modes t.q.* $DS \wedge OBS \models E$
- *un conflit* E *est minimal s'il n'existe pas de conflit* E' *t.q.* $E' \models E$*.*
- *Un conflit est dit positif (sous-entendu en termes de littéraux d'anormalités, ici* $\neg okC_i$*) s'il est une disjonction de littéraux de* Mod^-*.*

Un conflit est une conséquence de l'écart entre le comportement attendu et le comportement observé du système. C'est un impliqué de $DS \wedge OBS$. Un conflit minimal est un impliqué premier de $DS \wedge OBS$.

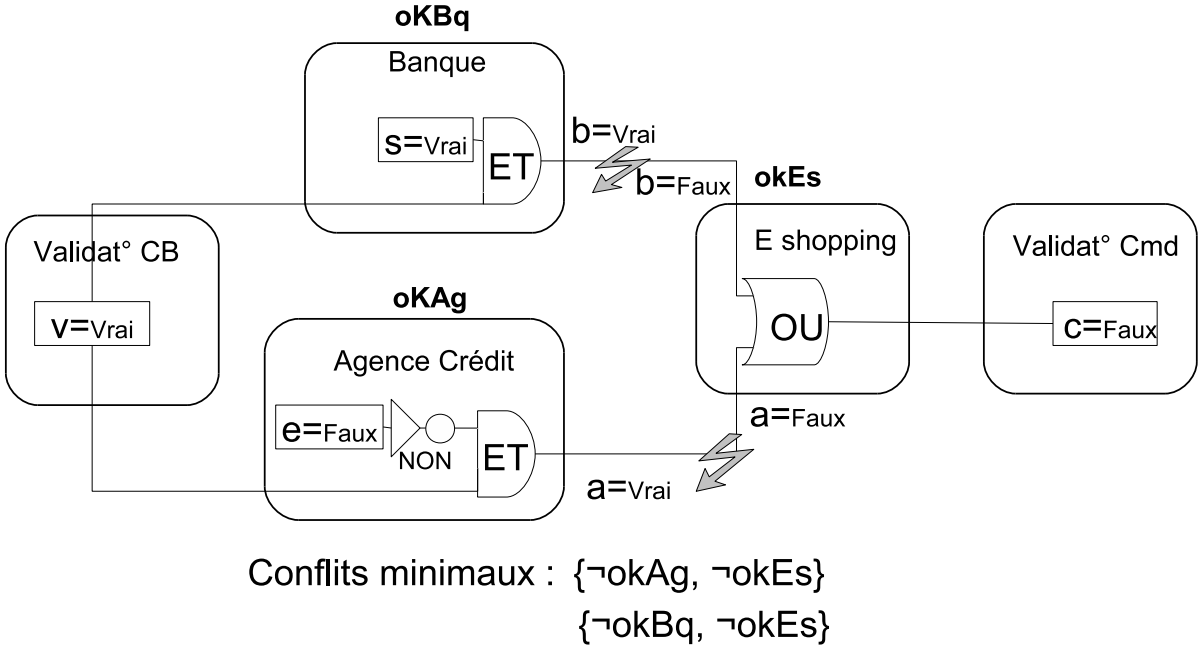


FIGURE 2.4 – Conflits du processus de validation d'une commande

Dans l'exemple suivant illustré Figure 2.4 nous expliquons un raisonnement permettant d'inférer de nouveaux conflits à partir de la description observée de notre exemple jouet. Le raisonnement que nous décrivons reprend les principes de l'ATMS [dK86] (système de maintien de vérité par hypothèses).

Exemple. 2.6 (Conflits minimaux du processus de validation de commande) *Nous avons vu que compte tenu du scénario 1 nous ne pouvions pas supposer que tous les services fonctionnent correctement, ainsi* $\neg okAG \vee \neg okEs \vee \neg okBq$ *est un conflit.*

Supposons que le service de la banque $okBq$ *et celui d'e-shopping* $okEs$ *fonctionnent correctement. Étant donné que le service de la banque se comporte comme une porte ET, en observant que le client est solvable* $(s = Vrai)$ *et que sa carte bleue est valide* $(v = Vrai)$ *en entrées, on s'attend à ce que la banque donne son accord* $(b = Vrai)$*. De même, étant donné que le service d'e-shopping se comporte comme une porte OU, en observant une commande invalide en sortie* $(c = Faux)$*, on s'attend à ce que ni la banque ni l'agence de crédit n'ait donné leur accord* $(b = Faux)$*,* $(a = Faux)$*. Or là, il y a un conflit entre la valeur attendue à la sortie du service de la banque* $(b = Vrai)$ *et celle attendue à l'entrée du service d'e-shopping* $(b = Faux)$*. On peut supposer que les services de la banque et de l'e-shopping fonctionnent correctement ensemble,* $\neg okBq \vee \neg okEs$ *est un conflit minimal. Par un raisonnement similaire nous pouvons déduire que* $\neg okBq \vee \neg okEs$ *est un autre conflit minimal. Par contre on ne peut pas déduire que seul l'un de ces services ne fonctionne pas correctement.*

Le théorème suivant de de Kleer et al [dKMR92] permet de calculer tous les diagnostics minimaux d'un système à partir de ses conflits minimaux.

Théorème 2.1 (Caractérisation des diagnostics minimaux [dKMR92]) *Étant donné un système observé satisfiable* $\langle DS, COMPS, OBS\rangle$ *et* Mod^- *un ensemble de littéraux de mode négatif,* $\Delta \subseteq Mod^-$ *est un diagnostic minimal de* $(DS, COMPS, OBS)$ *ssi* Δ *est un impliquant premier de l'ensemble des conflits minimaux positifs de* $(DS, COMPS, OBS)$*.*

Nous illustrons ce théorème en revenant à notre exemple de validation de commande.

Exemple. 2.7 (Détermination des diagnostics minimaux par les conflits) *Nous avons vu précédemment que les conflits minimaux (qui, ici sont positifs) peuvent s'encoder par la formule* $\Sigma_{conflits} : (\neg okBq \vee \neg okEs) \wedge (\neg okAg \vee \neg okEs)$. *Les impliquants premiers de* $\Sigma_{conflits}$ *sont* $\neg okES$ *et* $\neg okBq \wedge \neg okAg$. *Ils sont simplement obtenus en résolvant les conflits. Ce sont aussi les diagnostics minimaux du processus de validation de commande observé par le scénario 1.*

Par le théorème précédent on comprend que si l'on désire obtenir un diagnostic minimal par le calcul préalable des conflits, avant d'obtenir le premier diagnostic minimal, il faudrait tout d'abord compiler l'ensemble des conflits minimaux. D'autre part, sans pour autant avoir l'ensemble des conflits minimaux on peut tout de même avoir des diagnostics minimaux par la remarque suivante.

Remarque 2.1 (Diagnostic minimal inclus dans un diagnostic) *Étant donné un système observé* $\langle DS, COMPS, OBS\rangle$ *et un ensemble de littéraux de mode négatifs* $Mod^- = \{\neg okC_i : C_i \in COMPS\}$, *si* $\Delta \in Mod^-$ *est un diagnostic, alors il existe un diagnostic minimal* $\Delta' \subseteq \Delta$.

Ainsi, lors de la recherche des diagnostics, réduire successivement la taille d'un diagnostic quelconque en inversant une hypothèse de panne par une hypothèse de bon fonctionnement tout en restant cohérent, conduit à la formation de diagnostics minimaux. Inversement, nous noterons que tout sur-ensemble d'un diagnostic minimal n'est pas forcément un diagnostic cohérent.

Exemple. 2.8 *Supposons que l'on ajoute une règle d'exclusion entre les modèle de bon fonctionnement du service d'e-shopping et de la banque :* $(\neg okBq \Rightarrow okEs)$, *autrement dit, le service d'e-shopping et le service de la banque ne peuvent être tous les deux défaillants en même temps.*

Avec cette nouvelle règle, on constate que $\Delta_{all} = \{\neg okBq, \neg okEs, \neg okAg\}$ *n'est plus une conjonction cohérente et donc n'est plus un diagnostic. Par contre* $\Delta_{Es} = \{\neg okEs\}$ *et* $\Delta_{Ag,Bq} = \{\neg okBq, \neg okAg\}$ *qui sont des sous-ensembles de* Δ_{all}, *restent des diagnostics.*

Par cette dernière remarque nous constatons que les diagnostics minimaux ne caractérisent pas l'ensemble des diagnostic cohérents d'un système observé. En effet, tout prolongement d'un diagnostic minimal par un littéral de mode négatif, ne conduit pas forcément à la construction d'un diagnostic cohérent.

Dans le but d'avoir à la fois les diagnostics intéressants et caractéristiques de l'ensemble de tous diagnostics de même type, de Kleer et al ont introduit les diagnostics partiels et noyaux.

Définition 2.16 (Diagnostics partiels et diagnostics noyaux) *Étant donné un système observé* $\langle DS, COMPS, OBS\rangle$ *et un ensemble de littéraux de mode négatifs* $Mod^- = \{\neg okC_i : C_i \in COMPS\}$,

- $\Delta \subseteq Mod^- \cup \overline{Mod^-}$ *est un diagnostic partiel ssi toute extension de* Δ *(par des littéraux de modes positifs et négatifs) à tout* $COMPS$ *est un diagnostic cohérent.* Δ *est un diagnostic et pour tout sur-ensemble* Δ' *de* Δ, Δ' *est un diagnostic cohérent.*
- $\Delta \subseteq Mod^- \cup \overline{Mod^-}$ *est un diagnostic noyau ssi* Δ *est un diagnostic partiel et il n'existe pas de sous-ensemble* $\Delta' \subset \Delta$ *t.q.* Δ' *soit un diagnostic partiel.*

Implicitement, on comprend que tout prolongement d'un diagnostic partiel par un littéral de mode positif ou négatif est aussi un diagnostic partiel. Étant donné qu'un diagnostic noyau est un diagnostic partiel minimal, l'ensemble des diagnostics noyaux caractérisent tous les diagnostics partiels du système observé et ainsi tous les diagnostics.

Toujours dans la volonté d'avoir une représentation de l'ensemble des diagnostics significatifs d'un système, les travaux de Reiter [Rei87] caractérisent à travers les diagnostics abductifs les explications cohérentes d'un système observé qui sont directement impliquées (logiquement) par les observations. Un diagnostic abductif est plus précis qu'un diagnostic cohérent dans la mesure où ce dernier se contente d'être cohérent avec les observations et n'est pas nécessairement impliqué par les observations.

Définition 2.17 (Diagnostic abductif) *Étant donné un système observé* $\langle DS, COMPS, OBS\rangle$ *et un ensemble de littéraux de mode négatifs* $Mod^- = \{\neg okC_i : C_i \in COMPS\}$, $\Delta \subseteq Mod^-$ *est un diagnostic abductif du système observé ssi* $\Delta \wedge DS \not\models \bot$ *et* $\Delta \wedge DS \models OBS$

Un diagnostic cohérent est une assignation de variables de modes cohérente avec la description du système observé. Dans le cas général, la recherche de diagnostics cohérents peut s'apparenter à le recherche des modèles de la formule décrivant le système observé. Dans le cas des diagnostics abductifs, Inoue [Ino92] a montré que les diagnostics abductifs correspondaient à la négation de formules impliquées par le système muni de la négation des observations (c-à-d $\Delta \wedge DS \models OBS$ implique $\wedge DS \wedge \neg OBS \models \neg\Delta$). Cette remarque permet à tout système d'inférence de pouvoir compiler les diagnostics abductifs par un ensemble d'impliqués, conséquence du système observé. Dans un contexte distribué, un système d'inférence distribué comme Somewhere [ACG+06]serait en mesure de retourner les diagnostics abductifs minimaux.

Malheureusement, nous avons la remarque suivante :

Remarque 2.2 (Diagnostic abductif vide, Diagnostic minimal non vide) *Il peut arriver que l'ensemble des diagnostics abductifs soit vide alors que l'ensemble des diagnostics (et donc des diagnostics minimaux) n'est pas vide*

Ainsi, la détection de fautes par des diagnostics abductifs est parfois impossible ou nécessite des restrictions fortes sur les descriptions de systèmes à surveiller.

Exemple. 2.9 *Reprenons otre exemple jouet, ce système observé ne contient pas de diagnostics abductifs, alors même que le scénario 1 est un scénario anormal. Si par exemple, en plus du scénario 1, on observait que la banque avait donné son accord pour la transaction (c-à-d* $b = Vrai$*) alors* $\neg okAg \wedge \neg okEs$ *serait un diagnostic abductif.*

Lorsque le nombre de diagnostics d'un système peut être prohibitif, certaines approches s'intéressent à la compilation des diagnostics de cardinalité minimale.

Définition 2.18 (diagnostic de cardinalité minimale) *Un diagnostic de cardinalité minimale est un diagnostic cohérent t.q. pour tout autre diagnostic* Δ', $|\Delta| \leqslant |\Delta'|$

On notera qu'un diagnostic est minimal pour l'inclusion ensembliste alors qu'un diagnostic de cardinalité minimale est minimal pour le nombre de symboles de fautes qu'il contient. Nous soulignons que les diagnostics de cardinalité minimale font l'hypothèse que tous les diagnostics sont comparables entre eux, même ceux n'ayant aucun littéral de mode en commun. Les diagnostics minimaux pour l'inclusion ensembliste ne font pas cette hypothèse. C'est le cas de l'exemple 2.10 où le diagnostic Δ_{Es} est comparé à $\Delta_{Bq,Ag}$.

Exemple. 2.10 (diagnostic de cardinalité minimale du scénario 1) *Nous avons mis en évidence deux diagnostics minimaux :* $\Delta_{Es} = \{\neg okEs\}$ *et* $\Delta_{Bq,Ag} = \{\neg okBq, \neg okAg\}$ *pour expliquer le scénario 1. On a* $|\Delta_{Es}| < |\Delta_{Bq,Ag}|$. *Puisque l'ensemble vide n'est pas un diagnostic pour le scénario 1,* Δ_{Es} *est le seul diagnostic de cardinalité minimale.*

On remarquera que bien que les diagnostics de cardinalité minimal représentent un sous-ensemble des diagnostics minimaux, un diagnostic de cardinalité minimale vide équivaut à un diagnostic minimal vide. Plus formellement :

Remarque 2.3 *Il existe un diagnostic cohérent non vide pour un système observé ssi il existe un diagnostic de cardinalité minimale supérieure ou égale à 1.*

Synthèse sur la caractérisation des diagnostics

En résumé, un diagnostic est une explication cohérente du mode de fonctionnement des composants d'un système observé (cf def 2.13). Étant donnée une description, les diagnostics noyaux (def 2.16) caractérisent l'ensemble des diagnostics d'un système de façon compacte (tout prolongement d'un diagnostic noyau par des littéraux de mode est aussi un diagnostic). Quant aux diagnostics abductifs (def 2.17), ils ne détectent pas toujours toutes les explications cohérentes avec un système observé et peuvent conclure sur l'absence de diagnostic alors que le système suit un comportement anormal cf remarque (rq 2.2). Dans ce cas, des restrictions fortes sur la description du système sont nécessaires pour que les diagnostics abductifs représentent tous les diagnostics cohérents. Contrairement au diagnostic noyau, un diagnostic minimal n'est pas une représentation compacte des diagnostics cohérents (tout prolongement d'un diagnostic minimal par un littéral de mode n'est pas nécessairement un diagnostic cohérent). A défaut d'être compacts, les diagnostics minimaux permettent de détecter le comportement anormal de descriptions de systèmes très variées (y compris la réécriture de systèmes à évènements discrets traduits vers un langage propositionnel [GARK07]). Lorsque comparer les occurrences de pannes par leur nombre a du sens, on peut s'intéresser aux diagnostics de cardinalité minimale (cf def 2.18). Ils représentent un sous-ensemble des diagnostics minimaux. Dans notre étude nous ne considérons pas que deux diagnostics disjoints soient comparables, nous nous intéressons au contraire à calculer les diagnostics minimaux pour des descriptions de systèmes distribués propositionnalisés. Dans la suite, nous présentons différentes approches de l'état de l'art, envisagées pour la compilation des diagnostics cohérents.

2.1.4 Problèmes autour du diagnostic à base de cohérence

Le diagnostic à base de cohérence est lié à de nombreux problèmes de raisonnement en logique propositionnelle. Ici, nous faisons abstraction des différentes modélisations, et considérons en entrée de chaque problème une formule CNF Σ, souvent utilisée comme langage de description.

Le diagnostic à base de modèles

Description : Soient V un ensemble de symboles propositionnels, Σ le système observé traduit par une formule satisfiable construite sur une partition de V en $\mathcal{H}$ (les données hypothèses) et $\mathcal{NH}$ (les données non-hypothèses).

Question : Trouver tous les monômes I construits sur $\mathcal{H}$ t.q. $I \wedge \Sigma \not\models \bot$.

Étant donné un système décrit par un ensemble de propriétés prenant la forme d'une formule FNC, trouver un diagnostic est un problème NP-Difficile.Trouver l'ensemble des diagnostics est un problème #P-Difficile ; pour certaines instances, il peut y avoir un nombre exponentiel de diagnostics. Pour des raisons pratiques évoquées précédemment, on préfère calculer soit les diagnostics minimaux pour l'inclusion ensembliste soit les diagnostics de cardinalité minimale.

Le diagnostic à base de cohérence est lié au très populaire problème de satisfiabilité, certaines approches reprennent même les techniques et algorithmes de ce domaine pour accélérer la recherche de diagnostics.

SAT

Description : Soient V un ensemble de symboles propositionnels et Σ un ensemble fini de clauses construites sur V.

Question : Existe-t-il une interprétation de V qui satisfasse l'ensemble des clauses de Σ.

La plupart des algorithmes SAT prouvent la satisfiabilité de la formule par la recherche d'un modèle satisfaisant la formule. Ainsi, avec la problématique de satisfiabilité vient immédiatement la problématique de recherche de modèles. Beaucoup de problèmes, y compris le diagnostic, peuvent se formaliser en terme de recherche d'un modèle ou de tous les modèles d'une formule.

La recherche d'un modèle, de tous les modèles

Description : Soit Σ une formule construite sur l'ensemble de variables V.

Question : Trouver un modèle de Σ

Question : trouver tous les modèles de Σ.

Compiler tous les modèles d'une formule est une tâche difficile et souvent impossible en pratique pour des instances de grandes tailles. Afin d'éviter l'explosion combinatoire inhérente à la recherche des modèles, on s'intéresse aux modèles préférés. De même que les méthodes de recherche de modèles peuvent directement être réutilisées pour la recherche de diagnostic, les méthodes de recherche de modèles préférés peuvent être réutilisées pour la recherche de diagnostics minimaux.

La recherche d'un modèle préféré, de tous les modèles préférés

Description : Soient S, un ensemble de symboles propositionnels, Σ une formule construite sur S, $\mathcal{R}$ un ordre partiel ou total sur les interprétations.

Question : Si Σ est cohérente alors trouver un modèle M préféré de Σ tel que pour tout modèle M' de Σ, $M\mathcal{R}M'$

Question : Si Σ est cohérente alors trouver tous ses modèles préférés.

La première méthode envisagée pour le calcul de diagnostic se faisait par le calcul préalable des conflits [dK86]. Un conflit correspond à un impliqué de la formule encodant le système observé, un conflit minimal correspond à un impliqué premier, un diagnostic minimal correspond à un impliquant premier des conflits minimaux positifs.

Nous avons vu que les conflits correspondent à certains impliqués de la formule alors que les diagnostics correspondent à des impliquants de la formule.

Impliqués, impliqués premiers, impliquants premiers

Description : Soient S un ensemble de symboles propositionnels, Σ une formule construite sur S.

(déduction) Question : Soit E une clause, est ce que $\Sigma \models E$?

(impliqués premiers) Question : trouver l'ensemble des clauses E t.q. $\Sigma \models E$ et pour tout impliqué F ($\Sigma \models F$), alors $F \not\subset E$.

(impliquants premiers) Question : trouver l'ensemble des impliquants I t.q. $I \models \Sigma$ et pour tout impliquant I' ($I' \models \Sigma$), $I \models \Sigma$ et $I' \not\subset I$.

La question de la déduction $\Sigma \models E$? ne se borne pas à un problème de cohérence. Dans le cadre de requêtes, Σ est une base de connaissances qui est questionnée de nombreuses fois et qui évolue au cours du temps (rajout ou retraits d'informations). Ce problème induit donc d'autres problèmes

tel que la gestion de la base de connaissances afin de répondre au plus vite aux requêtes et prend place à travers un contexte plus général de compilation de connaissances. La problématique de diagnostic où les observations peuvent arriver au cours du calcul, prend en compte ce type de problèmes connexes au raisonnement. Nous détaillons dans la section suivante les différentes approches et techniques envisagées pour le diagnostic à base de cohérence de systèmes.

2.2 Les structures arborescentes au coeur des approches pour le diagnostic

En fonction des spécificités du système, selon que le système soit statique ou dynamique et selon les caractéristiques requises par l'outil de diagnostic (prise en compte des observations incrémentales, diagnostic en ligne, diagnostic hors ligne, ...) différentes approches ont été envisagées. Nous les regroupons à travers deux implémentations : les approches interprétées et les approches compilées. Comprendre la distinction entre ces approches dans un contexte centralisé nous permet de mieux comprendre les méthodes proposées dans un contexte distribué. De plus, nous avons remarqué qu'à la fois dans un contexte centralisé et dans un contexte distribué, les approches de diagnostic font une place prépondérante aux structures et calculs arborescents. Nous présentons donc comment les propriétés de ces structures améliorent à la fois la compacité de la représentation du système et la rapidité de recherche des diagnostics.

2.2.1 Décomposer pour mieux diagnostiquer

La décomposition arborescente de systèmes est une technique d'optimisation qui a permis des améliorations significatives des performances dans la résolution de problèmes difficiles dans de nombreux domaines. Cette technique fait partie des stratégies d'optimisation "diviser pour régner". Le problème initial est décomposé en un ensemble de sous-problèmes plus petits et organisés à travers un arbre de façon à diminuer le temps et l'espace nécessaire pour la résolution. Cette technique tient un rôle prépondérant pour la résolution de problèmes d'énumération auxquels s'apparente le calcul des diagnostics minimaux. Souvent on n'a guère d'autres choix que de décomposer. Elle permet de guider la recherche des approches interprétées (c-à-d des approches recherchant des diagnostics directement à partir de la description d'un système) et aussi de structurer les approches compilées (c-à-d des approches recherchant des diagnostics à partir de la réécriture d'un système vers un langage plus facilement traitable). Afin de ne pas interrompre la présentation des différents travaux liés au diagnostic, nous introduisons brièvement la décomposition arborescente. Dans les chapitres suivants nous présenterons un état de l'art plus important sur les techniques et méthodes de décomposition. Dans le but d'illustrer une première décomposition arborescente, à la Figure 2.5, nous indiquons dans chaque table les assignations de valeurs aux variables satisfaisant la description de chaque service ainsi que les observations du scénario 1 intervenant dans notre exemple introductif . Par souci de concision, nous avons remplacé le symbole $Vrai$ par 1 et le symbole $Faux$ par 0. Le symbole $*$ désigne 0 ou 1. Les tables représentent les descriptions des services par des contraintes extensionnelles.

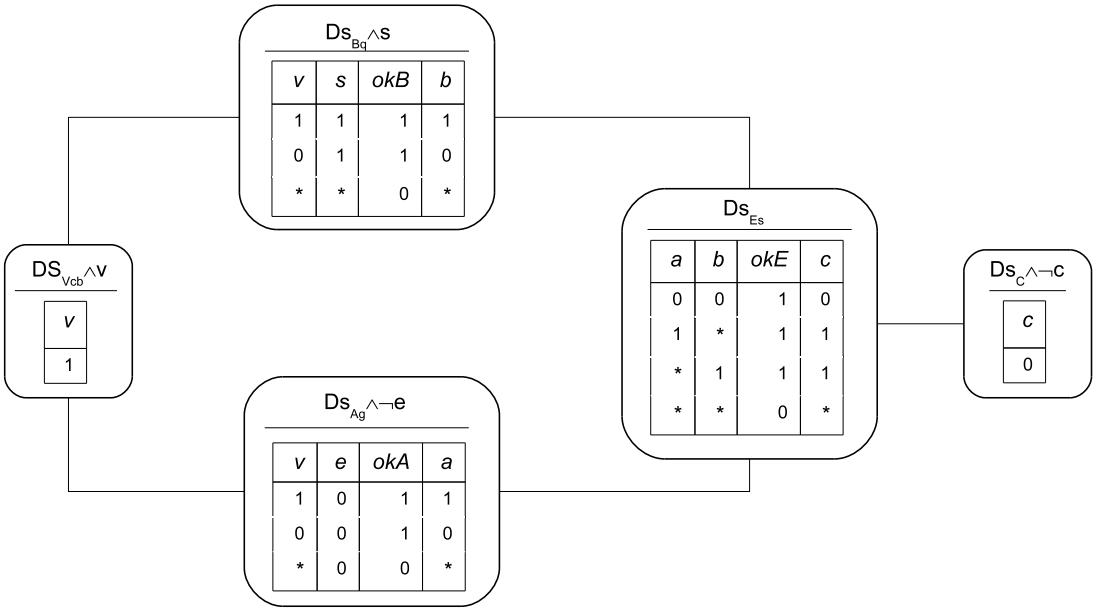


FIGURE 2.5 – Contraintes explicites décrivant le processus de validation d'une commande

Intuitivement, la décomposition arborescente découpe un problème initial vers un ensemble de clusters où chaque cluster contient un sous-problème du problème initial et l'ensemble des clusters est structuré à travers un arbre. En nous inspirant des travaux de Lask, Dechter et al [KDLD05], nous définissons la décomposition arborescente de descriptions de systèmes observés comme suit :

Définition 2.19 (Décomposition arborescente des descriptions d'un système) *Soit* $\langle DS, COMPS, OBS\rangle$ *la description d'un système observé sur un vocabulaire propositionnel* V *avec* $DS = \bigcup_{i\in COMPS} DS(i)$ *où* $DS(i)$ *est la description du composant* i *définie sur le vocabulaire* $voc(SD(i))$. *Notons* CT *l'ensemble un ensemble de clusters. Notons* Ψ *la fonction associant à chaque cluster* $ct \in CT$ *un sous-ensemble de la description du système observé initial* $\Psi(k) \subseteq \bigcup_{i\in COMPS} DS(i) \wedge OBS(i)$. *Notons* χ *la fonction associant à chaque cluster* $ct \in CT$ *un sous-ensemble des variables de* V, *correspondant au vocabulaire des descriptions qu'il couvre. Une décomposition arborescente est un triple* $\langle T, \chi, \psi\rangle$ *t.q.* $T(CT, E)$ *est un arbre où* E *représente les arrêtes et :*

1. *Pour toute description* $DS(i)$, *il existe un cluster* ct *t.q.* $DS(i) \cup OBS(i) \in \Psi(ct)$. *(Cette propriété permet de préserver les descriptions du problème initial).*
2. *Pour tout cluster* $ct \in CT$ $\bigcup_{DS(i)\in\Psi(ct)} voc(DS(i)) \subseteq \chi(ct)$.*(Cette propriété permet de préserver le vocabulaire du problème initial)*
3. *Pour toute variable* $v \in V$ *l'ensemble des clusters* $\{ct : v \in \chi(ct)\}$ *est un sous-arbre connecté de* T. *(Cette propriété est appelée running intersection ou encore connectivité du vocabulaire)*

Nous venons d'introduire une définition possible de la décomposition arborescente à partir des descriptions d'un système. D'autres décompositions arborescentes sont possibles, notamment à partir des fonctions ou encore du vocabulaire d'un système.

À partir du système Figure 2.5, nous illustrons à la Figure 2.6 la formation de clusters en vue d'une décomposition arborescente. Le cluster ct_1 regroupe les descriptions de DS_{Es} et $DS_C \wedge \neg c$. Le cluster ct_3 regroupe les descriptions de DS_{Vcb} et $DS_{Ag} \wedge \neg e$ alors que Le cluster ct_2 ne contient que la description $DS_{Bq} \wedge s$. Les clusters ainsi formés sont ensuite organisés à travers un arbre. Ainsi le cluster ct_2 est relié à ct_1, puis le cluster ct_3 est relié à ct_1.

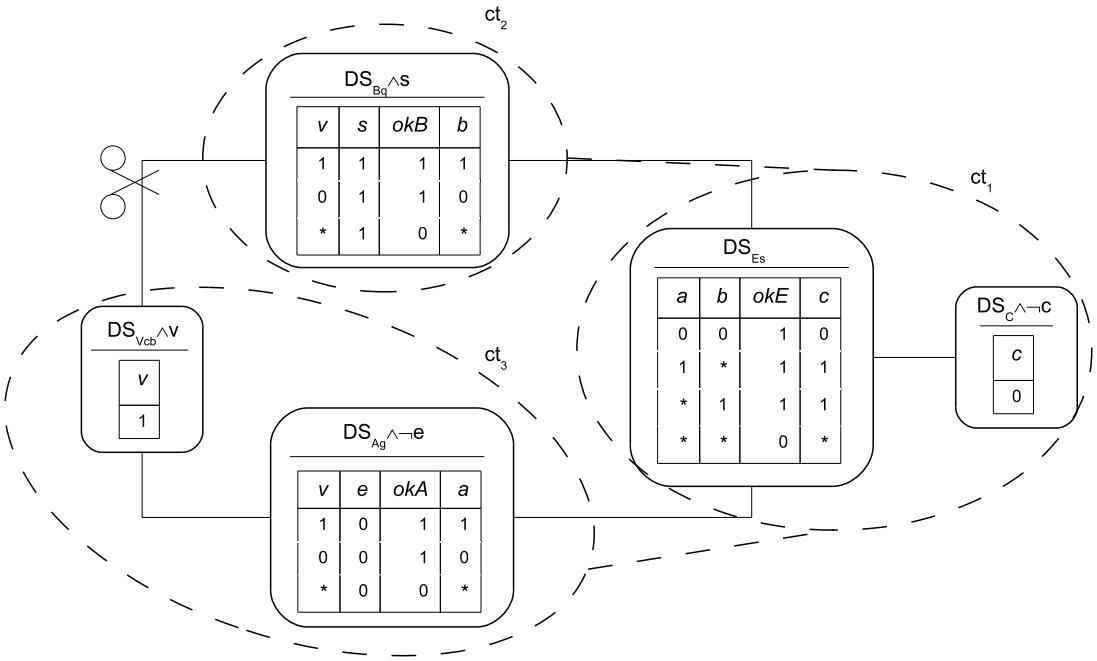


FIGURE 2.6 – Regroupement des services du processus de validation d'une commande

Afin de respecter la running intersection (cf Déf 2.19 pté 3), l'information partagée par des descriptions voisines dans le système initial (ex. la variable v entre DS_{VCB} et DS_{Bq}) mais déconnectées dans l'arbre de clusters nouvellement formé (ex. ct_2 et ct_3 ne sont pas connectés), est dupliquée à travers un chemin de l'arbre de clusters. En effet initialement le cluster ct_1 reliant ct_2 à ct_3 dans l'arbre de clusters dans la Figure 2.6 ne contient pas la variable v. La variable v ainsi que les assignations lui étant associés sont rajoutées au cluster ct_1 dans la Figure2.7. Cette figure décrit une décomposition arborescente du réseau de contraintes Figure 2.5. Nous pouvons vérifier que toutes les contraintes et le vocabulaire du système initial se retrouvent dans le système décomposé (cf propriétés 1 et 2 Déf 2.19).

D'autre part, à la Figure 2.7, nous présentons aussi la description de chacun des clusters du système décomposé sous forme de contraintes extensionnelles. Dans ce système, un simple parcours, des feuilles vers la racine, vérifiant la cohérence locale des tuples suffit à élaborer les diagnostic globaux du système. Par exemple, un premier diagnostic global $\{okA, okB, \neg okE\}$ se retrouvera à partir des tuples 1 de ct_2 et ct_3 qui sont cohérents avec le tuple 3 de ct_1 par les variables $\{a, b, v\}$. Par un parcours des feuilles vers la racine ct_1, on pourra se rendre compte qu'aucun tuple de ct_2 n'assigne v à 0. Par un parcours de la racine aux feuilles on déduira que le 2ème tuples de ct_2 ne participera à aucun diagnostic global. Nous venons de donner un exemple de ce que pourrait être une recherche de diagnostics à travers un système décomposé. Nous présentons les différentes approches envisagées pour le diagnostic dans les sections suivantes.

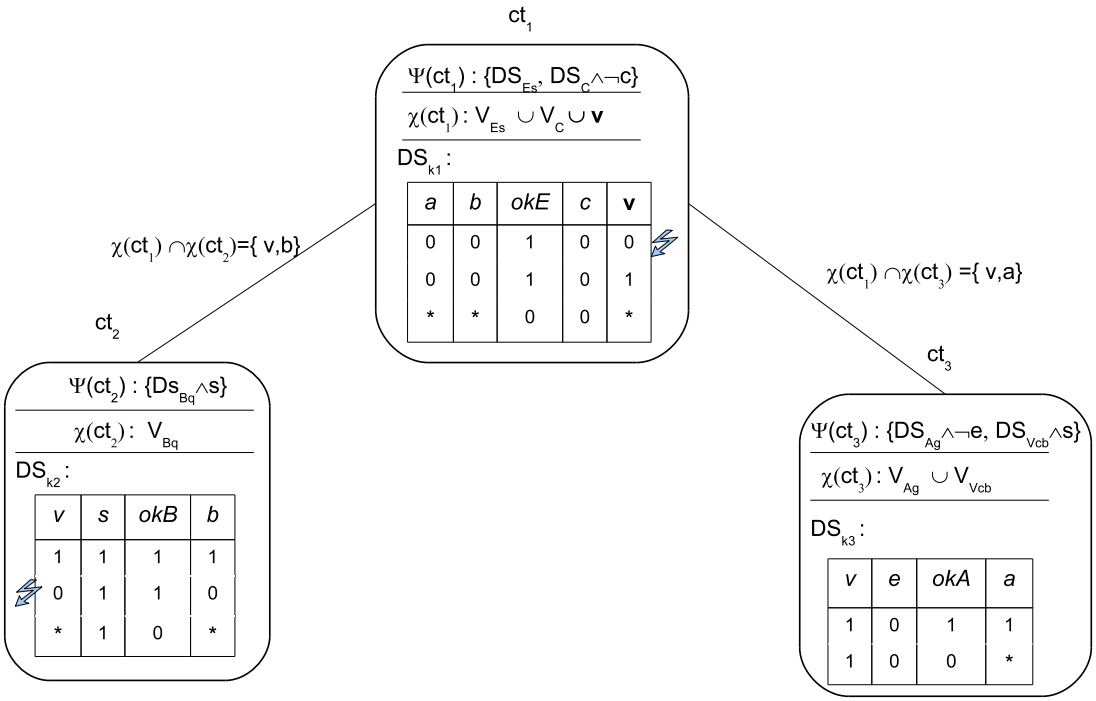


FIGURE 2.7 – Une décomposition arborescente du processus de validation d'une commande

2.2.2 Les approches interprétées ou compilées pour le diagnostic

Plusieurs approches ont été envisagées pour résoudre les problèmes du diagnostic de systèmes. Les premiers outils de diagnostic ont été implémentés par le TMS (Truth Maintenance System) de Doyle ou encore par l'ATMS (Assumption Truth Maintenance System) de de Kleer, le CMS ou le GDE (General Diagnosis Engine) de Kleer et Williams [dKW87]. Plus récemment le projet Lydia [FPvG08, lyd] propose un outil permettant de calculer entre autres les diagnostics les plus probables. Dans ses débuts, deux voies, l'approche compilée et l'approche interprétée, ont été suivies pour s'attaquer à la problématique de diagnostic. Ces directions s'opposent [dK86] et se complémentent [HD07, HD05a] à la fois. L'approche interprétée consiste à rechercher directement les diagnostics à partir de la description et des observations d'un système, alors que l'approche compilée consiste à réécrire la description d'un système vers un langage plus facilement traitable afin d'accélérer la recherche de diagnostics avec l'arrivée des observations.

Approches interprétées pour le diagnostic centralisé

Par une approche interprétée, la description courante du système et les observations sont prisent en compte au moment de la requête de diagnostic, sans traitement préalable. Dans ce cas (cf Figure 2.8, nous n'avons pas à attendre la fin du calcul avant d'avoir le premier diagnostic. Par contre, le temps de recherche d'un premier diagnostic peut être exponentiel en la taille de la description du système initial [BATJ91]. Le premier argument militant en faveur d'une approche interprétée est la prise en compte des observations au moment de la requête du diagnostic. Cela permet de réduire la complexité théorique du problème. Une observation prend la forme de l'assignation d'une valeur à une variable, et représente donc une valeur en moins à inférer lors du raisonnement. Comme nous l'avons déjà fait remarquer, le second argument en faveur de l'efficacité de l' approche interprétée est les progrès spectaculaires des travaux

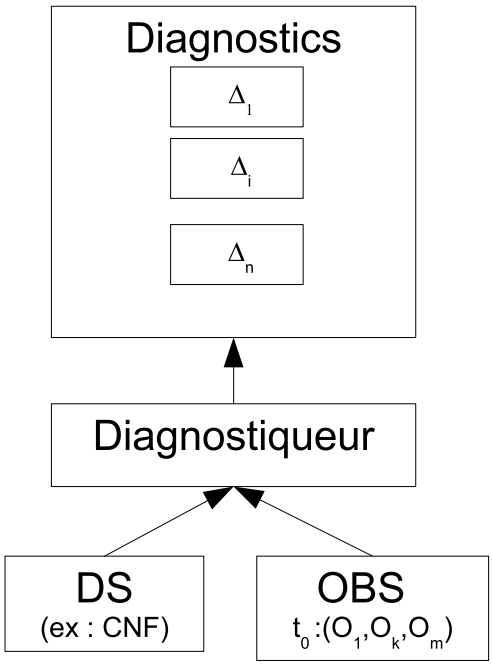


FIGURE 2.8 – Approche interprétée pour le diagnostic centralisé d'un système

sur la résolution du problème de satisfiabilité (SAT). Les solveurs récents sont capables de résoudre dans des temps réduits des problèmes industriels de grandes tailles [EB05, Bie08, AS09]. Les approches interprétées pour le diagnostic tirent bénéfice des résultats sur la satisfiabilité lorsqu'elles étendent les méthodes des solveurs SAT pour la recherche de modèles. Feldman et al [Fel06, Fel10] transposent la problématique de recherche de diagnostics de cardinalité minimale dans le problème Max-SAT et inversement. Comment ? Par exemple l'approche de Grastien et al [GARK07] propose un encodage propositionnel s'inspirant des travaux de [KS92] pour traiter des problèmes de diagnostic de systèmes à évènements discrets par des algorithme SAT [GA08]. Récemment Stein et al [SNL06] mais aussi Feldman et al [FPvG10] montrent expérimentalement qu'utiliser des approches de Max-SAT pour la recherche de diagnostics est une stratégie efficace et rapide. De même les travaux pionniers de De Kleer et Williams [dKW87] ont formulé la problématique de diagnostic à travers celle de satisfaction de contraintes CSP. De Kleer [dK89] a ensuite montré que les méthodes de résolution de CSP pouvaient être utilisées pour répondre aux fonctionnalités de l'ATMS et inversement. Plus récemment les travaux de Stumptner et Wotawa [SW03] proposent de calculer les diagnostics à travers la résolution d'un CSP rendu acyclique. Williams et Ragno [WR03] proposent un calcul des diagnostics cohérent minimaux guidé par la recherche de conflits et l'algorithme A*.

Un des autres avantages de l'approche interprétée est qu'elle prend en entrée la description du système dans son langage de représentation, il n'y a pas de coût additionnel pour stocker la description. L'ajout ou le retrait d'une nouvelle propriété de la description du système se traduit par l'ajout ou le retrait des clauses correspondantes et peut s'effectuer en temps linéaire par rapport à la taille de la description initiale. La rapidité et le faible côut de la mise à jour de la description sont des spécificités s'adressant tout particulièrement à la surveillance des systèmes dynamiques dont la description est amenée à être mise à jour rapidement entre deux requêtes de diagnostics. En résumé, dans un système centralisé, les approches interprétées de recherche de diagnostics minimaux étendent souvent des algorithmes de satisfiabilité booléenne

ou de résolution de CSP. Compte tenu du temps de réponse potentiellement prohibitif ces approches interprétées s'adressent essentiellement à des systèmes dont la requête de diagnostic est occasionnelle. Inversement, puisqu'une approche interprétée calcule directement les diagnostics à partir de la description du système, celle-ci peut constamment être mise à jour entre deux requêtes de diagnostics et ne demande pas de coût supplémentaire de stockage.

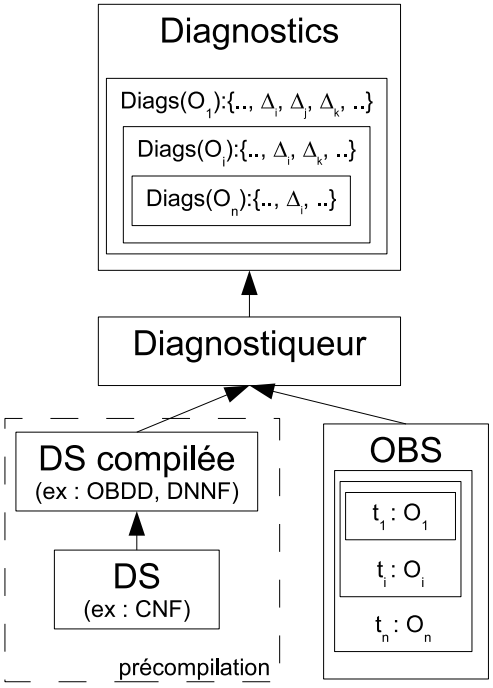


FIGURE 2.9 – Approche compilée pour le diagnostic centralisé d'un système

Approches compilées pour le diagnostic centralisé

Par une approche compilée, la description du système est réécrite vers un langage où la recherche d'un diagnostic est facilement traitable. Par exemple les diagnostics cohérents minimaux d'un système observé se retrouvent notamment à travers l'ensemble des impliquants premiers du système alors que les diagnostics abductifs minimaux se retrouvent à travers l'ensemble des impliqués premiers du système [dKMR92, CCCB96]. Dans ce contexte, Shiny et al [SP99] et Simon et al [SdV01] proposent un calcul efficace et compact des impliquants, resp impliqués premiers. Darwiche [Dar98], Huang et al [HD05b] ont proposé de représenter le système sous forme hiérarchique par une formule DNNF (forme normale négative décomposable), alors que Torta et al [TT02, TT07] se basent sur des OBDDs (diagrammes de décision binaires orientés). Comme nous le montre la Figure 2.9, une fois la description du système compilée, les observations peuvent être prises en compte incrémentalement pour élaborer les diagnostics. La prise en compte des observations et l'élaboration d' un diagnostic minimal a un coût en temps et en espace polynomial par rapport à la taille de la description compilée [HD05b] et font des approches compilées de bon candidats pour le diagnostic en ligne de systèmes. Toutefois, la rapidité des réponses des approches compilées est à pondérer avec la taille potentiellement exponentielle de la description du système réécrit par rapport à sa description initiale [DM02]. De même certaines opérations de mises à jour des descriptions pourront aussi provoquer une transformation potentiellement prohibitive par rapport à la taille de la description compilée [DM02]. Ainsi, cette

approche convient mieux à des systèmes dont les descriptions sont statiques. Nous remarquerons tout de même que ces dernières années, dans le but de réduire la taille des descriptions compilées et d'améliorer leurs mises à jour, les travaux sur la compilation de connaissances t.q. les arbres de BDD de Subbarayan et al [SBH07] revus par [FM09] ont mis en évidence à la fois des structures et des opérations capables de représenter de façon compacte un grand nombre de connaissances.

Compiler ou interpréter, deux approches complémentaires

Les propriétés favorables à une approche interprétée (système dynamique, représentation concise ...) sont défavorables à une approche compilée (système statique, représentation succincte non garantie ...). De même, les domaines où une approche compilée est avantageuse (rapidité des réponses, prise en compte incrémentale des observations, diagnostic en ligne) se présentent comme désavantageux pour une approche interprétée (temps de réponse non garanti, pas de prise en compte incrémentale des observations ni de diagnostic en ligne). Nous résumons les qualités et les défauts de chaque approche dans la Figure 2.10.

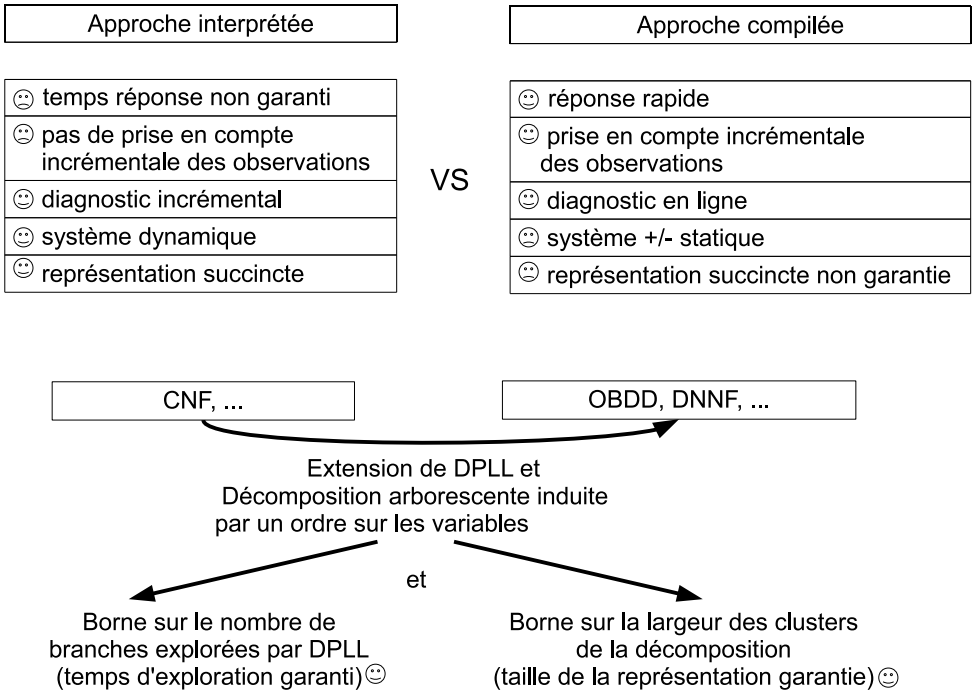


FIGURE 2.10 – Comparaison des approches interprétées et compilées

Toutefois il est possible de compiler une description à partir d'une approche interprétée. Les travaux de Huang et Darwiche [HD07] ont montré qu'en conservant les traces de l'exploration induite par approche interprétée de type DPLL, il est possible de compiler une description FNC vers un OBDD. Ces traces reprennent entre autres l'ordre par lequel DPLL a parcouru les variables et induisent une décomposition arborescente. Cette approche allie la rapidité d'exploration et l'efficacité des approches interprétées basées sur les méthodes de recherche de SAT ou CSP avec la concision des approches compilées telles que les OBDD ou les DNNF à travers la notion de décomposition arborescente. Lorsque l'ordre des variables est connu à l'avance, la largeur de la décomposition arborescente (c-à-d la taille du cluster ayant le plus large vocabulaire) induite par cet ordre permettra à la fois de garantir le temps de recherche (en donnant une borne sur le nombre maximal de chemins que pourra explorer DPLL) et la taille de la représentation

(en donnant une borne sur la taille maximale de l'OBDD). Nous résumons ces propriétés de la construction d'une description compilée par une approche interprétée Figure 2.11.

Les travaux de Darwiche [Dar04] exhibent un compilateur de CNF vers DNNF étendant DPLL beaucoup plus efficace que les compilateurs classiques. Bouquet et Jégou [BJ99] et plus récemment Huang et Darwiche[HD04] utilisent ce même principe pour compiler des OBDDs. Castell base aussi sa procédure de produit unioniste, permettant de compiler des impliquants ou des impliqués premiers [Cas96, CCCB96] sur une extension de DPLL. Pour toutes ces approches interprétées ou compilées, la largeur de la décomposition arborescente est un paramètre de qualité qui permet de borner l'explosion combinatoire inhérente à la complexité en temps et en espace de résolution du nombre de diagnostics.

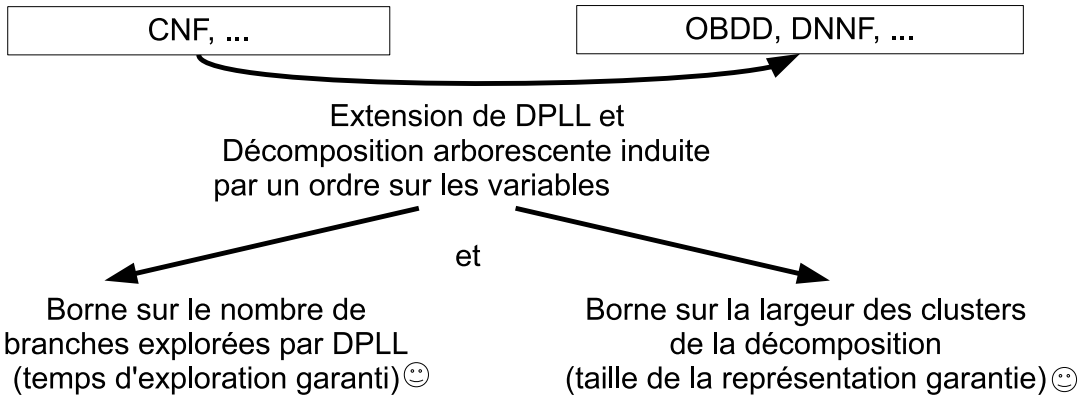


FIGURE 2.11 – Approche compilée par une approche interprétée

2.3 Problématique et travaux liés au diagnostic distribué

Les applications cherchant à préserver la confidentialité de leurs données sont omniprésentes sur le web. Il y a donc un intérêt croissant à pouvoir mener à pouvoir expliquer le comportement de tel systèmes tout en respectant la confidentialité des données de ces applications. Ainsi, en plus des questions usuelles de mises à jour, d'espace mémoire et de rapidité de la réponse du diagnostic auxquelles répondent les approches interprétées ou compilées, le contexte distribué apporte de nouvelles contraintes telles que la répartition géographique ainsi que le contrôle d'accès des descriptions manipulées. Dans un premier temps, nous introduisons informellement la problématique du diagnostic distribué avec contrôle d'accès en reprenant notre exemple de validation de commande. Ensuite, nous présentons comment les approches envisagées dans le contexte distribué se positionnent par rapport aux approches interprétées ou compilées vues précédemment et satisfont les nouvelles contraintes du contexte distribué.

2.3.1 Problématique informelle du diagnostic distribué avec contrôle d'accès

En centralisé, un unique outil de diagnostic a la connaissance des descriptions et observations de tous les composants du système qu'il supervise. En distribué, chaque pair diagnostiqueur a l'unique connaissance du comportement et des observations du sous-système qu'il supervise. Les diagnostiqueurs se comportent comme un véritable système pair-à-pair et peuvent être mis à jour, rejoindre ou quitter le réseau. Aucun pair diagnostiqueur n'a la connaissance du système global ni ne peut s'adresser directement à l'ensemble des autres pairs du réseau. Chaque pair ne

pourra raisonner et élaborer des diagnostics localement cohérent qu'à partir d'une description lui étant accessible et ne pourra vérifier la cohérence de ses diagnostics qu'avec son voisinage. Nous illustrons les différentes contraintes de la problématique de diagnostic distribué avec contrôle d'accès sur l'exemple du diagnostic des services de validation de commandes.

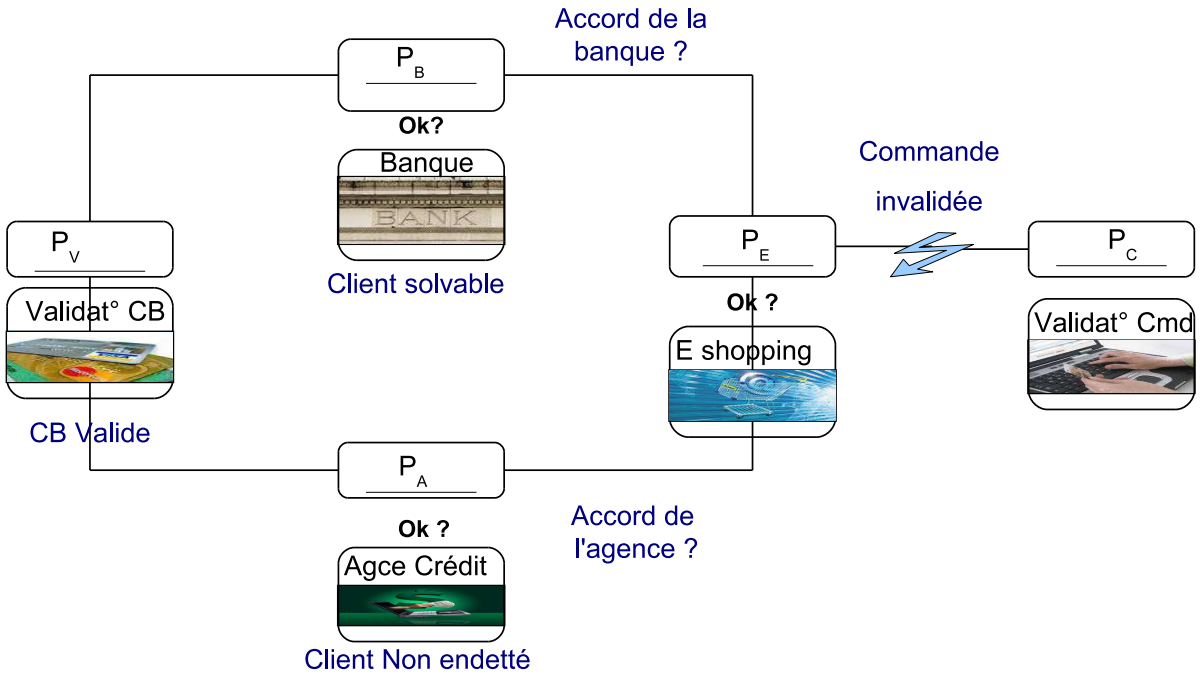


FIGURE 2.12 – Diagnostic distribué du processus de validation d'une commande

Exemple. 2.11 (Contrôle d'accès des services de validation de commande) *Reprenons l'exemple introductif cette fois-ci dans un contexte intrinsèquement distribué. Considérons à nouveau le scénario figure 2.12 où un client voit sa commande invalidée alors qu'il est solvable, qu'il n'est pas endetté et que sa carte bancaire est valide. Compte tenu de ce scénario anormal, le but du diagnostic est d'identifier le ou les services n'ayant pas suivi leur comportement attendu. Dans le cadre du diagnostic distribué, chaque service est surveillé par un pair diagnostiqueur ayant la connaissance de son comportement attendu et de ses observations. Ainsi, le pair P_B surveille le service de la banque. De même le pair P_E (resp. P_A, P_V, P_C) surveille le service d'e-shopping (resp. de l'agence, de validation de carte bancaire et de validation de la commande). Nous considérerons une politique d'accès gardant secret le mode de fonctionnement de chaque service et confidentielle toute information échangée entre les services hormis la validité de la carte bleue. Compte tenu de cette politique d'accès, aucun pair ne pourra regrouper la description globale du système ni même être en mesure de composer les diagnostics globaux du système.*

Problématique :
Le défi majeur du diagnostic distribué avec contrôle d'accès est de pouvoir expliquer le comportement d'un système distribué, par un ensemble de pairs ayant un accès et une visibilité réduits de ce système, tout comme l'aurait fait un unique diagnostiqueur ayant la connaissance globale de tout le système.

2.3.2 Travaux liés au diagnostic distribué

Dans la littérature, le passage du diagnostic de systèmes au diagnostic de systèmes distribués s'est accompagné d'un changement de formalisme. Du cadre logique, essentiellement appliqué au diagnostic de circuits électroniques, les travaux de Cassandras et Lafortune [CL99] et Sampath et al [SSL+96] ont proposé un cadre de systèmes à évènements discrets (automates [SW05], [PC05], réseau de Petri [BHFJ03], algèbre de processus [CPR02]) pour le diagnostic de réseaux de télécommunication [PC05], de réseaux électriques, la coordination de robot [KK05] ou encore plus récemment de services web [LYDM09]. Alors que le diagnostic à base de cohérence explique le comportement observé d'un système en identifiant ses éventuels composants anormaux, le diagnostic de systèmes à évènements discrets explique le comportement observé d'un système en retraçant les éventuelles exécutions suivies par le système. Toutefois, lorsque l'ensemble des exécutions finies d'un système à évènements discrets se traduit par un ensemble de modèles d'une formule propositionnelle (Grastien et al [GARK07]), diagnostiquer le système à évènements discrets revient à énumérer les modèles préférés ou encore les diagnostics cohérents de la formule correspondante.

D'autre part, tout comme en centralisé, la complexité de calcul du diagnostic dans les systèmes multi agents est aussi un problème difficile [RtTBW02] mais se mesure aussi par le nombre de messages, la taille des messages ou encore la taille des descriptions locales manipulées. Afin de s'attaquer à la complexité inhérente au calcul du diagnostic, les approches envisagées pour le diagnostic distribué étendent les principes des approches compilées ou interprétées que nous avons abordés pour le contexte centralisé. Afin de mieux comprendre les travaux relatifs au diagnostic distribué, nous les classifierons en fonction de leur dispositif de surveillance. Nous distinguons le dispositif distribué de diagnostiqueurs du dispositif supervisé par un unique diagnostiqueur. Dans un dispositif distribué, tous les agents sont égaux et exécutent le même protocole. Nous avons constaté que la plupart des approches interprétées du contexte distribué trouvent une reformulation dans ce dispositif. Dans un dispositif supervisé, un agent particulier, le superviseur, organise le calcul entre les agents. Il a accès à la connaissance du système global. De même nous avons constaté que différentes variantes des approches compilées du contexte distribué trouvent une reformulation dans ce dispositif. Nous détaillons les différentes variantes de l'état de l'art envisagées pour les dispositifs supervisés et distribués.

Le dispositif supervisé par un diagnostiqueur

Les approches compilées telles qu'elles ont été envisagées dans un contexte centralisé trouvent une correspondance presqu'immédiate dans le contexte distribué par l'intermédiaire d'un dispositif supervisé. Dans ce dispositif où un agent particulier, le superviseur, organise le calcul, nous distinguons quatre variantes.

La variante supervisée-compilée Dans la variante supervisée-compilée (illustrée Figure 2.13), le superviseur regroupe les descriptions (DS_{A_i}) des agents (A_i) qu'il chapeaute et les compile en une seule seule description (par exemple OBBD, DNNF, ...). À partir des observations (O_{A_j}, O_{A_i}, ...) reçues jusqu'au temps t_k, le superviseur retourne les diagnostics expliquant le comportement observé jusqu'à la dernière observation reçue. Torta et al [MTT04] suivent cette variante et proposent une méthode en ligne de diagnostic de systèmes multi agents, où chaque agent modélise le comportement d'un robot. Dans cette approche le module de surveillance en ligne, le superviseur, compile les comportements des robots par l'intermédiaire d'un OBDD. Au fur et à mesure des observations reçues, le module est capable de détecter des fautes dans le comportement d'un robot ou encore des erreurs d'interactions entre robots. La variante

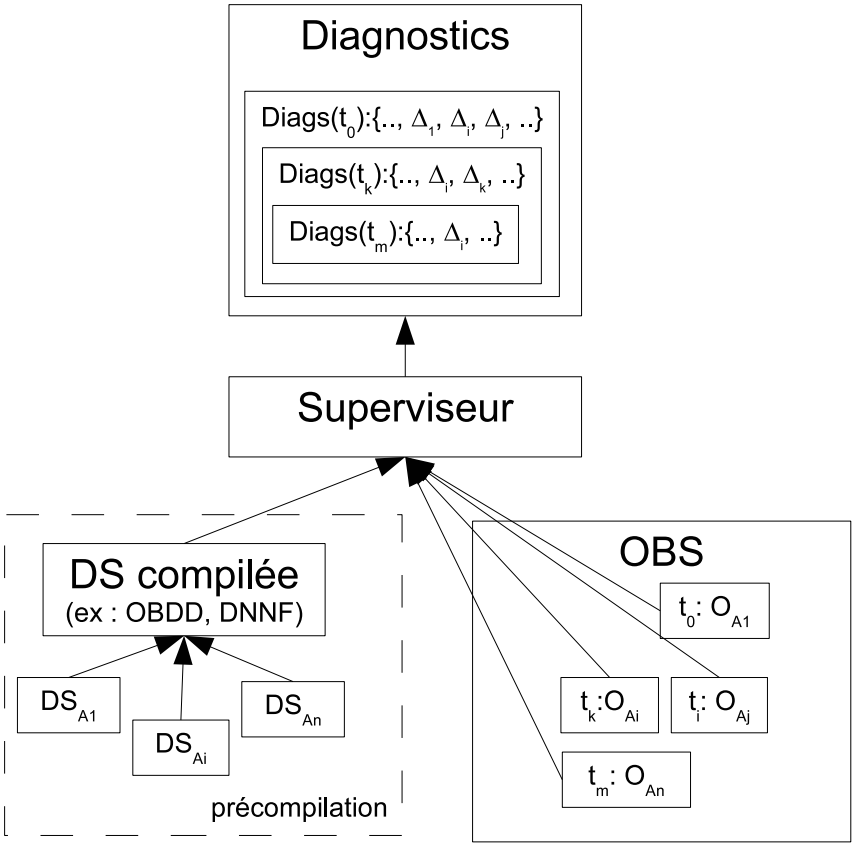


FIGURE 2.13 – Approche supervisée-compilée pour le diagnostic de systèmes distribués

supervisée-compilée transfère les qualités et les défauts des approches compilées du contexte centralisé vers un contexte distribué. Ainsi elle permet une mise à jour en ligne des diagnostics au fur et à mesure de l'arrivée des observations. Le calcul d'un diagnostic est rapide (polynomial dans la taille du langage compilé). La description non compilée de chaque agent est transmise au superviseur qui la compile. Dans ce cas, le superviseur a donc accès à la connaissance globale de tout le système. En raison de la politique de confidentialité c'est une hypothèse que nous ne faisons pas. Les observations sont transmises une fois qu'elles sont mesurées. Cette variante requiert donc un faible nombre de messages de taille raisonnable et l'essentiel du calcul est effectué par le superviseur. Tout se passe comme un système centralisé. Malgré les différentes optimisations, la taille de la description compilée par le superviseur est potentiellement exponentielle en la taille de la somme des descriptions des différents sous-systèmes. En définitif, pouvoir concentrer la compilation de l'union des descriptions du système distribué en un agent est souvent un calcul difficile voire impossible en pratique [RtTW03, PC05]. De plus, cette hypothèse ne répond pas à la contrainte de confidentialité des descriptions nécessaire dans la problématique de diagnostics avec contrôle d'accès. Les variantes suivantes ne font pas cette hypothèse et répartissent l'effort de calcul à travers les agents.

La variante : supervisée - semi compilée Dans la variante supervisée semi-compilée (illustrée Figure 2.14), les diagnostics locaux ($\Delta_k^{A_i}$, ...) sont tout d'abord compilés par chaque agent A_i à partir des observations O_{A_i} et des descriptions locales DS_{A_i}. Ces diagnostics sont ensuite composés par le superviseur afin d'élaborer des diagnostics globaux (Δ_j^G). La compi-

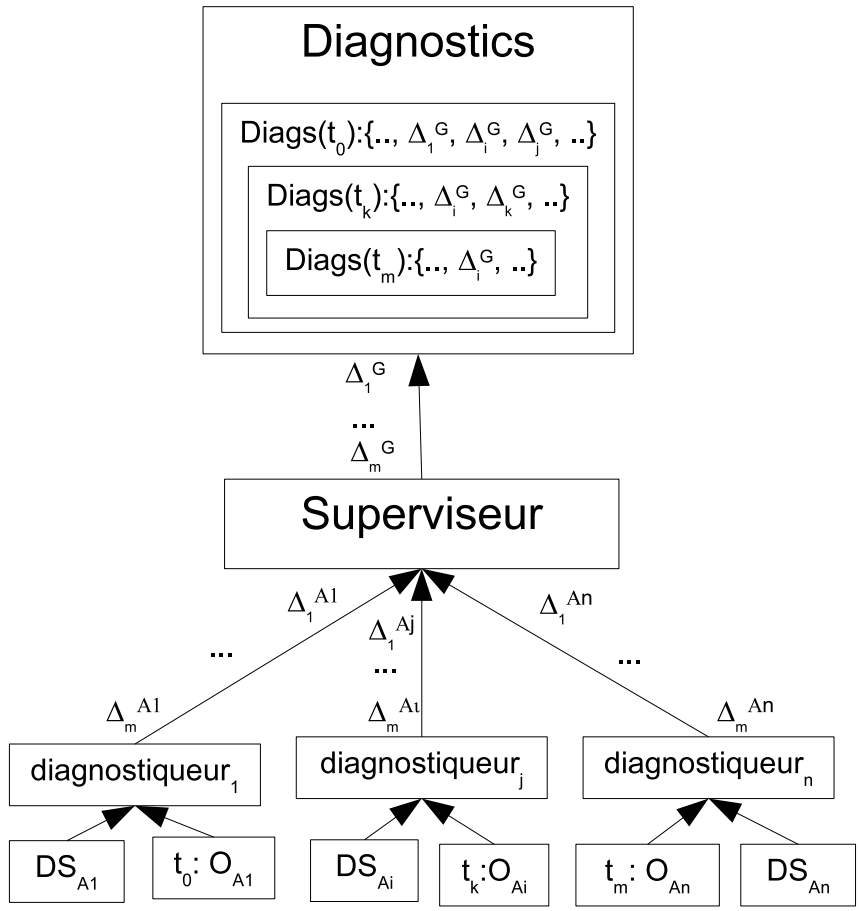


FIGURE 2.14 – Approche supervisée semi-compilée pour le diagnostic de systèmes distribués

lation des diagnostics globaux est donc répartie à la fois à un niveau local chez chaque agent mais aussi à un niveau global chez le superviseur. Console et al [CPD07] introduisent un cadre supervisé pour le diagnostic à base de modèles. Dans ce contexte un superviseur étend les diagnostic globaux dans une boucle où les diagnostiqueurs locaux sont visités à tour de rôle. Le superviseur prend connaissance de l'interaction entre les sous-systèmes au moment du diagnostic et permet à l'approche de s'adresser à des systèmes reconfigurables tout en nécessitant un faible nombre de messages potentiellement de grande taille (cf comparaison de différentes stratégies de diagnostics par Kalech et Kaminka [KK03]). Dans un formalisme de réseau d'automates communicants, Pencolé et Cordier [PC05] calculent les traces locales correspondant aux différentes exécutions possibles de sous-systèmes. Ces traces représentent des diagnostics locaux qui sont par la suite composés par un superviseur pour obtenir des diagnostics globaux. Par un mécanisme de fenêtres temporelles, de nouvelles observations et de nouveaux diagnostics sont élaborés à partir des diagnostics précédents. Cordier et Grastien [CG07] améliorent l'efficacité de la dernière approche en distinguant des moments de faibles interactions entre les systèmes conduisant à des diagnostics indépendants plus faciles à calculer. En résumé, la variante supervisée semi-compilée trouve un compromis entre les avantages et les inconvénients des approches interprétées et compilées. Localement les diagnostics compilés par les sous-systèmes reprennent donc toutes les caractéristiques des approches compilées. Globalement les diagnostics composés par le superviseur reprennent donc toutes les caractéristiques des approches interprétées. Cette

compilation locale est en plus facilitée par une bonne observabilité du système. L'effort de calcul est réparti à la fois à un niveau local par le calcul des diagnostics locaux chez chaque agent et à un niveau global chez le superviseur. De plus nous remarquerons qu'une bonne observabilité du système permet de diminuer le nombre de diagnostics locaux et donc de faciliter la formation de diagnostics globaux. Malheureusement compte tenu que le superviseur est en mesure d'expliquer le comportement global du système, cette variante ne répond pas non plus aux exigences du diagnostic distribué avec contrôle d'accès.

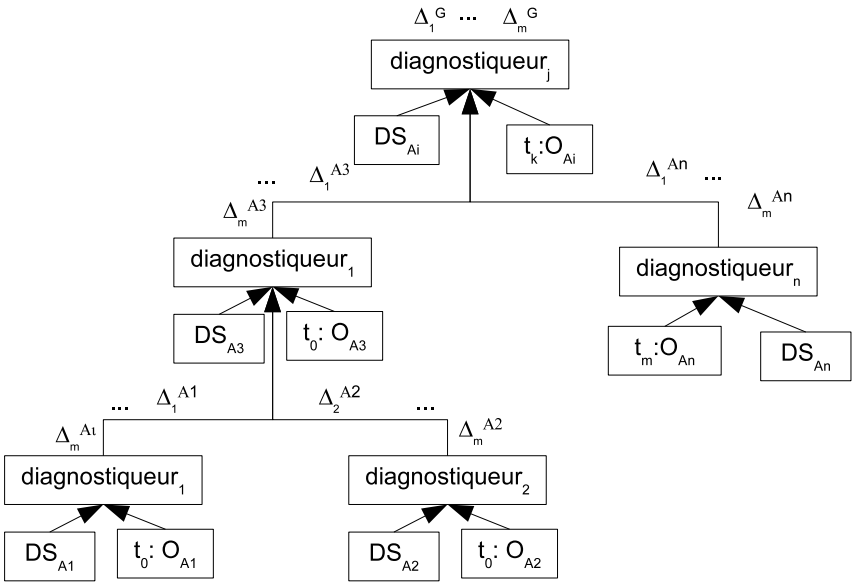


FIGURE 2.15 – Approche semi-supervisée semi-compilée pour le diagnostic de systèmes distribués

La variante : semi supervisée - semi compilée Dans la variante semi-supervisée semi-compilée illustrée Figure 2.15, la description du système global est préalablement décomposée, éventuellement par un superviseur. Les diagnostics minimaux locaux $\Delta_k^{A_i}$ sont remontés par un protocole distribué des feuilles vers la racine de telle sorte que la racine regroupe les diagnostics globaux Δ_m^G. Ainsi Provan [Pro02] suppose que le système est préalablement décomposée en une structure hiérarchique de clans. Ensuite, par un protocole distribué, les diagnostics locaux à chaque clan sont remontés et composés des feuilles à la racine. Au cours de leur remontée les diagnostics reçus sont abstraits par de nouveaux symboles. Cet algorithme distribué peut être vu comme une version distribuée du calcul centralisé des diagnostics à travers un système hiérarchique [Pro01, Moz91]. Su et Wonham dans [SW06] illustrent aussi un calcul hiérarchique et distribué des diagnostics mais cette fois ci appliqué à un cadre de réseaux d'automates. Dans ce cadre, les traces symbolisant les exécutions possibles d'un système sont remontées et synthétisées à travers une structure arborescente. La racine compose les diagnostics globaux. De même, Kurien et al [KKZ02] supposent aussi que la description du réseau décrivant un système embarqué est préalablement décomposée vers une structure arborescente et utilisent les résultats de Freuder [Fre82] et Dechter [Dec92] sur l'équivalence entre cohérence locale et cohérence globale des systèmes décomposés et peuvent ainsi garantir que tout diagnostic local se prolonge par un diagnostic global du système. De même, dans un formalisme de système à évènements discrets Su et Wonham [SW05] puis John et Grastien [JG08], tirent parti de l'équivalence entre cohérence locale et cohérence globale de traces d'un réseau d'automates structurés. Toujours dans une

variante semi-supervisée semi-compilée, mais sans hypothèse préalable de décomposition, Biteus et al [BFN06, BNF08] considère qu'un coordinateur ordonne et partitionne les agents à travers des modules pour le calcul des diagnostics de cardinalité minimale. Contrairement aux variantes vues précédemment la variante semi supervisée - semi compilée représente les diagnostics globaux du système à travers un ensemble de diagnostic locaux. Dans la mesure où les diagnostic locaux restent locaux, ces approches répondent en partie à la contrainte de confidentialité du diagnostic distribué. Malheureusement les différentes techniques de cette variante suggèrent une transformation du système vers une structure arborescente nécessitant une connaissance globale de la description du système.

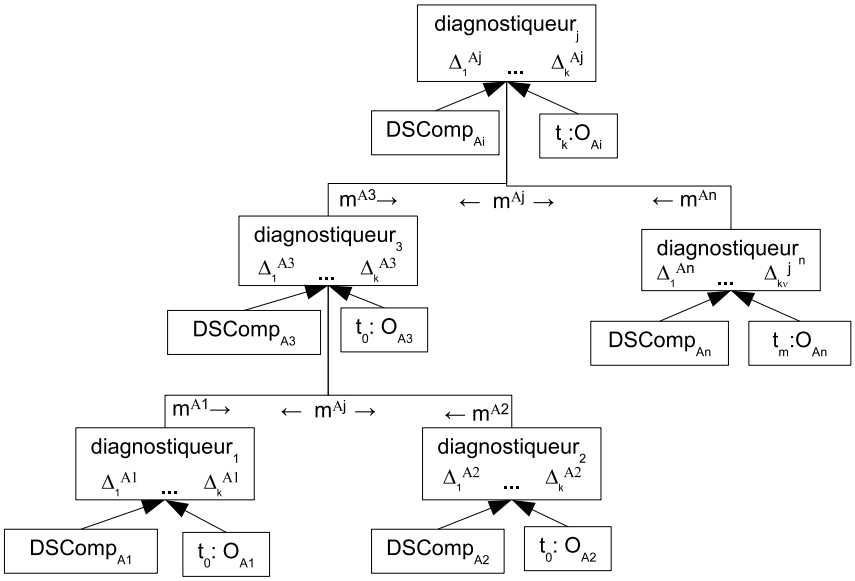


FIGURE 2.16 – approche semi-supervisée compilée pour le diagnostic de systèmes distribués

La variante semi-supervisée compilée Dans la variante semi supervisée et compilée (illustrée Figure 2.16), la description du système global est préalablement décomposée éventuellement par un superviseur de telle sorte que l'union des descriptions locales représente la description compilée d'un système global. Lorsque la compilation d'un système global est répartie à travers les agents. Dechter [DD96] propose de compiler la description d'un système vers un réseau de contraintes acyclique. Grâce aux propriétés des réseaux structurés, les différentes fonctionnalités de l'ATMS sont calculées par un nombre raisonnable de messages. Retrouver un diagnostic global se fait par une propagation de proche en proche de modèles locaux (c-à-d m^{A_i} dans la figure 2.16) ou encore de diagnostics locaux comme vu précédemment pour la variants semi-supervisée, semi-compilée. Dans un formalisme de système à évènements discrets Fabre et Benveniste [FB07] présentent un algorithme distribué chaotique pour le calcul de traces locales globalement cohérentes. Par rapport à une variante semi-supervisée semi-compilée, la variante semi-supervisé compilée permet un diagnostic en ligne. De même, parce que l'union des descriptions locales représente un système compilé un diagnostic peut être retrouvé par un nombre de messages raisonnable. Par contre, la taille maximale des descriptions locales manipulées par chaque agent par une variante semi-supervisée et compilée, peut être bien supérieure à la taille de la description manipulée par une variante semi-supervisée semi-compilée. En effet représenter tous les modèles d'une formule locale dans [DD96] est prohibitif. De plus, nous remarquerons,

tout comme pour l'approche semi-supervisée semi-compilée que cette variante suggère un partitionnement supervisé du système qui nécessite une connaissance globale de la description du système.

Le dispositif distribué de diagnostiqueurs

Dans le dispositif distribué, chaque agent a la connaissance de la description d'un sous-système qu'il supervise. Les agents de diagnostic sont tous égaux, il n'y a pas de superviseur. Nous distinguons deux variantes implémentant ce dispositif.

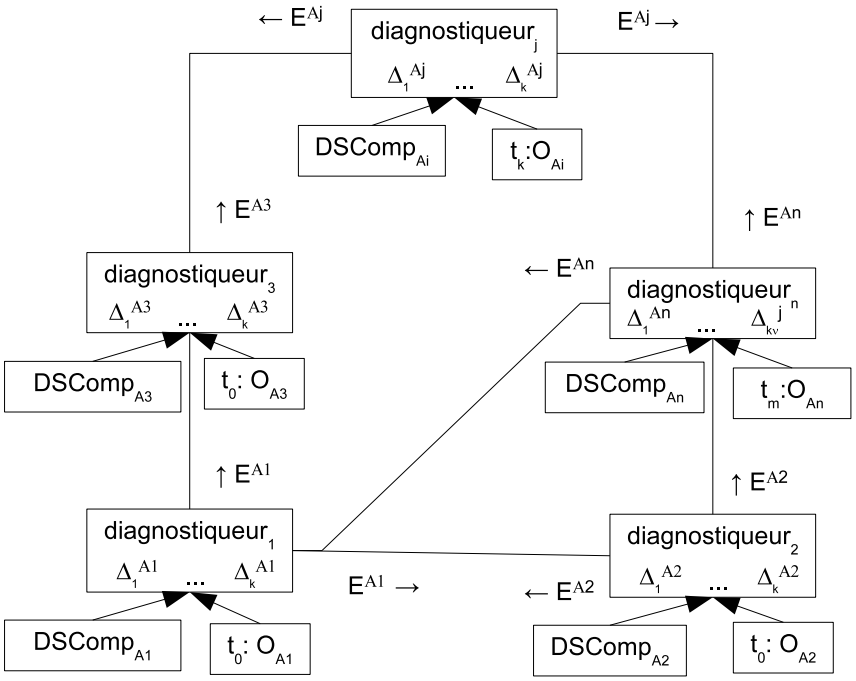


FIGURE 2.17 – Approche distribuée interprétée pour le diagnostic de systèmes distribués

La variante distribué - interprétée La variante distribuée interprétée, illustrée Figure 2.17, transpose dans un contexte distribué les techniques des approches interprétées utilisées pour le calcul de diagnostic abductif, les méthodes de recherche basées sur SAT ou encore des méthodes de recherche issues des CSP. Mason et al [MJ89] puis Beckstein et al [BFK93] et Roos et al [RtTW03] proposent une extension de l'ATMS de de Kleer [dK86] applicable au contexte distribué. Dans ces travaux, les nouvelles conséquences inférées localement (représentées par E^{A_i} dans la Figure 2.17) sont transmises de proche en proche par le biais de messages entre différents systèmes de maintien de vérité. Dans ce cas, le pair initiateur regroupe à la fin du calcul distribué les diagnostics globaux. Kalech et Kaminka [KK05] appliquent les principes de la révision de croyances et du raisonnement abductif de l'ATMS dans un contexte distribué pour le diagnostic de coordination entre robots. Plus récemment Adjiman et al [ACG+06], Gia Hien Nguyen et al [CNR06], et Abdallah et al [AGR09] ont proposé une amélioration de ces techniques d'inférence distribuée en étendant le principe de la résolution dirigée [Ino92] pour les systèmes pair à pair. D'autre part, en se basant sur la problématique de diagnostic et les CSP [dK89], Meir Kalech et al [KKME06] recherchent l'ensemble des diagnostics minimaux pour des problèmes de coordination de robots en utilisant des algorithmes de CSP distribués [YDIK98].

Ainsi la recherche de diagnostics distribués par une variante interprétée peut aussi tirer parti de l'efficacité des différentes méthodes de CSP distribué (propagation avant asynchrone [MZ07], retour arrière asynchrone [BMBM05]) qui ont étendues avec succès des techniques de CSP envisagées dans un cadre centralisé (propagation avant [BG95], retour Arrière [DF02]). Les approches utilisant un dispositif distribué pour résoudre la problématique de diagnostic reprennent à la fois les qualités et les défauts des approches interprétées du contexte centralisé. La description des sous-systèmes et les observations sont interprétées directement à partir du langage de représentation. L'ajout ou le retrait d'une nouvelle propriété à la description des sous-systèmes se fait en temps linéaire. Les dispositifs envisagés sont totalement distribués et ne supposent pas de superviseurs. De plus, tout comme dans un contexte centralisé où le temps de calcul est potentiellement prohibitif, les méthodes de recherche dans un contexte distribué nécessitent un nombre de messages non concurrents potentiellement exponentiel. Toutefois certains travaux dans le raisonnement distribué (Adjiman et al [ACG+06], Gia Hien Nguyen et al [CNR06], ou encore Abdallah et al [AGR09]) respectent la confidentialité des connaissances locales en vérifiant qu'à l'étape du raisonnement aucun pair ne divulgue ses connaissances locales. D'autre part Yokoo et al [YSH02] proposent un protocole permettant de sécuriser la résolution d'un problème de satisfaction de contraintes distribué par l'encryption des information partagées par les agents. Ce protocole garantit qu'aucune information privée et aucun renseignement sur les assignations décidées par les agents ne sont révélées. Ainsi la variante distribuée Interprétée répond à toutes les contraintes du diagnostic distribué. Le seul bémol réside dans le nombre de messages qui est potentiellement exponentiel et pour certaines approches de taille exponentielle. Le problème du nombre de messages est résolu par la variante suivante.

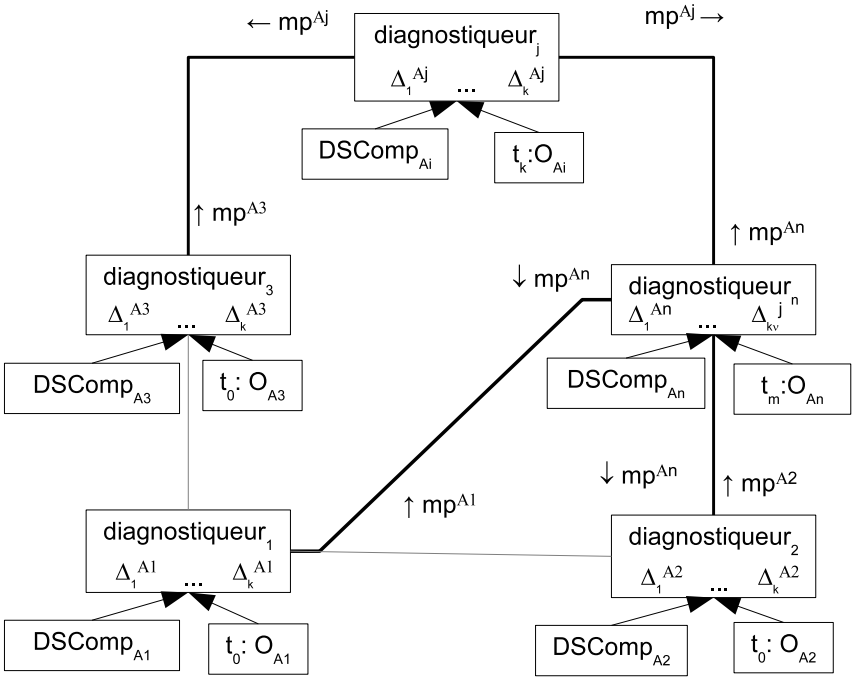


FIGURE 2.18 – Approche distribuée semi-compilée pour le diagnostic de systèmes distribués

La variante distribuée semi-compilée Dans une variante distribuée semi compilée (Figure 2.18), chaque diagnostiqueur compile ses diagnostics localement cohérents à travers une

structure arborescente. Les diagnostic globaux se retrouvent en prolongeant les diagnostics locaux. Nous ne connaissons pas de travaux sur le diagnostic distribué pour la variante distribué semi-compilée. Néanmoins, compte tenu de la relation existante entre la problématique de diagnostic et celle d'optimisation de contraintes (COP) (cf Dechter et Fattah [FD95]), les travaux de Petcu et Faltings [PF05] DPOP, dans le cadre des COP distribués, exhibent des techniques de décomposition arborescente et de programmation dynamique. Les solutions de DPOP peuvent correspondre aux diagnostics de cardinalité minimale. Dans des travaux récents, Silaghi et Faltings [SF02] puis Faltings et Leauté [FLP08, LF11] adaptent l'algorithme DPOP [PF05] pour qu'il réponde aux exigences de confidentialité des données en encryptant des décisions prises par les agents.

Les protocoles de raisonnement avec confidentialité envisagés dans ce cadre autorisent un agent à raisonner à partir d'une information en-cryptée ne lui étant pas destinée. Dans notre problématique de raisonnement avec contrôle d'accès un pair ne pourra raisonner qu'à partir des formules lui étant destiné et auxquelles il a accès.

Synthèse Nous résumons les différentes caractéristiques des approches compilées, interprétées, supervisées et distribuées à travers la Figure 2.19. Les travaux indiqués en gras dans le tableau respectent en partie les exigence de confidentialité pour le diagnostic distribué.

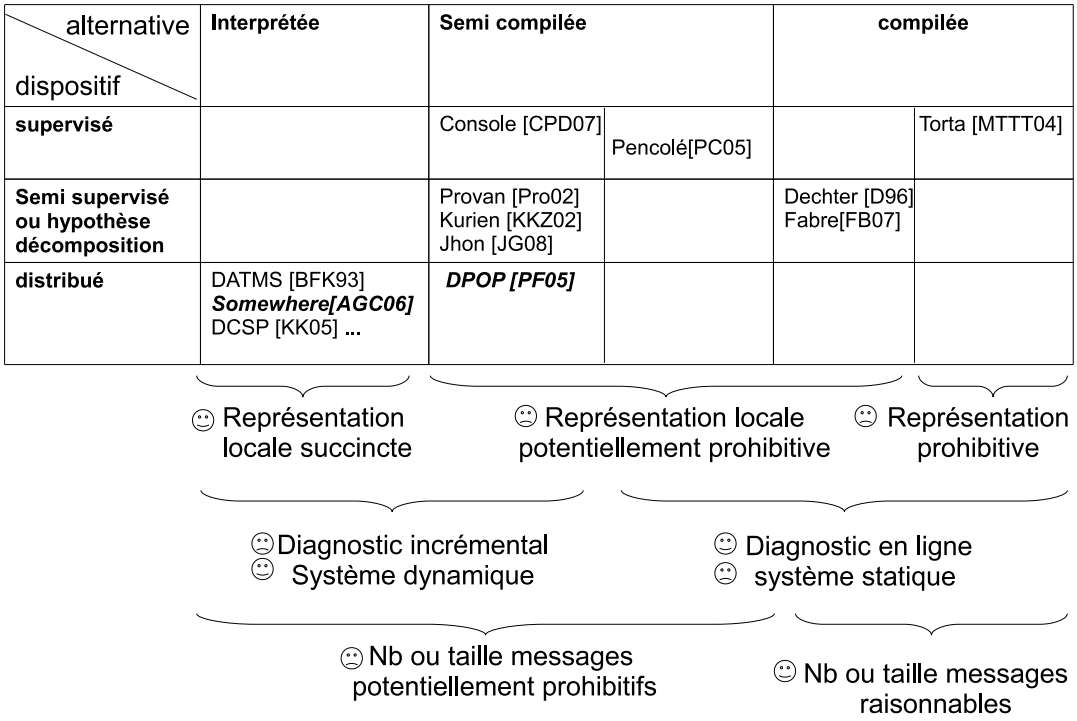


FIGURE 2.19 – Résumé des approches pour le diagnostic de systèmes distribués

Parmi les approches envisagées dans un contexte distribué, les approches interprétées ont trouvé une traduction naturelle vers un dispositif totalement distribué. Plus difficilement, les approches compilées ou semi compilées se sont transposées dans le contexte distribué par l'intermédiaire d'un superviseur ou en faisant l'hypothèse de la décomposition préalable du système

global. À notre connaissance, les seules approches compilées ou semi-compilées totalement adaptées à un contexte distribué sont les variantes de l'algorithme DPOP. Toutefois ces approches n'ont pas été pensées pour les problèmes de diagnostic distribué et notamment la recherche de diagnostics minimaux. Dans la suite nous proposons un protocole distribué construisant une meilleure décomposition arborescente que DPOP. Tout de suite nous introduisons le cadre nous permettant de retourner les diagnostics minimaux par un raisonnement totalement distribué tout comme l'aurait fait un unique diagnostiqueur.

Chapitre 3

Cadre du diagnostic distribué de systèmes avec contrôle d'accès

L'argument majeur favorisant un raisonnement dans un contexte distribué au lieu d'un contexte centralisé est la possibilité de pouvoir raisonner en présence de larges systèmes intrinsèquement distribués. Dans de tels systèmes, les différents composants géographiquement répartis communiquent entre eux par l'intermédiaire de messages. De même, lorsque le diagnostic s'adresse à un grand nombre de composants, il est facile de s'imaginer que toute fusion des descriptions des composants d'un système à travers un superviseur se transformerait en problème intraitable et pourrait donc être à proscrire pour des raisons de complexité.

De plus, la plupart des approches envisagées dans le cadre du diagnostic distribué ne prévoient pas de politique où le partage du vocabulaire entre les pairs est limité. En présence d'applications où l'accès à l'information est restreint par une politique d'accès, ces approches peuvent donc être inadaptées. Dans ce chapitre, nous présentons un cadre permettant d'expliquer les comportements anormaux d'un système distribué où le partage de l'information est restreint par une politique d'accès. À partir de ce cadre, nous énonçons formellement la problématique de diagnostic de systèmes distribués respectant une politique de contrôle d'accès. Ensuite, dans le chapitre 4 nous proposons d'identifier les caractéristiques graphiques des systèmes pour lesquels la problématique de diagnostic distribué peut être résolue de façon sûre. Nous verrons que cette caractérisation tisse des liens étroits entre la problématique de diagnostic de système distribué et celle de la cohérence globale de descriptions locales. Dans le chapitre 5, nous tirons parti de ces propriétés en présentant un algorithme distribué permettant de décomposer un système en une structure arborescente en vue de le diagnostiquer.

3.1 Système de pairs diagnostiqueurs avec contrôle d'accès

La plupart des approches envisagées dans le cadre du diagnostic distribué ne prévoient pas de politique d'accès au partage du vocabulaire des descriptions des systèmes. En présence d'applications où l'accès à l'information est restreinte par une politique d'accès, ces approches sont donc inadaptées. Nous introduisons les systèmes de pairs diagnostiqueurs qui permettent d'expliquer les comportements anormaux d'un système distribué et en plus modélisent : le réseau d'accointances (support du partage de l'information) et la politique de contrôle d'accès (limitant la connaissance du système global accessible à chaque pair). À l'aide de ce cadre, nous énonçons la problématique de diagnostic distribué avec contrôle d'accès.

3.1.1 Le graphe d'accointances

Dans la plupart des applications distribuées, il est possible pour des pairs d'interagir en échangeant des connaissances par le biais de messages. C'est le cas par exemple des services webs qui peuvent interagir les uns les autres à partir d'interfaces d'appels spécifiques. C'est aussi le cas d'applications distribuées du Web Sémantique [ST05, HIST05, AGR09], chaque pair décrivant une ontologie pouvant interagir avec les autres pairs dont l'ontologie contient des concepts sémantiquement proches. Afin de diagnostiquer le comportement d'un système géographiquement réparti, nous nous intéresserons à l'élaboration d'algorithmes où la communication entre les différents pairs du système se fait par l'intermédiaire de messages. Un message $m_{i,j}$ est un ensemble de formules ou variables échangé entre les pairs p_i et p_j. À l'envoi du message $m_{i,j}$, son contenu est connu de l'expéditeur p_i. À la réception du message, son contenu est connu du destinataire p_j. Tout message $m_{i,j}$ envoyé par un pair p_i est reçu par p_j et transite par un canal de communication bidirectionnel asynchrone. Un pair ne pourra interagir qu'avec un nombre limité de voisins. L'ensemble des liens de voisinage définit le graphe d'accointances du système.

Définition 3.1 (Graphe d'accointances) *Soit P un ensemble de pairs d'un système, le graphe d'accointances $G(P, ACQ)$ est t.q. :*

- *chaque noeud de G modélise un pair $p \in P$,*
- *tout lien $\{p_i, p_j\} \in ACQ$ entre pairs traduit la possibilité d'échange de messages entre p_i et p_j.*

Nous illustrons Figure 3.1 le graphe d'accointances des pairs du système de validation de commande. Un graphe d'accointances peut être vu comme le support du partage de l'information.

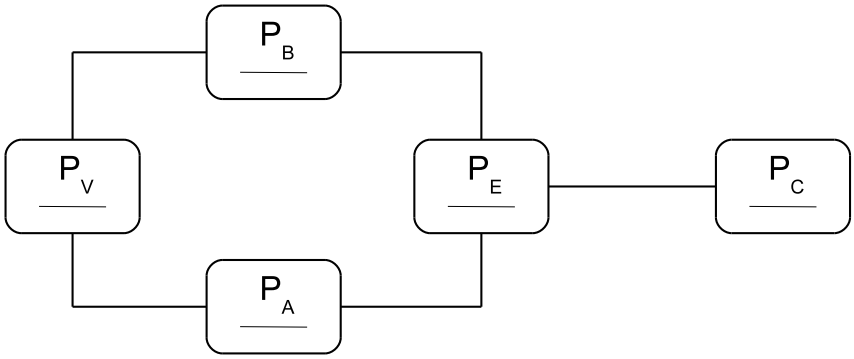


FIGURE 3.1 – Graphe d'accointances du système de validation de commande

Il est indépendant de la description du comportement des pairs. Dans notre modélisation, il y a une indépendance entre le support du partage de l'information et l'information partagée entre pair. Ce n'est pas toujours le cas, par exemple les travaux de [AGR09] considèrent que les liens de communications sont dirigés par des éléments de la description de chaque pair.

3.1.2 Le contrôle d'accès

Par souci de généralité, nous nous sommes intéressé à modéliser une politique de contrôle d'accès quelconque portant sur le vocabulaire des pairs. Ainsi, chaque pair aura accès à un certain vocabulaire lui permettant de décrire une connaissance, raisonner, mettre à jour ou encore échanger des formules avec ses voisins. Nous modélisons simplement le contrôle d'accès par une fonction $K : P \times Vars \rightarrow \{\top, \bot\}$ t.q. $K(p, v) = \top$ si le pair p a accès à la variable v,

$K(p, v) = \bot$ sinon. Bien que K soit présentée comme fonction globale, nous considérons qu'elle a la capacité de modéliser l'union des différentes politiques d'accès locales suivies par chaque pair. Notons toutefois que, si cette politique autorise pour l'instant des règles assez complexes de restriction du partage de l'information (par exemple une variable peut être locale à un sous ensemble de pairs), nous serons contraints (pour des raisons heuristiques) de restreindre cette liberté à partir du chapitre 5 : une variable sera soit publique soit privée (de manière identique à Somewhere [ACG+06]).

Pour un pair p, une politique d'accès limite strictement la connaissance du pair au vocabulaire et formules auxquels il a accès.

Définition 3.2 (Vocabulaire accessible, vocabulaire accessible partagé) *Soit $K : P \times Vars \rightarrow \{\top, \bot\}$ une fonction modélisant le contrôle d'accès dans le réseau :*

1. *V_p^∇ représente le vocabulaire accessible par le pair p t.q. $V_p^\nabla = \{v : K(p, v) = \top\}$.*
2. *le vocabulaire accessible sera dit partagé entre deux pairs p_i et p_j ssi il est égal à $V_{p_i}^\nabla \cap V_{p_i}^\nabla$*

Un pair ne pourra raisonner qu'à partir de son vocabulaire accessible. Les messages reçus ou envoyés à un voisin ne pourront être formés qu'à partir d'un vocabulaire accessible partagé. Le contenu d'un message pourra être codé par exemple par une méthode de chiffrement asymétrique. Dans ce cas un pair qui souhaite communiquer des connaissances sensibles à un voisin conserverait de son côté une clé privée pour crypter le contenu du message et communiquerait au voisin concerné une clé publique afin que le contenu d'un message soit décrypté. Dans ce sens, la politique de contrôle d'accès pourrait être implémentée par des protocoles bien connus t.q. l'encryptage d' El Gamal [Gam84] ou le protocole PGP "Pretty Good Privacy" de Zimmerman [Zim95].

Dans l'exemple suivant, nous introduisons dans le vocabulaire P_B deux nouveaux mode de fonctionnement $st0$ et $st1$ pour le pair P_B. Ils correspondent respectivement à un collage à 0 et un collage à 1 de la variable v et raffinent le mode $\neg okB$ (voir [dKW89] pour une extension à plusieurs modes comportementaux).

Exemple. 3.1 (Contrôle d'accès des services de validation de commandes) *Considérons la politique de contrôle d'accès énoncée dans l'exemple 2.11.*

1. *Chaque pair garde peut lire son mode de fonctionnement et ses variable locales ($K(P_E, okE) = \top$, $K(P_B, okB) = \top$, $K(P_B, st1) = \top$ $K(P_B, st0) = \top$, $K(P_A, okA) = \top$), $K(P_A, e) = \top$), $K(P_B, s) = \top$).*
2. *Tout vocabulaire échangé entre pairs est accessible aux pairs, exception faite pour v.*
   - *Seuls les pairs P_E et P_B ont accès à la variable b ($K(P_E, b) = K(P_B, b) = \top$).*
   - *Seuls les pairs P_E et P_A ont accès à la variable a ($K(P_E, a) = K(P_A, a) = \top$).*
   - *Seuls les pairs P_E et P_C ont accès à la variable c ($K(P_E, c) = K(P_C, c) = \top$).*
3. *Les pairs P_B, P_A, P_V et P_E auront accès à la variable v renseignant sur la validité des détails de la carte. ($K(P_B, v) = K(P_A, v) = K(P_V, v) = K(P_E, v) = \top$)*
4. *Pour toute association entre un pair p et une variable s non mentionnée par un des cas ci-dessus $K(p, s) = \bot$*

Nous résumons la politique de contrôle d'accès de l'exemple précédent à la Figure 3.2. La politique de contrôle d'accès de chaque pair pouvant être indépendante des accointances de ces derniers, nous n'illustrons pas les les liens d'accointances dans la dernière figure.

Nous rappelons qu'un pair ne pourra raisonner qu'à partir des formules et du vocabulaire auxquelles il a accès. Le vocabulaire accessible par un pair sera toujours un sur-ensemble du

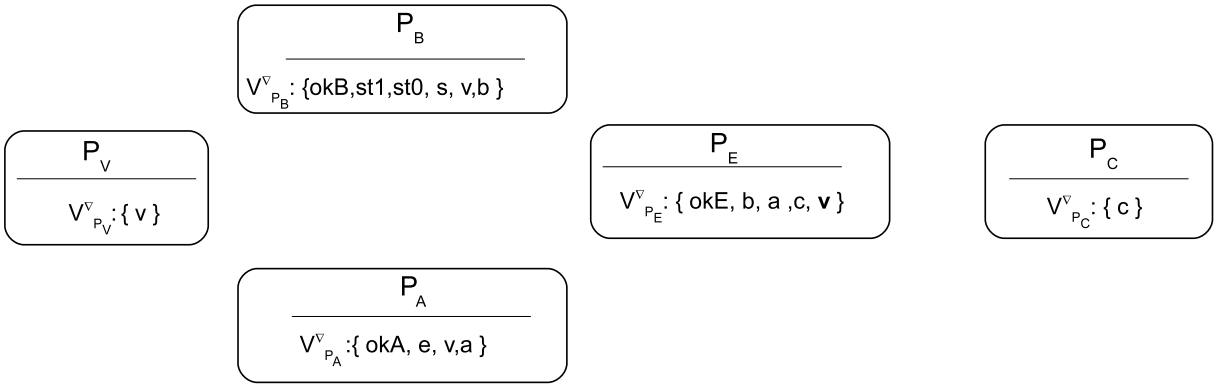


FIGURE 3.2 – Vocabulaire accessible induit par une politique d'accès ex : 3.1

vocabulaire décrivant le comportement de ce pair. A partir de son vocabulaire accessible, un pair ne pourra interagir qu'avec un nombre limité d'autres pairs voisins définis dans le réseau d'accès.

3.1.3 Les descriptions locales des pairs diagnostiqueurs

Le cadre du diagnostic propositionnel à base de cohérence introduit par de Kleer et Reiter [dKMR92] a été appliqué au contexte centralisé. Pour le diagnostic de systèmes distribués, d'autres formalismes tels que : les systèmes à évènements discrets [SSL+96], les automates [SW05], les réseaux de Petri [BHFJ03] ont été préférés en raison de leur expressivité et de leur facilité à représenter des comportements concurrents. Comme nous l'avons remarqué dans l'état de l'art, les travaux de Grastien et al [GARK07] ont montré que tout problème de diagnostic formulé à travers un système à évènements discrets peut être traduit vers un problème de satisfaction d'une formule propositionnelle. De plus leurs expérimentations ont montré que les algorithmes SAT résolvaient de plus grandes instances que les algorithmes traditionnels de diagnostic. En présence d'un problème propositionnalisé, la problématique de diagnostic consiste à rechercher des modèles préférés [Fel10] du système réécrit. Le cadre du diagnostic défini par de Kleer et al [dKMR92] peut donc s'étendre à l'analyse de comportements distribués traduits par une formule propositionnelle. C'est de ce cadre que nous nous sommes inspiré pour décrire les comportements attendus et observés portés à la connaissance de chaque pair diagnostiqueur.

Définition 3.3 (Description d'un pair diagnostiqueur) *Étant donnée une politique d'accès K, la description du sous-système d'un pair diagnostiqueur p se modélise par un tuple $DS_p : \langle V_p, L_p, \Sigma_p \rangle$ t.q :*

1. *V_p représente l'ensemble des variables utilisées pour décrire le comportement surveillé par le pair p ($V_p \subseteq V_p^{\triangledown}$)) t.q. $v \in V_p : K(p, v) = \top$.*
2. *L_p désigne le langage cible formé sur V_p, il est constitué d'un ensemble cohérent de littéraux renseignant sur l'état de fonctionnement du sous-système correspondant à p (littéraux de modes comportementaux).*
3. *Σ_p est une description propositionnelle à partir du vocabulaire V_p, du comportement et des observations du pair p.*

Nous soulignons ici que nous nous intéressons à compiler tous les diagnostics minimaux d'un système distribué. Initialement la description d'un pair diagnostiqueur traduira les comportements attendus et observés d'un sous-système sur une période donnée.

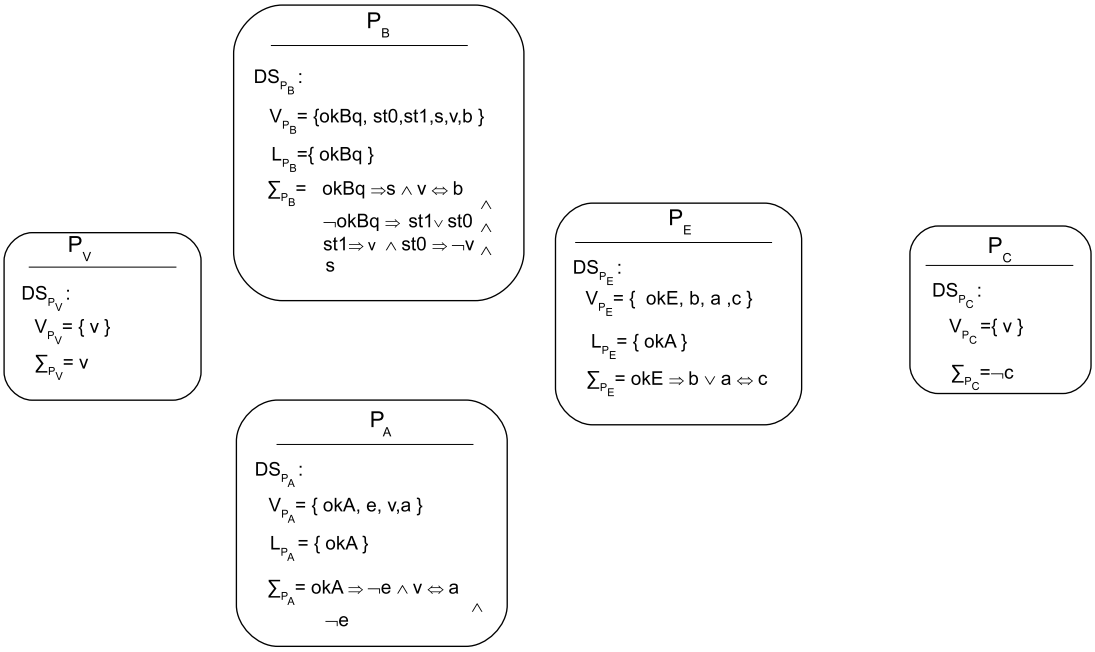


FIGURE 3.3 – Descriptions des comportements attendus et observés de l'exemple jouet

Exemple. 3.2 (pairs diagnostiqueurs du processus de validation de commande) *Nous illustrons à la figure 3.3 la description de chaque pair diagnostiqueur modélisant le comportement attendu et observé d'un service. Chaque pair diagnostiqueur décrit le comportement d'un des services intervenant dans notre exemple jouet. Par rapport aux descriptions introduites dans le chapitre précédent, seule la description du pair P_B (de la Banque) a été enrichi de deux nouvelles variables de mode $st0$ et $st1$. La formule encodant le comportement attendu du service de la banque DS_{P_B} traduit que dans un état normal (okB), la banque donne son accord (b), ssi la carte du client est valide (v) et qu'il est solvable (s). Dans un état anormal, la validité de la carte d'un client est collé à 0 ($st0$) ou à 1 ($st1$). Le comportement observé est modélisé par le prédicat s : le client est solvable. Concernant la description du comportement des autres services, nous nous ramenons à l'exemple introductif du chapitre précédent.*

3.1.4 Système de pairs diagnostiqueurs

A l'aide des définitions de la description des sous-systèmes, du réseau d'accès ou encore du contrôle d'accès nous formalisons la description d'un système de pairs diagnostiqueurs.

Définition 3.4 (Système de pairs diagnostiqueurs) *Un système de pairs diagnostiqueurs est un tuple $S : \langle P, G, K, DS \rangle$ où*

- *P est un ensemble de pairs diagnostiqueurs p,*
- *$G(P, ACQ)$ représente son graphe d'accointances,*
- *$K : P \times Vars \rightarrow \{\top, \bot\}$ modélise la politique d'accès aux variables des pairs,*
- *$DS : \{DS_p : p \in P\}$ représente l'ensemble des descriptions des sous-systèmes initialement modélisées par chaque pair p de P t.q. $DS_p : \langle V_p, L_p, \Sigma_p \rangle$.*

Nous illustrons à la figure 3.4 la description du système de pairs diagnostiqueurs des services de validation de commande ainsi que le scénario anormal observé dans l'exemple 2.11. Chaque pair est représenté par un rectangle aux angles arrondis. À l'intérieur de chaque rectangle nous

décrivons par $V_p^{\triangledown}$ le vocabulaire accessible au pair p et par DS_p la description du comportement attendu et observé par le pair. Le graphe d'accointances est représenté par des liens entre les rectangles. Le vocabulaire accessible et pouvant être partagé entre pairs voisins est noté dans un ensemble précédé de $^{\triangledown}$. Par exemple l'ensemble $^{\triangledown}\{b, v\}$ indique que P_E et P_B pourront s'échanger des formules construites à partir des variables v et b.

En résumé, dans un système de pairs diagnostiqueurs la description locale à chaque pair représente les comportements attendus et observés d'un sous-système. Les pairs sont connectés à travers un graphe d'accointances et suivent une politique de contrôle d'accès. Dans le but de pouvoir expliquer le comportement global d'un système, nous verrons par la suite que chaque pair sera amené à faire évoluer sa description en tenant compte des connaissances reçues du voisinage.

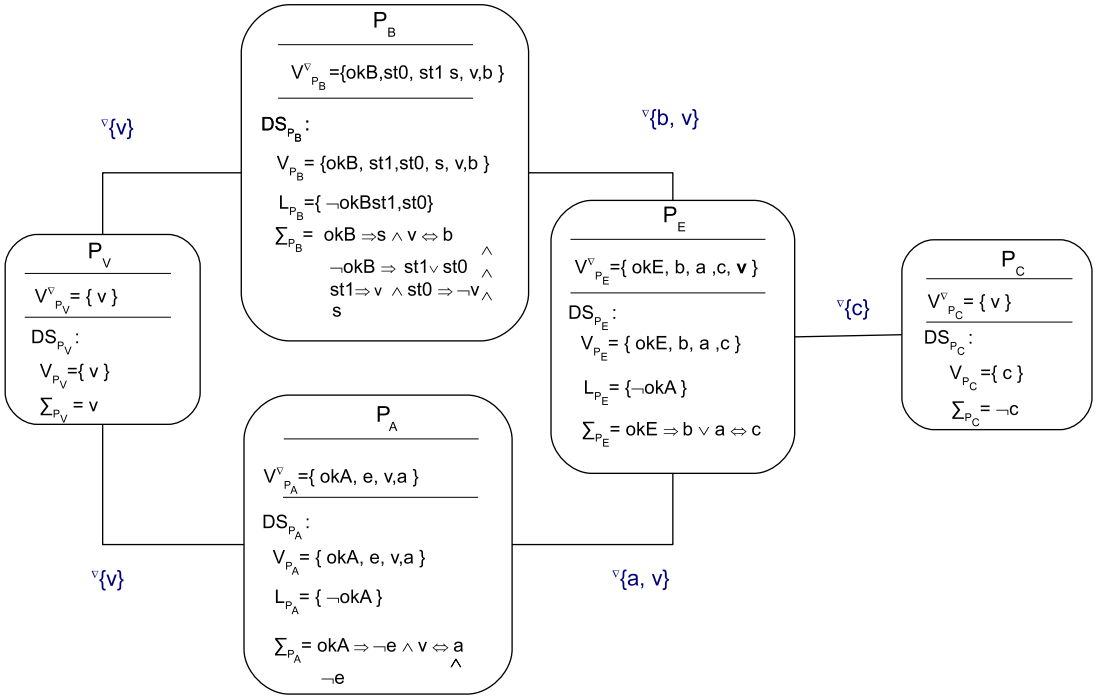


FIGURE 3.4 – Description d'un système de pairs diagnostiqueurs

3.2 Formalisation de la problématique de diagnostic distribué avec contrôle d'accès

Le défi du diagnostic distribué avec contrôle d'accès est de pouvoir expliquer le comportement global d'un système distribué, tout comme l'aurait fait un unique diagnostiqueur, par un ensemble de pairs ayant chacun un accès limité à la description du système global. Dans un premier temps, nous introduisons, à travers les descriptions accessibles, les évolutions d'un système de pairs où chaque pair a pu enrichir sa propre description de nouvelles connaissances en provenance des pairs voisins, tout en respectant une politique d'accès. À partir des descriptions accessibles, nous présentons les diagnostics localement cohérents. Ces explications représentent ce que peuvent calculer chaque pair diagnostiqueur. La problématique de diagnostic distribué consiste à expliquer le comportement global d'un système distribué par un ensemble de diag-

nostics localement cohérents calculés par les pairs. Plus précisément nous cherchons à compiler la description de comportements attendus et observés par des pairs diagnostiqueurs vers une description accessible où :

- tout diagnostic localement cohérent est une explication partielle d'un diagnostic global (correction),
- tout diagnostic global se répartit à travers un ensemble de diagnostics localement cohérents (complétude).

3.2.1 Un pair diagnostiqueur, à quoi a-t-il accès ?

Dans notre problématique nous supposons que chaque pair ne pourra expliquer que la partie de la description du système global à laquelle il a accès. Nous qualifions une telle description de description accessible. Une description accessible par un système représente aussi l'enrichissement de chacune des descriptions initiales des pairs par de nouvelles connaissances déduites par le pair ou en provenance de pairs voisins. Plus formellement nous avons la définition suivante :

Définition 3.5 (Description et système accessibles) *Soit* $S : \langle P, G, K, DS\rangle$ *un système t.q.* $DS = \{DS_p : \langle V_p, L_p, \Sigma_p\rangle : p \in P\}$. *Une description* $DS' : \{DS'_p : \langle V'_p, L'_p, \Phi_p\rangle : p \in P\}$ *est dite accessible par le système* S *ssi pour chaque pair* p, DS'_p *est t.q. :*

$$\forall v \in V'_p : K(p, v) = \top,$$

$$\bigcup_{p\in P} V'_p = \bigcup_{p\in P} V_p,$$

$$\bigcup_{p\in P} L'_p = \bigcup_{p\in P} L_p$$

De même, le système $S' : \langle P, G, K, DS'\rangle$ *est appelé système accessible de* S'.

Dans la définition précédente, nous constatons que pour chaque pair d'une description accessible, le vocabulaire, les formules et le langage cible respectent la politique de contrôle d'accès du pair. Par ailleurs, pour un pair p, Φ_p représente l'évolution de la description locale d'un pair enrichie lors du raisonnement distribué. Φ_p pourra n'avoir ni la même forme, ni la même sémantique que Σ_p. Nous rappelons ici que, compte tenu des mises à jours fréquentes inhérentes aux systèmes dynamiques et à l'éventuelle explosion combinatoire qu'impliquerait la prise en compte de ces mises à jours, nous focalisons la tâche de diagnostic sur une période donnée pour laquelle la description du système et le comportement observé n'évoluent pas. Afin de garantir que le comportement du système global traduit par une description accessible reste inchangé, nous nous assurons que la sémantique de cette dernière est équivalente à celle du système initial.

Définition 3.6 (Description et système accessibles équivalents) *Soient un système* $S : \langle P, G, K, DS\rangle$ *t.q* $DS : \{DS_p : \langle V_p, L_p, \Sigma_p\rangle : p \in P\}$ *et une description accessible* $DS' : \{DS'_p : \langle V'_p, L'_p, \Phi_p\rangle : p \in P\}$. DS" *est dite équivalente à* DS' *ssi pour chaque pair* p *:*

$$\bigwedge_{p\in P} \Phi_p \equiv \bigwedge_{p\in P} \Sigma_p$$

De même, le système accessible $S' : \langle P, G, K, DS'\rangle$ *sera dit équivalent à* S.

Une description accessible équivalente à un système initial représente une réécriture de la description initiale de ce système vers une description où la sémantique globale restera inchangée mais où les descriptions locales pourront être enrichies par un raisonnement distribué. Pour une description accessible DS' équivalente à DS, chaque formule locale Φ_p pourra avoir une forme et une sémantique différente de Σ_p.

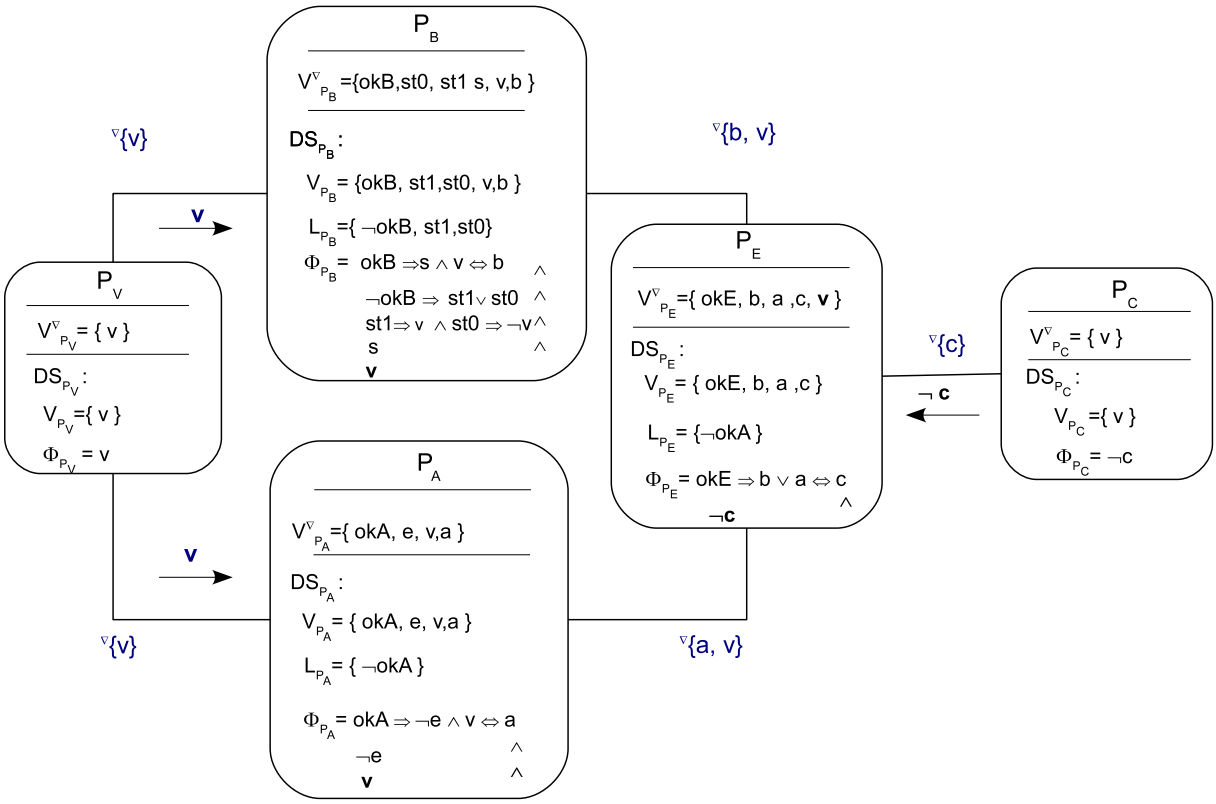


FIGURE 3.5 – Système accessible équivalent à la Figure 3.4

Exemple. 3.3 *À la Figure 3.5 nous représentons un système accessible équivalent au système de pairs diagnostiqueurs introduit Figure 3.4. Par rapport à la description initiale, les pairs P_A et P_B ont enrichi leurs descriptions de l'observation v reçue du pair P_V. De même le pair P_E a enrichi sa description de l'observation $\neg c$ en provenance de P_C. Ici la formule Φ_p de chaque pair conserve la même forme (CNF)que Σ_p. Les pairs qui ont enrichi leurs descriptions d'observations des pairs voisins ont fait évoluer la sémantique de leurs descriptions locales initiales (l'ensemble de modèles locaux satisfaisant la description locale n'est plus la même).*

À travers les descriptions accessibles nous avons modélisé le comportement du système distribué auquel chaque pair a accès. Dans la section suivante, nous modélisons ce que peut expliquer chaque pair diagnostiqueur à partir des descriptions accessibles.

3.2.2 Que peut expliquer un pair à partir d'une description accessible ?

Étant donné un système de pairs diagnostiqueurs avec contrôle d'accès, il est évident de supposer que chaque pair est en mesure d'expliquer le comportement d'une description locale à laquelle il a accès. Nous qualifions de telles explications de diagnostics locaux.

Définition 3.7 (Diagnostic local d'un pair) *Étant donné un système accessible* $S : \langle P, G, K, DS\rangle$ *t.q.* $DS : \{DS_p : \langle V_p, L_p, \Phi_p\rangle : p \in P\}$*, un ensemble de littéraux cohérents* $\Delta_p \subseteq L_p$ *est un diagnostic local cohérent de* DS_p *ssi :*

$$\Delta_p \wedge \overline{L_p \setminus \Delta_p} \wedge \Phi_p \not\models \bot \tag{3.1}$$

Par la définition précédente, on code donc chaque diagnostic local d'un pair par l'ensemble de littéraux de mode qui prennent la valeur vraie par ce diagnostic. On ne fait pas l'hypothèse que ces littéraux soit sémantiquement disjoints comme dans [dKW89], par exemple $\{\neg okB, st1, st0\}$.

Exemple. 3.4 *Dans la Figure 3.6 nous représentons les diagnostics locaux de la description accessible représentant le système initial de pairs (Figure 3.4). Dans cet exemple, chaque formule encodant une description locale accessible à un pair est représentée par l'ensemble de modèles qui la satisfont (c-à-d une contrainte extensionnelle).*

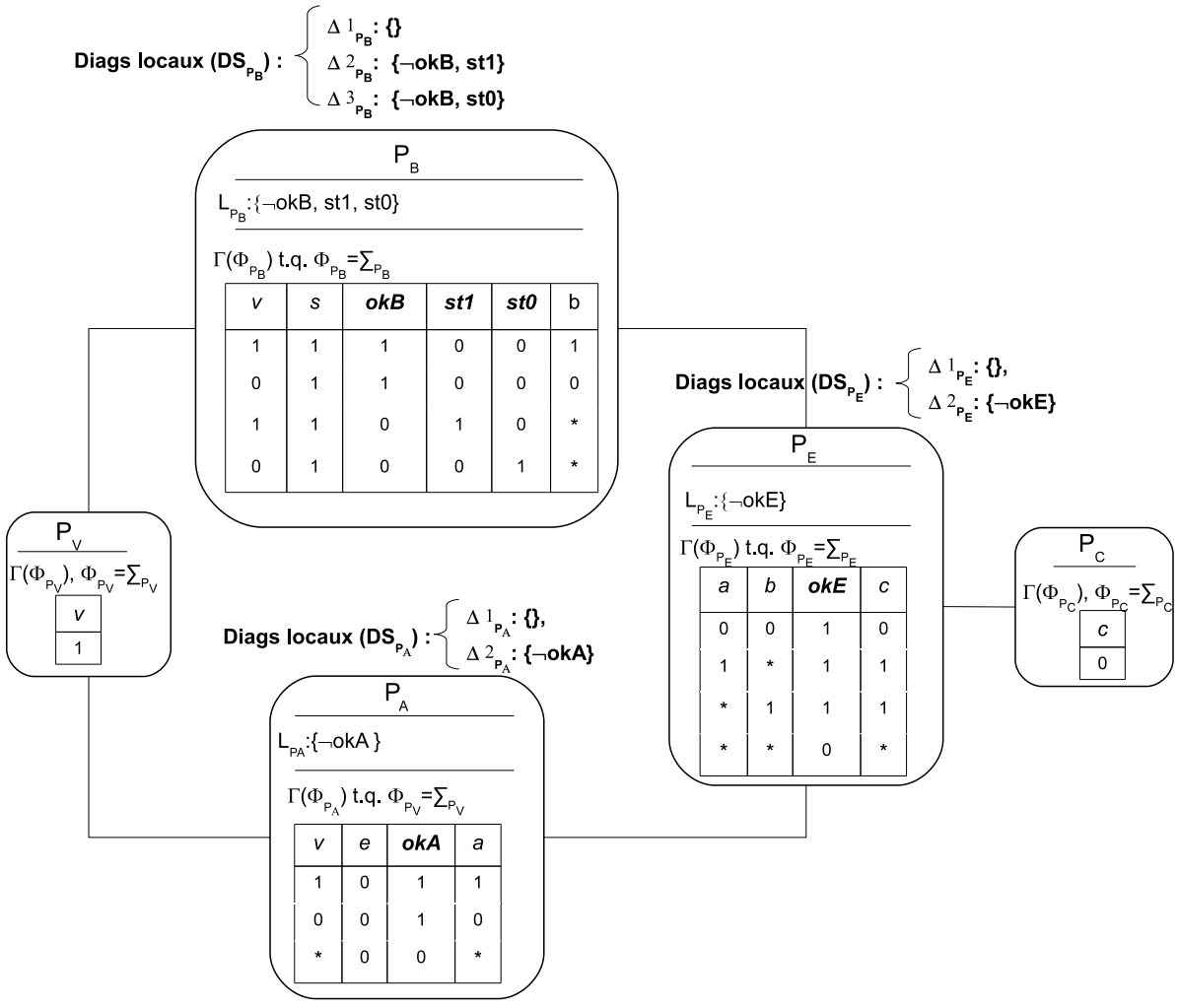


FIGURE 3.6 – Diagnostics locaux

De plus, un pair sera aussi en mesure de vérifier que ses diagnostics locaux sont cohérents avec les diagnostics des pairs voisins. Pour constater cela nous nous rapprochons les notions de diagnostics et modèles locaux d'une formule par l'opération de restriction.

Étant donnée une formule Φ et un sous-ensemble V de ses variables, la restriction de Φ sur V est notée par $restrict(\Phi, V)$ et correspond à appliquer récursivement sur Φ l'opération de décomposition de Shannon sur toutes les variables x de Φ qui n'apparaissent pas dans V. Cette opération est connue sous le nom de $< \ forgetting \ >$ en compilation de bases de connaissances, et est bien connue pour être un problème NP-Difficile. Cependant, quand Φ est un monôme (c-à-d une conjonction de littéraux) I, l'opération de projection est réduite à la restriction syntaxique sur le vocabulaire du monôme I.

Définition 3.8 (Restriction d'un monôme sur un vocabulaire, des littéraux) *La restriction d'un monôme I sur un ensemble de variables V se définit par :*

$$restrict(I, V) = I' \ t.q \ I \in \Pi \ t.q. \ vars(I') = vars(I) \cap V$$

La restriction de I sur un ensemble de littéraux L cohérent se définit par :

$$restrict(I, L) = I \cap L \ t.q. \ I \in \Pi\}$$

Nous rappelons qu'un modèle est une interprétation (c-à-d une interprétation de valeurs aux variables) satisfaisant une formule (voir Définition 2.8). Dans la suite lorsqu'il n'y aura pas d'ambiguité nous représenterons un modèle par le monôme qu'il satisfait.

Maintenant, en reprenant la définition de diagnostic local, nous constatons qu'il peut aussi être vu comme une assignation des variables du langage cible cohérente avec la description du pair. Ainsi, un diagnostic local représente un modèle partiel de la description du pair. Nous déduisons donc la remarque suivante :

Remarque 3.1 (modèle d'un diagnostic local) *Soit $DS_p : \langle V_p, L_p, \Phi_p \rangle$ une description accessible locale à un pair p. Pour tout diagnostic local Δ_p de Φ_p, il existe un modèle m_p satisfaisant Φ_p et prolongeant Δ_p.*

$$\forall \Delta_p, \exists m_p, \ t.q. \ restrict(m_p, L_p) = \Delta_p \tag{3.2}$$

Dans la Figure 3.6 nous pouvons vérifier que les diagnostics locaux se retrouvent à partir de l'expression sur le langage cible des modèles locaux des descriptions des pairs. Lorsqu'il n'y aura pas d'ambiguïté, nous considérerons un modèle ou un diagnostic comme une conjonction de littéraux ou encore une assignation de valeurs aux variables. Prenons la première ligne de DS_{Bq} décrivant le modèle $m = \{v = Vrai, s = Vrai, okB = Vrai, st1 = Faux, st0 = Faux, b = Vrai\}$. En sachant que le langage cible est $L_{Bq} = \{\neg okBq, st0, st1\}$. $restrict(m, L_{Bq}) = \{\}$ ce qui correspond au diagnostic local $\Delta^1_{Bq} = \{\}$. De même on remarquera que $restrict(m, vars(L_{Bq})) = \Delta^1_{Bq} \cup \overline{L_{Bq} \setminus \Delta^1_{Bq}} = \{\} \cup \{okBq, \neg st0, \neg st1\}$.

Inversement nous constatons qu'à partir de tout modèle local d'une formule il sera aussi possible de retrouver un diagnostic local de la description d'un pair.

Remarque 3.2 (diagnostic d'un modèle local) *Soit $DS_p : \langle V_p, L_p, \Phi_p \rangle$ la description accessible locale à un pair p. Tout modèle local m_p satisfaisant Φ_p contient un diagnostic local Δ_p exprimé à travers les assignations de variables du langage cible.*

$$\forall m_p, \exists \Delta_p, \ t.q. \ restrict(m_p, L_p) = \Delta_p \tag{3.3}$$

Maintenant que nous avons constaté la correspondance entre diagnostic et modèle, on notera par la remarque suivante qu'il sera possible à deux pairs de vérifier que leurs modèles locaux sont cohérents.

Remarque 3.3 (modèles cohérents) *Deux modèles locaux m_p, $m_{p'}$ des pairs voisins p, p' sont cohérents ssi leurs assignations sur leur vocabulaire en commun est identique :*

$$restrict(m_p, V_p \cap V_{p'}) = restrict(m_p, V_p \cap V_{p'}) \tag{3.4}$$

À partir de cette remarque nous déduisons que chaque pair pourra vérifier que ses modèles locaux sont cohérents avec des modèles des pairs voisins uniquement à partir du vocabulaire partagé avec ces derniers.

Définition 3.9 (Modèle, formule, localement cohérents) *Soit $S : \langle P, G, K, DS\rangle$ un système de pairs.*

- *Un modèle m_p local à un pair p est dit localement cohérent ssi pour toute arête (p, p') de ACQ, il existe un modèle $m_{p'}$ local à p' t.q. m_p et $m_{p'}$ sont cohérents.*
- *Un système sera dit localement cohérent ssi les modèles de chacune de ses descriptions locales sont localement cohérents.*

La notion de cohérence locale introduite dans la définition précédente se ramène à la notion d'arc cohérence utilisée en CSP. En effet, étant donné un réseau de contraintes binaires, une valeur vx d'une variable x est dite arc cohérente ssi pour toute variable y liée à x par une contrainte $C_{x,y}$ il existe une valeur vy cohérent avec vx (c-à-d qui satisfait la contrainte $C_{x,y}$). Si on associe chaque description Φ_p de notre contexte distribué à une variable x d'un réseau de contraintes, chaque modèle de Φ_p à une valeur du domaine de x, chaque lien du graphe d'accointances à une contrainte binaire satisfaite par les valeurs provenant de modèles cohérents, alors un modèle localement cohérent d'un système de pairs correspond à une valeur de variable arc cohérente d'un CSP. L'arc cohérence de CSP est une propriété centrale dans la résolution de CSP, elle permet de réduire la taille du domaine de chaque variable sans pour autant écarter des solutions. Elle a été très étudiée à la fois dans le cadre centralisé [Mac77, BC93] mais aussi dans le cadre distribué [ND98, Ham02]. Par exemple l'algorithme distribué DisAC-9 d'Hamadi [Ham02] permet de satisfaire l'arc cohérence par un nombre de messages optimal (c-à-d en $O(n^2d^3)$ où n représente le nombre de variables et d la taille d'un domaine). Nous verrons dans le chapitre 6 que la structure dans laquelle nous ressituons notre approche nous permettra de satisfaire l'arc cohérence par un simple parcours arborescent.
Étant donné que tout pair a la possibilité d'échanger des formules définies sur le vocabulaire partagé avec son voisinage, et mettre à jour sa description nous déduisons la remarque suivante : *Quelque soit la politique de contrôle d'accès, un système de pairs sera toujours en mesure de vérifier qu'il est localement cohérent.*
En revenant à notre problématique de diagnostic, et à partir de la relation qu'il existe entre diagnostics et modèles nous définissons les diagnostics localement cohérents comme suit :

Définition 3.10 (Diagnostic localement cohérent) *Un diagnostic local Δ_p est dit localement cohérent ssi il existe un modèle localement cohérent m_p qui le prolonge : $restrict(m_p, L_p) = \Delta_p$.*

Ainsi, de même que pour les modèles locaux, nous avons la remarque suivante :

Quelque soit la politique de contrôle d'accès, un système de pairs sera toujours en mesure de vérifier que tous diagnostics locaux sont localement cohérents.

Exemple. 3.5 *Nous illustrons à la Figure 3.7 l'ensemble des diagnostics localement cohérents de la description accessible représentant le système de pair initial (Figure 3.4). Les diagnostics*

localement cohérents représentent un sous-ensemble des diagnostics locaux (Figure 3.6). Nous indiquons par un éclair les modèles locaux qui ne sont pas cohérents avec au moins un autre modèle local de chacune des descriptions des pairs voisins. À partir des assignations sur le langage cible des modèles localement cohérents (non annotés par un éclair), nous déduisons les diagnostics localement cohérents. Par exemple $\{\neg okB, st0\}$ *est un diagnostic local de* DS_{P_B} *(cf Figure 3.6) sans pour autant en être un diagnostic localement cohérent (cf Figure3.7). En effet l'ensemble des modèles locaux de* DS_{P_B} *pouvant satisfaire* $\neg okB$ *et* $st0$ *sont représentés par la dernière ligne de la table* $\Gamma(\Phi_{P_B})$*. Or pour cette ligne tous les modèles ont l'assignation* $v = 0$ *qui n'est pas cohérente avec l'assignation* $v = 1$ *de* DS_{P_V}*. Par la définition 3.10* $\neg okB, st0$ *n'est pas un diagnostic localement cohérent.*

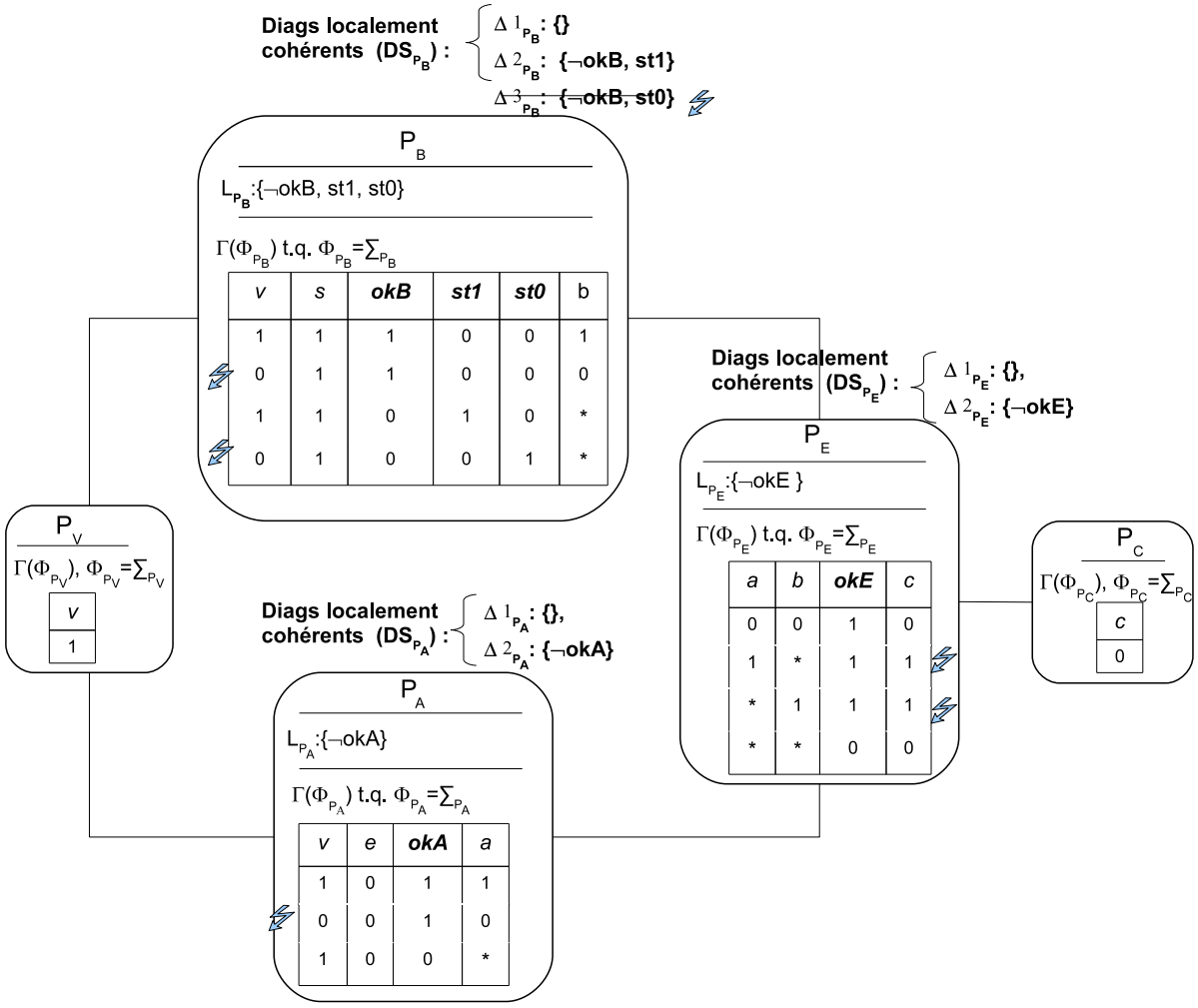


FIGURE 3.7 – Diagnostics localement cohérents

En résumé, en raison du contrôle d'accès et du graphe d'accointances d'un système, un pair ne sera autorisé à expliquer qu'une partie de la description du système global à laquelle il a accès. Par les diagnostics localement cohérents nous avons formalisé ce que chaque pair peut calculer à partir de descriptions accessibles. Maintenant nous nous intéresserons à formaliser la problématique de diagnostic distribué.

3.2.3 Problématique : Comment expliquer le comportement global d'un système par les diagnostics accessibles localement cohérents des pairs ?

L'objectif du diagnostic distribué est de pouvoir expliquer le comportement global d'un système distribué par un ensemble de pairs tout comme l'aurait fait un unique pair diagnostiqueur qui aurait la connaissance globale du système. Si un tel diagnostiqueur omniscient existait, ses explications prendraient la forme de diagnostic globaux.

Définition 3.11 (Diagnostic global d'un système) *Un diagnostic global $\Delta^G \subseteq \bigcup_{p\in P} L_p$ d'un système $S : \langle P, G, K, DS\rangle$ est une explication cohérente du comportement de S exprimée à travers l'union des langages cibles des pairs c-à-d :*

$$\Delta^G \cup \overline{\bigcup_{p\in P} L_p \setminus \Delta^G} \bigwedge_{p\in P} DS_p \not\models \bot \tag{3.5}$$

Malheureusement, compte tenu de la politique d'accès, il sera bien souvent impossible à un pair diagnostiqueur d'avoir la connaissance de la description globale du système. De même, il lui sera aussi parfois interdit de centraliser la totalité du langage cible. Toutefois, afin de pouvoir contourner l'obstacle que nous pose la politique d'accès, nous chercherons par le calcul des diagnostics distribués à répartir un diagnostic global à travers l'ensemble des pairs.

Définition 3.12 (Diagnostic distribué d'un système) *Un diagnostic distribué $\Delta^D = \{\Delta_p \subseteq L_p : p \in P\}$ d'un système $S : \langle P, G, K, DS\rangle$ est un diagnostic global de S réparti à travers le système et exprimé à travers le langage cible de chacun des pairs t.q.*

$$\bigwedge_{\Delta_p\in\Delta^D} \Delta_p \cup \overline{L_p \setminus \Delta_p} \bigwedge_{p\in P} DS_p \not\models \bot \tag{3.6}$$

- *Un diagnostic global Δ^G de S est dit induit par le diagnostic distribué $\Delta^D = \{\Delta_p : p \in P\}$ ssi $\Delta^G = \bigcup_{\Delta_p\in\Delta^D} \Delta_p$.*
- *Un diagnostic distribué $\Delta^D = \{\Delta_p : p \in P\}$ est minimal si pour le diagnostic global Δ^G qu'il induit et pour tout autre diagnostic global Δ'^G, $\Delta'^G \subseteq \Delta^G$ implique que $\Delta'^G = \Delta^G$.*

À partir de cette définition et parce qu'il ne sera pas toujours possible pour un pair de regrouper les diagnostics globaux d'un système, nous nous assurerons que tous les diagnostics distribués d'un système pourront être retrouvés à travers un ensemble de diagnostics localement cohérents.

Propriété 3.1 (Système complet pour le diagnostic distribué) *Un système accessible $S' : \langle P, G, K, DS'\rangle$ est dit complet pour le diagnostic distribué de S ssi tout diagnostic distribué $\{\Delta_p : p \in P\}$ de S est un ensemble de diagnostics localement cohérents de S'.*

De même et aussi parce qu'il ne sera pas toujours possible pour un pair de regrouper les diagnostics globaux d'un système nous nous assurerons que tout diagnostic localement cohérent pourra se prolonger par un diagnostic distribué.

Propriété 3.2 (Système correct pour le diagnostic distribué) *Un système accessible $S' : \langle P, G, K, DS'\rangle$ est dit correct pour le diagnostic distribué de S ssi tout diagnostic local d'un pair $p \in P$ de S' est localement cohérent et peut être prolongé par un diagnostic global induit par un diagnostic distribué de S.*

De par la correspondance étroite existante entre cohérence locale des diagnostics et l'arc cohérence en CSP, nous tenons à préciser que même si un système pourra garantir la cohérence locale de ses diagnostics (et en quelque sorte l'arc cohérence) cela ne suffira pas à garantir un diagnostic distribué correct. Ainsi, bien que les approches vérifiant l'arc cohérence distribuée nous seront nécessaires, elles ne nous seront pas suffisantes.

En définitive, pour un système distribué garantissant un diagnostic distribué correct et complet on sera en mesure d'expliquer le comportement global du système par un ensemble de pairs tout comme l'aurait fait un unique pair superviseur qui aurait la connaissance de toutes les descriptions. Toutefois, nous noterons qu'à la fin du calcul de ses diagnostics globaux, un pair superviseur saurait répondre dans un temps quasi immédiat (c-à-d au pire linéaire dans le nombre de diagnostics) si oui ou non une explication du comportement du système est un diagnostic (c-à-d si elle est cohérente avec la description du système global). Or un système garantissant un diagnostic distribué correct et complet ne garantit pas de pouvoir mettre en avant, par la décision de chaque pair, un diagnostic distribué en un temps raisonnable. Nous chercherons donc en plus à garantir qu'un système est compilé pour le diagnostic distribué.

Proposition 3.1 (Système compilé pour le diagnostic distribué) *Soient S un système et S' un système accessible à S, S' est dit compilé pour le diagnostic distribué de S ssi*

1. *S' est correct et complet pour le diagnostic distribué de S,*
2. *tout pair est en mesure de vérifier qu'un diagnostic localement cohérent fait partie d'un diagnostic distribué minimal par un algorithme distribué polynomial en temps et en nombre de messages par rapport à la taille des descriptions locales de S'.*

Exemple. 3.6 (Système compilé pour le diagnostic distribué) *La Figure 3.8 représente chaque formule de la description accessible Figure 3.5 par son ensemble de modèles. Nous rappelons que pour la description accessible Figure 3.5 les pairs P_B et P_A ont reçu l'observation v de P_V, P_E a reçu $\neg c$ de P_C. Ce système accessible garantit un diagnostic correct et complet du système de pairs initial (Figure 3.4). Nous pouvons vérifier que tous les diagnostics locaux de chacune des formules $\{\Phi_{Ag}, \Phi_{Bq}, \Phi_{Es}\}$ sont localement cohérents. Ils font aussi parti d'un des deux diagnostics globaux minimaux du système initial (Figure 3.4) qui sont $\{\neg okB, st1, \neg okA\}$ (c-à-d Diag Distrib1) ou $\{\neg okE\}$ (c-à-d Diag Distrib2). En suivant les lignes discontinues qui prolonge un modèle localement cohérent par un autre modèle d'un autre pair il est possible de déduire les diagnostics distribués Diag Distrib1 et Diag Distrib2. Nous remarquerons que si effectivement chacune des formules $\{\Phi_{Ag}, \Phi_{Bq}, \Phi_{Es}\}$ est représentée par son ensemble de modèles localement cohérents alors ce système est aussi un système compilé pour le diagnostic distribué. Par contre si chacune des formules $\{\Phi_{Ag}, \Phi_{Bq}, \Phi_{Es}\}$ est sous une forme normale conjonctive quelconque comme c'est le cas Figure 3.5, ce système n'est pas compilé pour le diagnostic distribué. En effet compiler ne serait-ce qu'un diagnostic local est une tâche NP-Difficile. Nous noterons toutefois que la compilation de modèles locaux d'une formule nous sert à illustrer les différents aspects de la problématique de diagnostic distribué. En pratique calculer l'ensemble des modèles d'une description même locale est souvent intraitable. Dans la suite chapitre 6, nous proposerons des représentations locales plus succinctes pour le diagnostic distribué.*

Avant de proposer un algorithme distribué permettant de résoudre notre problématique de diagnostic distribué, nous cherchons avant tout à caractériser les systèmes garantissant un diagnostic distribué correct et complet. Dans ce cadre, nous montrons ici que les systèmes accessibles équivalents à un système observé initial permettent de garantir la propriété de complétude du diagnostic distribué.

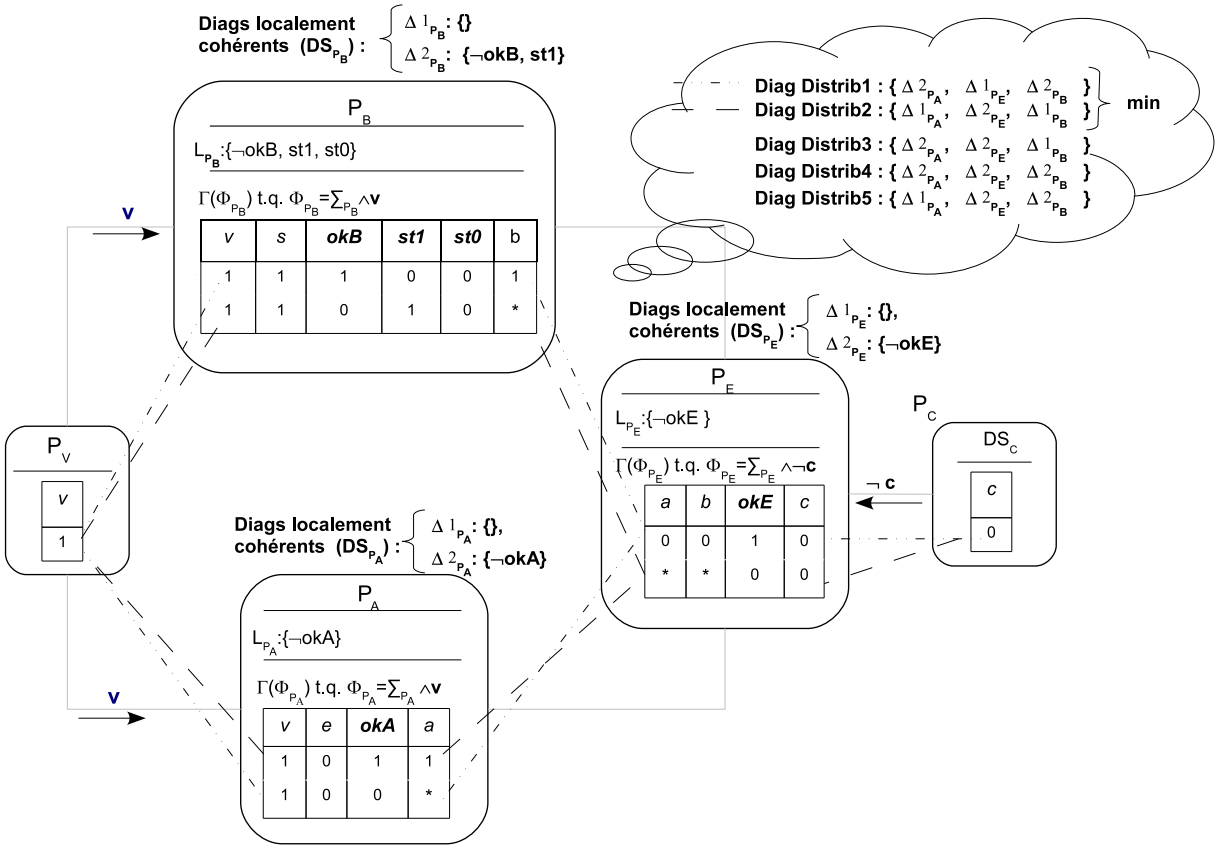


FIGURE 3.8 – Diagnostics distribués d'un système accessible

Proposition 3.2 (Un système accessible équivalent suffit au diagnostic distribué complet) *Soit S un système de pairs diagnostiqueurs et S' un système accessible et équivalent à S, alors S' garantit un diagnostic distribué complet pour S.*

Preuve. Le langage cible et la sémantique de l'union des descriptions locales d'un système accessible S' restent identiques à ceux du système initial S. Ainsi tout diagnostic distribué $\Delta^D = \{\Delta_p : p \in P\}$ de S est aussi un diagnostic distribué de S'. Considérons L'_p le langage cible de p dans S' et $\Delta'_p = restrict(\bigcup_{\Delta_p \in \Delta^D} \Delta_p, L'_p)$ la restriction à ce langage du diagnostic global induit par Δ^D. S et S' étant sémantiquement équivalents, alors Δ'_p est cohérent avec tout sous-ensemble des descriptions de S'. En particulier Δ'_p est cohérent avec chacune des descriptions des pairs voisins à p dans S' donc localement cohérent dans S'. □

Contrairement à la propriété de complétude, caractériser les systèmes garantissant un diagnostic distribué correct est plus compliqué. C'est ce que nous ferons dans le chapitre suivant.

Chapitre 4

Caractérisation graphique des systèmes garantissant un diagnostic distribué correct

Le but de la caractérisation que nous présentons dans chapitre est d'identifier les systèmes accessibles qui garantissent un diagnostic distribué correct. Nous rappelons que lors d'un raisonnement distribué, chaque pair pourra échanger des formules avec ses voisins. Afin de respecter la politique de confidentialité, les formules échangées entre deux pairs voisins seront exprimées à partir de variables accessibles communes aux ces deux pairs.

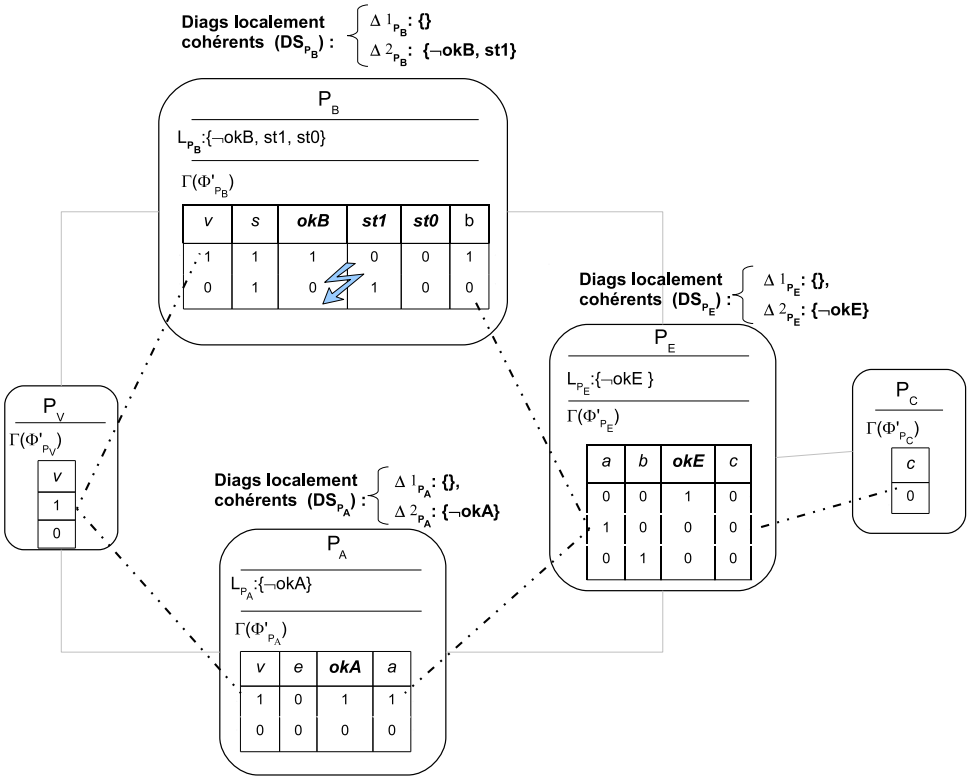


FIGURE 4.1 – Système ne garantissant pas le diagnostic distribué correct

C'est grâce aux échanges de formules l'échange de formules que chacun des pairs pourra faire enrichir sa description et augmenter son vocabulaire. Nous appelons le schéma d'un système, la répartition du vocabulaire de chaque pair à travers le réseau d'accointances. Les échanges de formules entre les pairs feront aussi évoluer le schéma d'un système. Cependant, à cause de la politique de confidentialité, il se pourra qu' aucun pair n'ait la connaissance du schéma du système. Par la caractérisation graphique nous entendons identifier les schémas qui, quel que soit le système modélisé, permettent de garantir un diagnostic distribué correct.

Pour mieux comprendre la problématique de la caractérisation graphique, considérons le système présenté à la Figure 4.1. Le vocabulaire utilisé pour la description de chacun des pairs et le réseau des accointances sont identiques à ceux utilisé par système illustré à la Figure 3.8. Ces deux systèmes ont le même schéma (cf. Figure 4.4). Seuls les modèles satisfaisant les formules de chaque pair changent. Précédemment, nous avons remarqué que pour le système présenté à la Figure 3.8 tous les diagnostics localement cohérents font partie d'un diagnostic global. Dans la la Figure 4.1 ce n'est pas le cas. Considérons le diagnostic $\Delta^1_{P_B}$. Si l'on cherche à prolonger le modèle dont $\Delta^1_{P_B}$ est issu (cf. ligne pointillée) c'est possible localement avec les pairs P_E et P_V. $\Delta^1_{P_B}$ est donc un diagnostic localement cohérent. Cependant, il n'existe pas de diagnostic global contenant $\Delta^1_{P_B}$. Le schéma présenté à la Figure 4.4, modélise à la fois un système correct pour le diagnostic distribué (cf. Figure 3.8) et un système qui ne l'est pas (cf. Figure 4.1). Ce schéma ne garantit pas un diagnostic distribué correct pour tous les systèmes qu'il modélise, il n'est donc pas correct pour le diagnostic distribué. Dans la suite, nous caractérisons les schémas des systèmes nous permettant de garantir un diagnostic distribué correct.

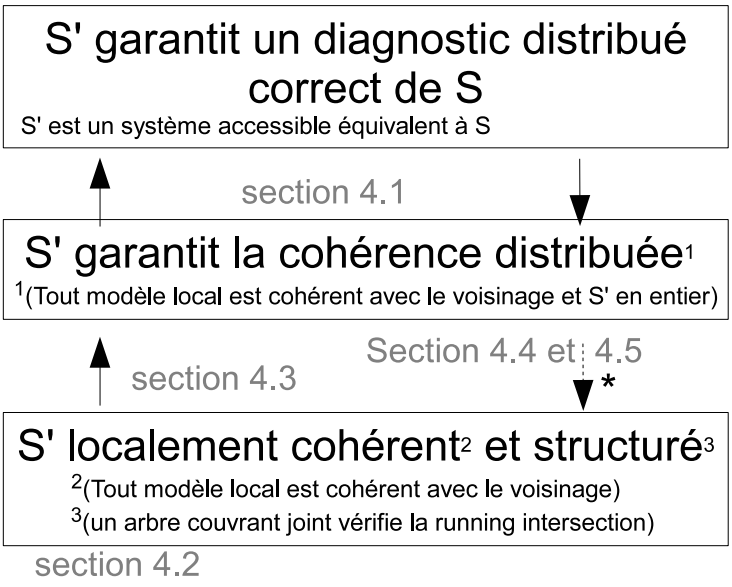


FIGURE 4.2 – Caractérisation graphique des systèmes garantissant un diagnostic distribué correct

Nous résumons à travers la Figure 4.2 le plan général qui permet de guider la caractérisation. Dans un premier temps à la section 4.1, nous simplifions la recherche des systèmes garantissant un diagnostic distribué correct par la recherche de systèmes vérifiant la cohérence distribuée. Un système vérifie la cohérence distribué si tous les modèles locaux d'un pair sont localement cohérents et peuvent se prolonger par un modèle du système global. Dans la section 4.2, nous présentons différentes propriétés nécessaires à la caractérisation graphique des systèmes garantissant la cohérence distribuée. Dans ce cadre, nous définissons formellement le schéma d'un

système. Ensuite, nous focalisons notre étude sur les schémas joints, ce sont des schémas où les pairs contenant une même variable forment un graphe connexe. Ces schémas sont à la base de la caractérisation graphique de la cohérence distribuée. Nous montrons, dans la section 4.3, que les systèmes structurés, c-à-d , les systèmes dont le schéma contient un arbre couvrant joint suffisent à garantir la cohérence distribuée. Cette propriété étend un résultat bien connu du domaine des CSP et de Base de Données. Nous montrons par ailleurs, dans la section 4.4 que l'équivalence entre cohérence locale et cohérence globale ne peut être garantie par un schéma non joint. Enfin, notre dernière propriété, qui constitue l'un des résultats majeurs de cette caractérisation, montre que dans le cas des systèmes d'interactions, c-à-d , les systèmes où les pairs contenant une même variable forment une clique), seuls les systèmes structurés garantissent la cohérence distribuée.

4.1 Du diagnostic à la cohérence distribuée et vice versa

Le but de cette section est de montrer que caractériser les systèmes garantissant un diagnostic distribué correct revient à caractériser des systèmes garantissant la cohérence distribuée, c-à-d , les systèmes où tout modèle d'une description locale à un pair fait aussi partie d'un modèle de la description globale du système.

Comme nous le montre la Figure 4.3, on passe par une étape intermédiaire où l'on fait correspondre un système accessible S' garantissant un diagnostic distribué correct d'un système S, au système S' garantissant que ses diagnostics locaux sont globalement cohérents avec S'. La différence est subtile et nous permet de concentrer sur les seules propriétés des systèmes accessibles S'. Pour cela nous faisons l'hypothèse qu'un système accessible S' est équivalent au système initial S dont il est issu.

Comme l'illustre la Figure 4.3, nous montrons que caractériser les systèmes accessibles qui garantissent un diagnostic distribué correct, revient à caractériser les systèmes où tout diagnostic local peut se prolonger en un diagnostic global, c-à-d les systèmes où tous les diagnostics locaux sont globalement cohérents. Ensuite, nous montrons que caractériser les systèmes où tous les diagnostics locaux sont globalement cohérents, revient à caractériser les systèmes où tout modèle local se prolonge en un modèle du système globale, c-à-d les systèmes où tous les modèles locaux sont globalement cohérents.

Dans la suite, afin de garantir que tout diagnostic global d'un système S soit aussi un diagnostic global d'un système accessible S', nous supposons que le système S' est équivalent à S. Nous rappelons que l'équivalence entre systèmes n'impose pas que les descriptions locales à chaque pair restent inchangées. Au contraire, les descriptions locales auront pu s'enrichir de nouvelles connaissances. De même nous rappelons qu'un diagnostic local signifie un diagnostic local localement cohérent avec les descriptions des accointances de son pair.

4.1.1 D'un diagnostic distribué correct à la cohérence globale des diagnostics locaux

Cette sous-section est une étape intermédiaire et montre que caractériser les systèmes S' accessibles par S où les diagnostics localement cohérents de S' peuvent être prolongés par des diagnostics globaux de S, revient à caractériser les systèmes accessibles S' équivalents à S où les diagnostics localement cohérents de S' se prolongent en ses diagnostics globaux de S' (et donc de S). Dans ce but, nous précisons tout d'abord ce que nous entendons par la cohérence globale des diagnostics locaux.

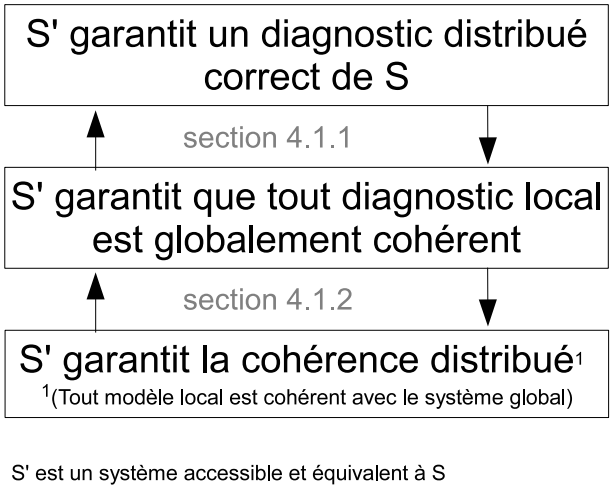


FIGURE 4.3 – Du diagnostic distribué à la cohérence distribuée

Définition 4.1 (Diagnostic local globalement cohérent) *Soit $S' : \langle P, G, K, DS' \rangle$ un système accessible équivalent au système de pairs S. Un diagnostic Δ_p local à p est dit globalement cohérent ssi il est cohérent avec l'union des descriptions des pairs du système.*

À partir de cette définition nous mettons en relation la cohérence globale de diagnostics locaux et les systèmes accessibles garantissant un diagnostic distribué correct de S.

Propriété 4.1 (Diagnostic locaux globalement cohérents et diagnostic distribué correct) *Soit $S' : \langle P, G, K, DS' \rangle$ un système accessible localement cohérent équivalent au système S. S' est correct pour le diagnostic distribué de S ssi tout diagnostic local de S' est globalement cohérent.*

Preuve. (=>) Soit Δ'_p un diagnostic local d'un système S' correct pour le diagnostic distribué de S alors Δ'_p est inclus dans un diagnostic global Δ^G de S. Puisque S' et S sont équivalents alors tout diagnostic global de S est aussi un diagnostic global de S'. Δ'_p est donc inclus dans un diagnostic global de S', Δ'_p est globalement cohérent.
(<=) Soit Δ'_p un diagnostic local globalement cohérent dans S'. Il existe donc un diagnostic global Δ'^G de S' qui le prolonge. S' et S sont équivalents alors tout diagnostic global de S est aussi un diagnostic global de S'. Δ'_p se prolonge donc par un diagnostic global de S. □

À l'aide de cette dernière propriété nous déduisons que caractériser les systèmes accessibles S' équivalents à un système initial S et corrects pour le diagnostic distribué de S, revient simplement à caractériser les systèmes localement cohérents où tout diagnostic local est globalement cohérent.

4.1.2 De diagnostics locaux globalement cohérents à la cohérence distribuée

Dans cette sous-section nous repartons du résultat intermédiaire de la sous-section précédente et faisons correspondre les systèmes où les diagnostics locaux sont globalement cohérents aux systèmes où les modèles locaux sont globalement cohérents. Nous précisons ce que nous entendons par modèle local globalement cohérent.

Définition 4.2 (modèle, sous-système globalement cohérents) *Soit $S' : \langle P, G, K, DS' \rangle$ un système accessible équivalent au système de pairs S :*

1. *un modèle m_p local à p est dit globalement cohérent ssi il est cohérent avec l'union des descriptions des pairs du système.*
2. *un sous-système est dit globalement cohérent ssi tous les modèles locaux qui le satisfont sont globalement cohérents.*

À l'aide de cette définition nous formalisons ce que nous entendons par la cohérence distribuée.

Définition 4.3 (Cohérence distribuée) *Un système de pairs localement cohérent $S : \langle P, G, K, DS\rangle$ vérifie la cohérence distribuée lorsque tous ses modèles locaux sont globalement cohérents.*

Dans un premier temps nous rapprochons les modèles d'une description des diagnostics globalement cohérents par la propriété suivante.

Propriété 4.2 (La cohérence distribuée suffit au diagnostic distribué correct) *Un système vérifiant la cohérence distribuée vérifie aussi que ses diagnostics locaux sont globalement cohérents.*

Preuve. Soit $DS : \{DS_p : \langle V_p, L_p, \Phi_p\rangle : p \in P\}$ une description accessible d'un système $S : \langle P, G, K, DS\rangle$ où la cohérence locale implique la cohérence globale. Soit Δ_p un diagnostic localement cohérent de DS_p. Par définition, tout diagnostic localement cohérent est cohérent avec Φ_p et chacune des descriptions des pairs voisins de p. Il existe donc un modèle accessible local m_p prolongeant Δ_p t.q. $\Delta_p = restrict(m_p, L_p)$. Or, tout modèle localement cohérent de DS est aussi globalement cohérent. Δ_p est donc globalement cohérent. □

Exemple. 4.1 *À la Figure 3.8, tous les modèles localement cohérents sont aussi globalement cohérents. On peut vérifier que les diagnostics localement cohérents exprimés à travers l'expression sur le langage cible de ces modèles sont aussi globalement cohérent. Ils font tous partie d'un diagnostic distribué.*

Inversement, nous rapprochons les diagnostics globalement cohérents d'une description de ses modèles globalement cohérents en remarquant que pour un langage cible particulier $vars(L) = V$, tout diagnostic global est un modèle global. C'est à l'aide de cette remarque que nous montrons qu'un système qui vérifie que ses diagnostics locaux sont globalement cohérents quelque soit le langage cible, alors ses modèles locaux sont aussi globalement cohérents.

Propriété 4.3 (Pour tout langage cible, la cohérence distribué est nécessaire) *Si pour tout langage cible, les diagnostics localement cohérents d'un système sont globalement cohérents alors ce système vérifie la cohérence distribuée.*

Preuve. Démonstration par l'absurde. Supposons qu'il existe un système $S : \langle P, G, K, DS\rangle$ t.q. pour tout langage cible les diagnostics localement cohérents sont globalement cohérents et que la cohérence locale de chacune des descriptions n'implique pas la cohérence globale. Si la cohérence locale de DS n'implique pas sa cohérence globale, il existe un modèle localement cohérent m_p de Φ_p qui n'est pas globalement cohérent. Si pour tout langage cible les diagnostics localement cohérents de DS sont globalement cohérents, alors la supposition est aussi vraie pour L le langage cible couvrant le vocabulaire du système global t.q. $(L = \bigcup_{p\in P} L_p$ et $\forall p, vars(L_p) = V_p)$. Notons que pour L tout diagnostic localement cohérent Δ_p de Φ_p est aussi un modèle de Φ_p. Or par hypothèse, tout diagnostic localement cohérent est globalement cohérent. En particulier

pour L où modèles et diagnostics sont équivalents, tout modèle Δ_p de Φ_p est donc globalement cohérent. Cette dernière remarque est en contradiction avec l'existence d'un modèle localement cohérent m_p de Φ_p qui n'est pas globalement cohérent. □

Exemple. 4.2 *En revenant à l'exemple de la Figure 3.8, considérons pour chaque pair le langage cible local formé à partir des littéraux positifs du vocabulaire du pair. Dans ce cas, chaque diagnostic localement cohérent est associé à un unique modèle localement cohérent. Pour ce langage cible ainsi choisi, l'implication entre cohérence locale et cohérence globale des diagnostics entraine l'implication entre cohérence locale et cohérence globale des modèles de chaque pair.*

Nous formalisons ci-après la correspondance entre diagnostic et cohérence distribuée à travers la propriété suivante :

Proposition 4.1 (la cohérence distribuée garantit un diagnostic distribué et vice versa) *Un système est correct pour le diagnostic distribué, pour tous ses langages cibles ssi il vérifie la cohérence distribuée de ses modèles.*

Preuve. (<=) est prouvé par la proposition 4.2
(=>) est prouvé par la proposition 4.3 dont (=>) est un corollaire. □

Par la dernière propriété on déduit que caractériser les systèmes où tous les diagnostics localement cohérents sont globalement cohérents indépendamment du langage cible revient à caractériser les systèmes où la cohérence locale des descriptions des pairs implique leur cohérence globale. En conclusion de cette section et en considérant les résultats intermédiaires nous déduisons que caractériser les systèmes accessibles S' garantissant un diagnostic distribué correct de S revient à caractériser les systèmes localement cohérents, accessibles et équivalents à S vérifiant la cohérence distribuée.

4.2 Propriétés graphiques des systèmes garantissant la cohérence distribuée

Avant de poursuivre la caractérisation des systèmes accessibles S' garantissant un diagnostic distribué correct de S par les systèmes vérifiant la cohérence distribuée, nous avons besoin d'abstraire un système de pairs par son schéma. Intuitivement, un schéma modélise la répartition du vocabulaire à travers le réseau d'accointances du système de pairs. C'est grâce aux propriétés du schéma d'un système que nous introduisons ici que nous pourrons caractériser les systèmes garantissant un diagnostic distribué correct.

Définition 4.4 (Schéma d'un système) *Le schéma d'un système $S : \langle P, G, K, DS \rangle$, est un graphe $G((P, V), ACQ)$ formé à partir du graphe d'accointances et du vocabulaire Voc du système tq*

1. *$G(P, ACQ)$ est le graphe d'accointances*
2. *$V : P \rightarrow 2^{Voc}$ associe à chaque pair p son vocabulaire V_p tq $\forall v \in V_p : K(p, v) = \top$.*

Exemple. 4.3 *Nous illustrons à la Figure 4.4 le schéma des systèmes de la Figure 3.8 et de la Figure 4.1. Nous notons entre accolades $\{vars\}$ les variables partagées entre les descriptions de pairs voisins.*

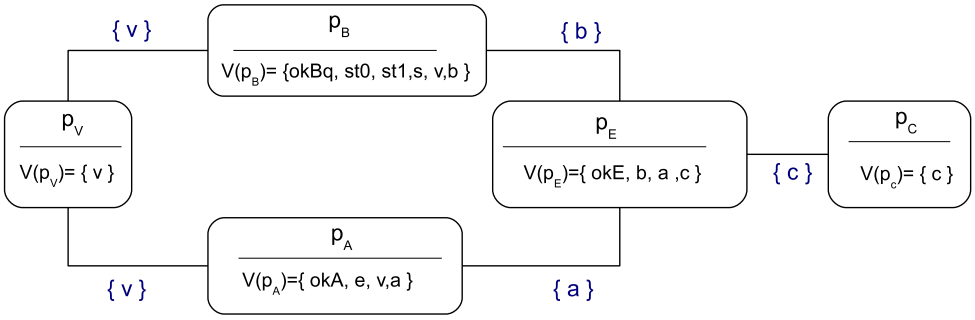


FIGURE 4.4 – Le schéma des systèmes des Figures 3.8 et 4.1

Un schéma est l'abstraction d'un système représentant la répartition du vocabulaire des pairs à travers le réseau. Pour un schéma G correspond donc un ensemble de systèmes $Sys(G)$ pour lesquels G modélise les liens d'accointance et la répartition du vocabulaire. Par abus de langage, nous dirons qu'un schéma : $G((P,V),ACQ)$ vérifie la proposition $Prop$ si et seulement si tous les systèmes $S : \langle P, G, K, DS\rangle$ qu'il modélise, S vérifie la proposition $Prop$ (c-à-d $\forall S \in Sys(G) Prop(G,S)$). Nous caractérisons les schémas et les systèmes corrects pour la compilation distribuée par les propriétés suivantes :

Définition 4.5 (schéma, système : joint,, structuré) *Soit $G((P,V),ACQ)$ un schéma. G est dit :*

-joint *si pour toute variable v, l'ensemble des pairs contenant v dans leur vocabulaire $\{p \ tq \ v \in V(p)\}$ induit un sous-graphe connecté de G.*
Un système est dit joint si son schéma est joint.

-structuré *s'il contient un arbre joint couvrant tous les pairs.*
Un système est dit structuré si son schéma est structuré.

Exemple. 4.4 *Le schéma de la Figure 4.4 est joint. Prenons par exemple la variable v, elle est contenue dans le vocabulaire des pairs $\{P_V, P_A, P_B\}$. Le sous-graphe du schéma induit par $\{P_V, P_A, P_B\}$ est un graphe connecté. D'autre part, nous illustrons à la Figure 4.5 un arbre couvrant joint représenté par des traits pleins. Afin de respecter la propriété de connectivité, par rapport à la Figure 4.4, le pair P_E contient en plus la variable v mise en gras dans le vocabulaire du pair.*

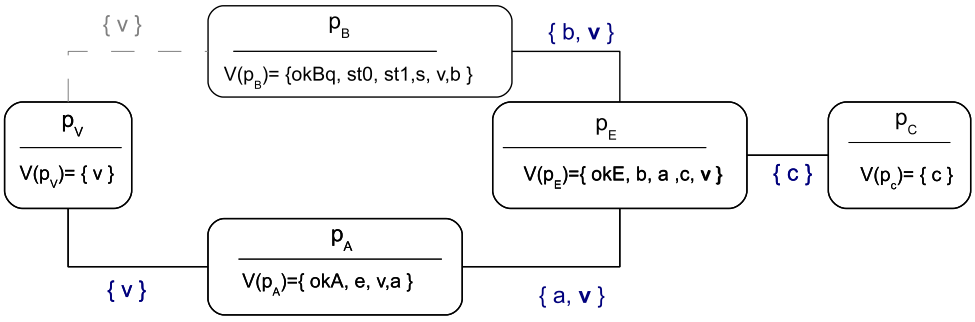


FIGURE 4.5 – Le schéma structuré du schéma de la Figure 4.4

De la définition précédente découle trivialement que tout graphe structuré est joint. D'autre part lorsque l'ensemble des pairs partagent le même ensemble de variables nous avons la propriété suivante :

Propriété 4.4 (Graphe joint étiqueté par un même ensemble de variables) *Tout schéma joint dont les pairs sont étiquetés par un même ensemble de variables est structuré.*

Trivialement, il suffit de constater que tout arbre couvrant un tel schéma est un arbre joint de ce schéma.

4.3 Un schéma structuré suffit à garantir la cohérence distribuée

Dans le cadre des CSPs, les travaux de Freuder [Fre82] ou Dechter [Dec92] ont montré qu'un réseau de contraintes acyclique qui satisfait l'arc cohérence (la cohérence locale) est globalement cohérent. Comme nous le verrons lorsque nous aborderons les systèmes d'interactions, la modélisation des réseaux de contraintes est dépendante du vocabulaire partagé. Ce n'est pas le cas pour les systèmes que nous considérons, nous redémontrons donc la propriété dans un cadre plus général. À travers la Figure 4.6 nous ressituons les systèmes structurés dans la caractérisation des systèmes garantissant un diagnostic distribué correct cf Figure 4.2.

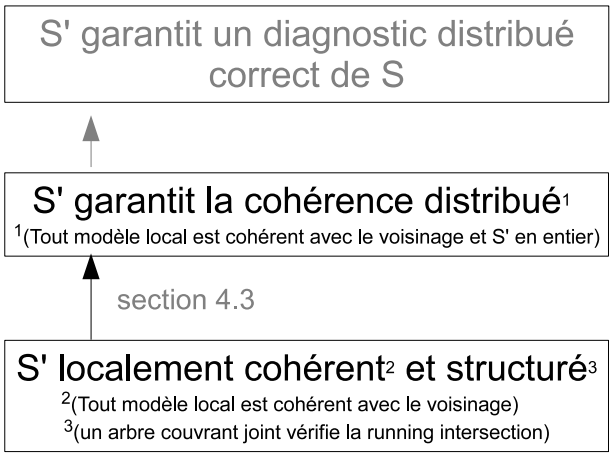


FIGURE 4.6 – Un système structuré suffit à garantir un diagnostic distribué correct

Nous démontrons tout d'abord la propriété suivante :

Propriété 4.5 (Cohérence d'une union de modèles partiels) *Toute union de modèles partiels deux à deux cohérents est cohérente.*

Preuve. Soit $M^n = \bigcup_{i=1..n} m_i$ une union de n modèles deux à deux cohérents. Montrons par récurrence sur n que M^n est cohérent.
Cas de base $n = 2$. Par définition M^2 est l'union de deux modèles m_1 et m_2 cohérents. M^2 est donc cohérent.
(HR) Supposons que toute union M^{n-1} de n-1 modèles deux à deux cohérents est cohérente. Montrons que M^n est cohérente. Posons $M^n = M^{n-1} \cup m_n$. Par (HR) M^{n-1} est cohérente. Compte tenu de la définition de M^n, on a m_n cohérent avec chacun des modèles m_j tq $1 \leq j \leq n-1$. On en déduit que m_n est cohérent avec M^{n-1}. L'union des modèles formée par M^n est donc

cohérente. □

Avant de présenter la condition suffisante à la cohérence distribuée nous présentons la propriété suivante. Elle nous est aussi utile pour la démonstration du théorème 4.1 que nous énonçons dans la suite.

Propriété 4.6 (Sous-arbre d'un arbre joint) *Tout sous-arbre d'un arbre joint est un arbre joint*

Preuve. Soient A un arbre joint et SA un de ses sous-arbre. Tous les chemins entre deux pairs de A respectent la *running intersection*. Or SA est un sous-arbre de A, alors tous les chemins de SA sont inclus dans les chemins de A. Tous les chemins de SA respectent donc la *running intersection*. □

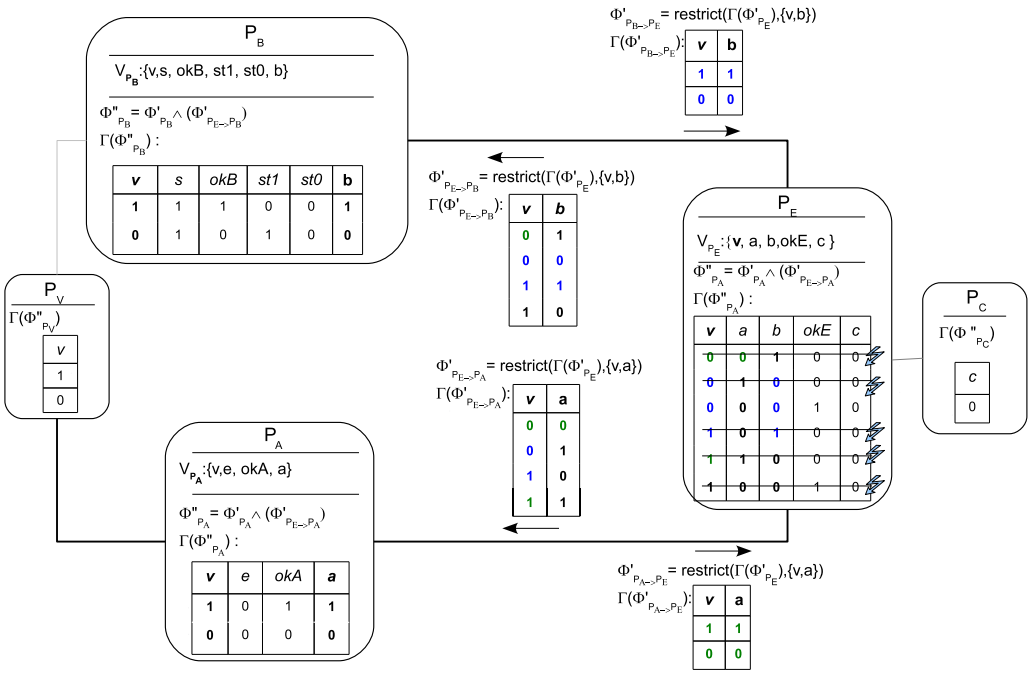


FIGURE 4.7 – Système structuré du schéma de la Figure 4.5

A l'aide de ces deux propriétés nous prouvons une condition suffisante sur le schéma garantissant la cohérence distribuée.

Théorème 4.1 (Condition suffisante pour la cohérence distribuée) *Tout système structuré, localement cohérent, garantit la cohérence distribuée.*

Preuve. Soit S un système structuré localement cohérent et $G((P,V),ACQ)$ son schéma. Notons A_r un arbre joint couvrant de G enraciné en un pair quelconque r. Afin de montrer le théorème 4.1, montrons par récurrence sur la profondeur d de A_r que tout modèle local de Φ_r est globalement cohérent.

Cas de base $d = 0$. La racine r est l'unique pair de A_r, tous ses modèles locaux sont globalement cohérents.

(HR) Supposons que tout modèle local à la racine d'un système structuré localement cohérent contenant un arbre joint couvrant de profondeur $1 \leq d" < n$ est globalement cohérent. Soit un système structuré localement cohérent dont l'un des arbres joints couvrant A_r enraciné en r est de profondeur $d = n$. Notons P' l'ensemble des voisins de r dans G. Pour tout voisin p' de r notons $A_{p'}$ le sous-arbre de A_r enraciné en p'. Par la propriété 4.6, on sait que $A_{p'}$ est un arbre joint de profondeur $< n$. Par (HR) tout modèle local $m_{p'}$ d'un voisin p' de r est globalement cohérent. $m_{p'}$ peut donc se prolonger en un modèle $m_{A_{p'}}$ cohérent avec l'union des descriptions des pairs du sous-arbre $A_{p'}$. En effet, on a : $profondeur(A_{p'} < n$ pour le système de pairs couvert par $A_{p'}$. Soient m_r un modèle local à r et $m_{A_{p'}}$, $m_{A_{p"}}$ deux modèles des sous-arbres enracinés en p', $p"$ voisins de r tq $m_{A_{p'}}$ et $m_{A_{p"}}$ sont cohérents avec m_r. Par la cohérence des modèles on a que toute assignation de variables partagées entre r et les pairs de A'_p sont identiques. De même toute assignation de variables partagées entre r et les pairs de $A_p"$ sont identiques. Or A_r est un arbre joint. Ainsi toute variable partagée entre $A_{p'}$ et $A_{p"}$ est contenue dans r. Avec la remarque précédente on déduit que toute assignation de variables partagées entre $m_{A_{p'}}$ et $m_{A_{p"}}$ sont identiques, $m_{A_{p'}}$ et $m_{A_{p"}}$ sont donc cohérents. En résumé nous obtenons que tous modèles des sous-arbres des pairs voisins de r, cohérents avec m_r, sont deux à deux cohérents. Par la propriété 4.5, nous déduisons que toute union de modèles des sous-arbres des pairs voisins de r cohérents avec m_r, est cohérente. m_r est donc globalement cohérente. □

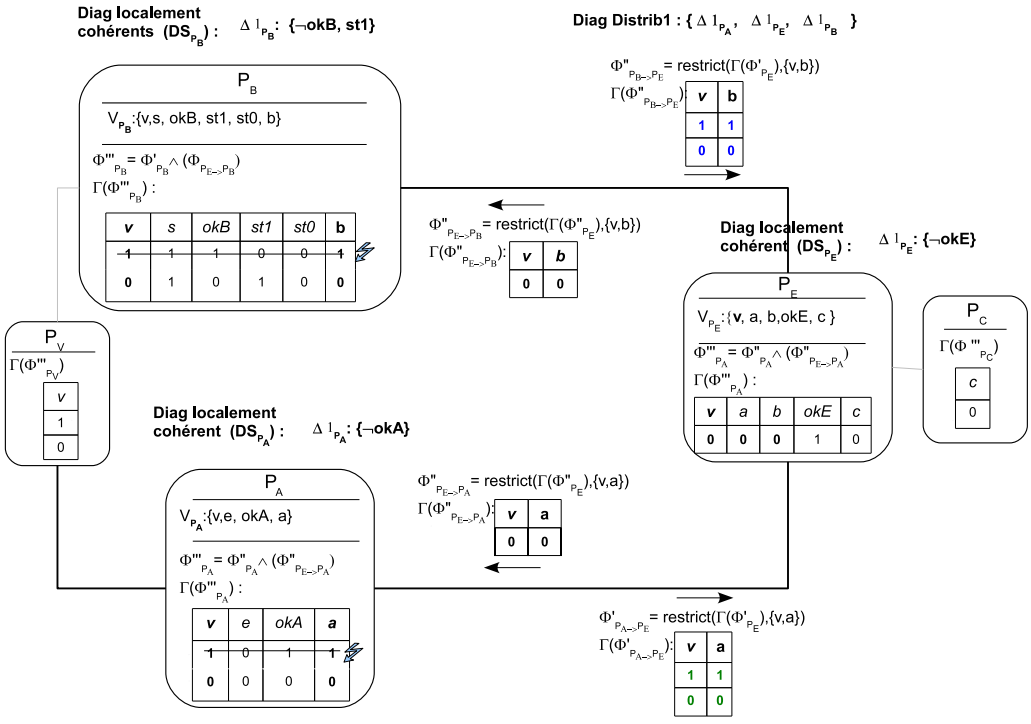


FIGURE 4.8 – Système structuré vérifiant la cohérence distribuée du schéma Figure 4.5

Nous avons illustré en début de chapitre (cf. Figure 4.1), un système non structuré où la cohérence locale n'impliquait pas la cohérence globale. Nous rendons ce système correct pour le consistance distribué en deux étapes. Premièrement à la Figure 4.7 nous le structurons. Son

schéma correspond au schéma structuré présenté à la Figure 4.5. À partir de la description du système initial Figure 4.1, seule la description de P_E est enrichie de la variable v. Nous avons vu que nous pouvons tester la cohérence de deux modèles uniquement à partir de leur expression sur les variables partagées. Afin de rendre chaque description localement cohérente, chaque couple de pairs (p_i, p_j) s'échangent une formule $\Phi'_{p_i \to p_j}$ satisfaite par la restriction des modèles de p_i exprimée sur le vocabulaire de p_i et p_j ($\Phi'_{p_i \to p_j} = restrict\Phi_{p_j} V(p_i) \cap V(p_j)$). À la réception d'un message $\Phi'_{p_i \to p_j}$ p_j met à jour sa description de façon à ce que tous ses modèles soient cohérents avec p_i. Il n'y a que P_E qui met à jour ses modèles et retire les modèles devenus incohérents (signalés par un éclair). Nous remarquons que sans l'ajout de la variable v au vocabulaire de la description de P_e, aucun modèle local à P_E n'aurait été supprimé. En réitérant ce processus, ce sont cette fois les descriptions accessibles des pairs P_B et P_A qui sont mises à jour. Ce système structuré vérifie la propriété de cohérence distribuée et garantit un diagnostic distribué correct.

À la Figure 4.8 nous présentons un système structuré localement cohérent. Nous avons mis en relief les assignations des variables partagées identiques entre les pairs P_{Ag} et P_{Es} et entre les pairs P_{Es} et P_{Bq}. De même au dessus de chaque pair nous illustrons les diagnostics localement cohérents faisant partie d'un diagnostic distribué du système global.

Nous rappelons que caractériser les systèmes où la cohérence locale implique la cohérence globale (c-à-d les systèmes garantissant la cohérence distribuée) nous permet de caractériser les systèmes garantissant un diagnostic distribué correct. Par le théorème 4.1, nous déduisons que tout algorithme capable de transformer un système observé de pairs vers un système accessible équivalent, localement cohérent et structuré suffit à garantir un diagnostic distribué correct mais n'est pas nécessaire. Dans la suite nous mettons en évidence des conditions nécessaires à la cohérence distribuée.

4.4 La cohérence distribuée nécessite un schéma joint

Une propriété suffisante nous renseigne sur les caractéristiques que peut satisfaire le schéma des systèmes accessibles pour garantir la cohérence distribuée. Une propriété nécessaire nous renseigne sur les caractéristiques que doivent respecter les schémas des systèmes accessibles. À travers la Figure 4.9, nous positionnons les schémas joints dans la caractérisation des systèmes garantissant un diagnostic distribué correct cf Figure 4.2.

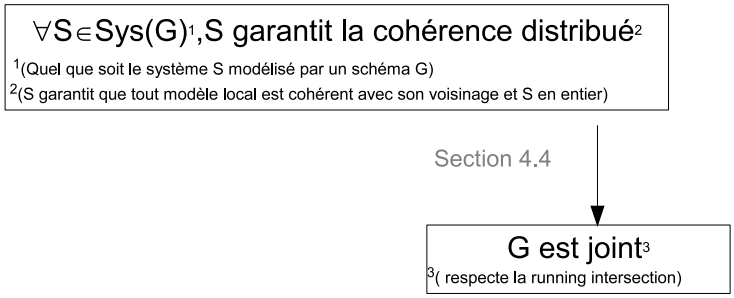


FIGURE 4.9 – Condition nécessaire pour la cohérence distribuée

Lemme 4.1 (système non joint localement cohérent, globalement incohérent) *Un schéma non joint ne garantit pas la cohérence distribuée pour tous les systèmes localement cohérents qu'il modélise.*

Afin de montrer ce lemme, nous montrons que pour tout schéma non joint G, il existe au moins un système localement cohérent qui accepte G comme schéma mais qui ne respecte pas la cohérence distribuée. La preuve de ce lemme est assez simple, mais nous n'avons pas trouvé de travaux l'ayant démontré donc nous le faisons. Dans la preuve nous considérons un modèle m comme une assignation de valeurs aux variables, $m(v)$ désigne l'assignation de la variable v dans le modèle m.

Preuve. Soit $G((P,V),ACQ)$ un schéma non joint. G contient au moins une variable v' tq l'ensemble des pairs contenant v' ne forme pas un sous-graphe connexe de G. Notons P' un ensemble de pairs induit par une des composantes connexes de G contenant v'. Rappelons qu'un modèle m est une interprétation, c-à-d , une assignation de valeurs aux variables. Nous notons $m(v)$ la valeur assignée à la variable v dans le modèle m. Considérons le système S acceptant G comme schéma formé par la description DS suivante :

1. Chaque formule $\Phi_{p'}$ des pairs p' de P' est satisfaite par un unique modèle m' t.q. $m'(v) = Vrai$ si $v \in V(p') \setminus \{v'\}$ et $m'(v') = Faux$.
2. Chaque formule Φ_p des pairs p de $P \setminus P'$ est satisfaite par un unique modèle m t.q. $m(v) = Vrai$ si $v \in V(P \setminus P')$.

A)Montrons que S est localement cohérent.

Par construction, pour chaque pair, l'unique modèle de chaque formule est une assignation positive du vocabulaire, exception faite pour les pairs P' qui assignent une unique variable : v' à $Faux$ et le reste des variables à $Vrai$. Par conséquent, s'il y a une incohérence entre pairs voisins, elle ne peut intervenir qu'entre un pair $p' \in P'$ et un autre pair $p \notin P'$ voisin d'un pair de P'. Or G est non joint par la variable v', et le graphe induit par les pairs de P' est une composante connexe de G contenant v' donc aucun pair dans le voisinage direct du graphe induit par les pairs de P' ne contient v'. Ainsi, il n'existe pas de couples de pairs voisins dont les modèles sont incohérents, DS est localement cohérent.

B)Montrons que G est globalement incohérent.

G est non joint par la variable v', alors il existe au moins un autre ensemble de pairs P" non joint de P' induit par un des sous-graphes connexes de G contenant v'. Par construction, le modèle de chaque description d'un pair de P" contient l'assignation $v' = Vrai$. Ainsi, puisque l'unique modèle de chacun des pairs de P' assigne v' à $Faux$, il n'existe pas de modèle global cohérent pour G. □

Exemple. 4.5 *Nous illustrons à la Figure 4.10 la construction d'un système non joint utilisé dans la preuve. Le schéma présenté est non joint par la variable v'. v' est à la fois contenue dans le graphe connexe P' formé par p'_1 et p'_2 mais aussi dans p"$_1$ qui n'est pas connecté à P'. L'unique modèle satisfaisant chaque théorie est une assignation positive(1) sauf pour les pairs de P' qui assignent v' négativement (0). Puisqu'aucun voisin des pairs de P' ne contient la variable v', le système est localement cohérent. Par contre, puisqu'il existe un autre pair p"$_1$ qui assigne la variable v' à 1 le système ne respecte pas la cohérence globale.*

Théorème 4.2 (Un schéma joint est nécessaire à la cohérence distribuée) *Tout schéma garantissant la cohérence distribué est joint.*

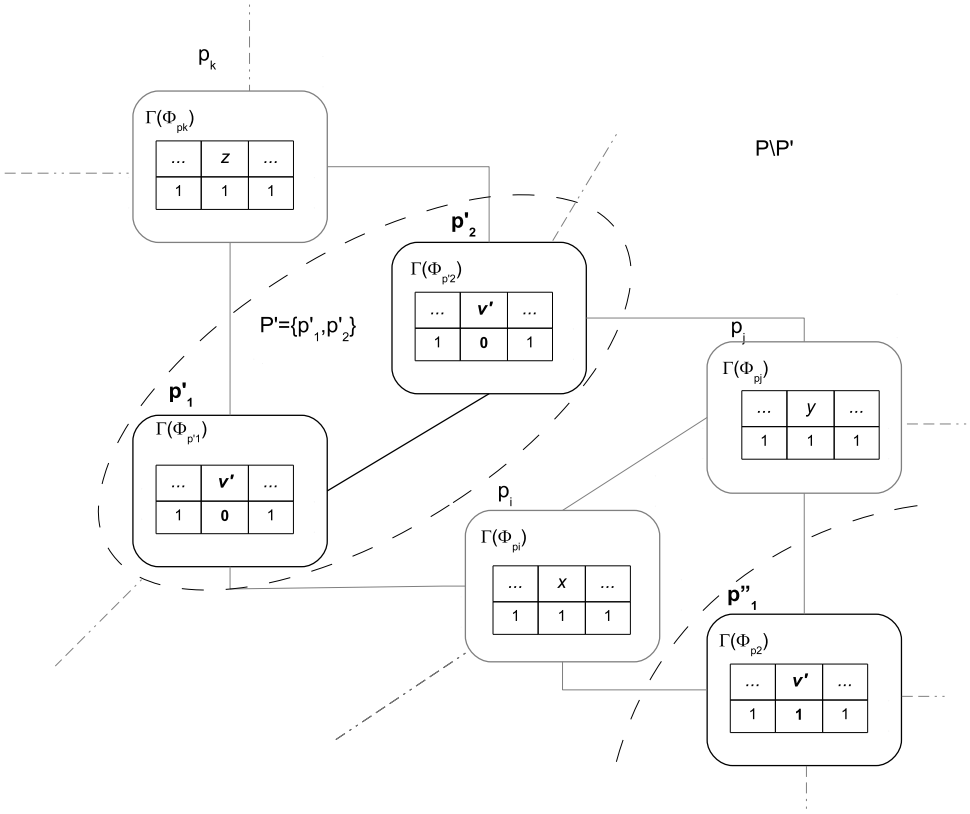


FIGURE 4.10 – Système non joint où la cohérence locale n'implique pas la cohérence globale

Preuve. Par l'absurde et à l'aide du lemme 4.1. Supposer qu'il existe un schéma non joint G que n'importe quel système localement cohérent qu'il modélise vérifie la cohérence distribuée est en contradiction avec le lemme 4.1. Il existe toujours un système où la cohérence locale n'implique pas la cohérence globale. □

4.5 Pour les systèmes d'interactions, seul un schéma structuré garantit la cohérence distribuée

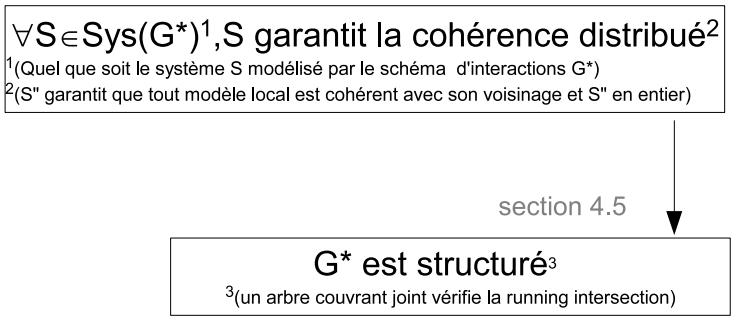


FIGURE 4.11 – Condition nécessaire pour la cohérence distribuée de systèmes d'interactions

Contrairement aux sections précédentes où nous avons situé la caractérisation des systèmes garantissant la cohérence distribuée dans un cadre général, dans cette section nous focaliserons la caractérisation des systèmes garantissant la cohérence distribuée aux systèmes d'interactions. Un système d'interactions est un système pour lequel tout vocabulaire partagé entre deux pairs implique un lien d'accointance entre ces derniers. Nous montrons que pour ces systèmes, seuls les schémas structurés garantissent la cohérence distribuée.

À travers la Figure 4.11, nous ressituons les schémas d'interactions dans la caractérisation des systèmes garantissant un diagnostic distribué correct cf. Figure 4.2.

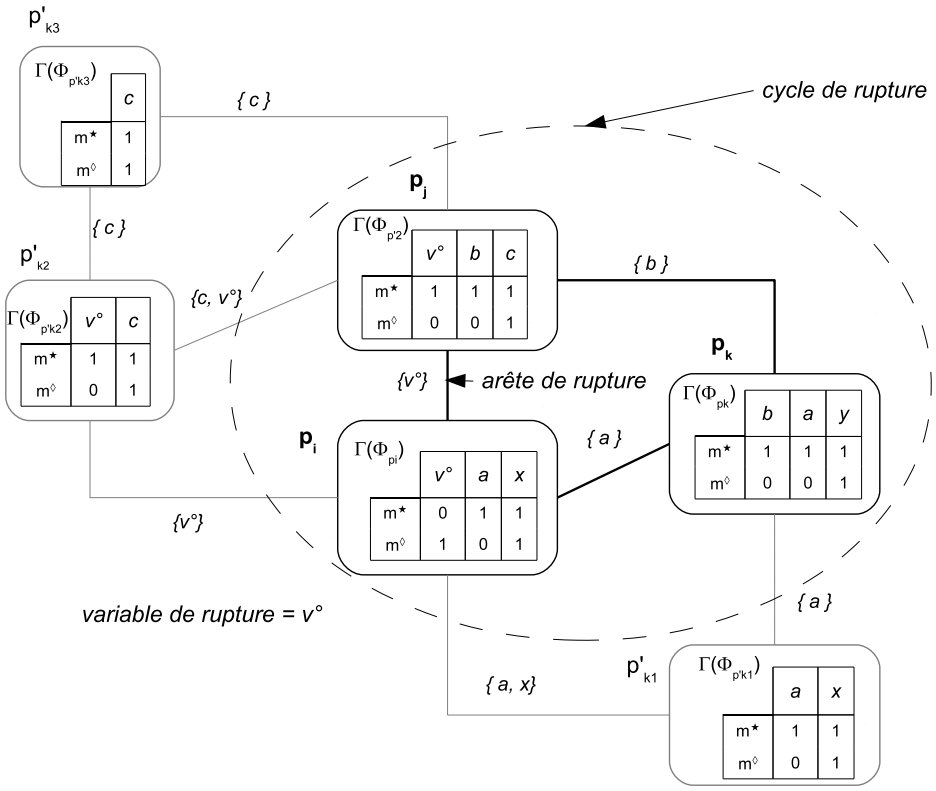


FIGURE 4.12 – Systèmes d'interactions non structuré localement cohérent globalement incohérent

À travers la Figure 4.14, nous montrons par un exemple que lorsque le système d'interactions n'est pas structuré, il modélise au moins un système où la cohérence locale n'implique pas la cohérence globale. Le schéma ne peut donc pas garantir un diagnostic distribué correct. On peut vérifier que tout modèle local $m^\star$ ou $m^\diamond$ d'un pair p est localement cohérent. Pourtant, prenons le cercle formé par les pairs p_i, p_j et p_k. On constate que l'on ne peut prolonger de façon cohérente le modèle $m^\star_{p_i}$ de p_i que par les modèles locaux $m^\star_{p_k}$ de p_k et $m^\diamond_{p_j}$ de p_k. Or $m^\star_{p_k}$ et $m^\diamond_{p_j}$ sont incohérents à cause de l'assignation contradictoire sur b. De même, on ne peut prolonger de façon cohérente le modèle $m^\diamond_{p_i}$ que par les $m^\diamond_{p_k}$ et $m^\star_{p_j}$. Or $m^\diamond_{p_k}$ et $m^\star_{p_j}$ sont incohérents pour les même raisons. Pour ce système la cohérence locale n'implique pas la cohérence globale. C'est la généralisation de cet exemple qui nous permettra de démontrer la propriété statuant que seuls les schémas d'interactions structurés garantissent la cohérence distribuée. Tout d'abord nous introduisons l'opération de *restriction* de schéma sur un sous-ensemble de variables. Elle nous permettra de focaliser la caractérisation sur un sous-système induit par un vocabulaire intéressant. Pour mieux comprendre, prenons la variable a. Le sous-schéma restreint à la variable

a comprend les pairs p_i, p'_{k1} et p_k, chacun ayant un vocabulaire réduit à $\{a\}$ ce schéma est structuré. En effet, l'arbre (p_k, p_i), (p_k, p'_{k1}) est annoté par a est un arbre joint. Maintenant, considérons les variables $\{a, b\}$. Le sous-schéma restreint au vocabulaire $\{a, b\}$ comprend les pairs p_i , p_k, p'_{k1} et p_j, il est aussi structuré. Il suffit d'ajouter l'arête (p_k, p'_{k1}) à l'arbre formé précédemment et restreindre le vocabulaire des pairs concernés à $\{a, b\}$. Enfin, considérons les variables $\{a, b, v^\circ\}$. Le sous-schéma restreint au vocabulaire $\{a, b, v^\circ\}$ comprend les pairs p_i, p_k, p'_{k1}, p_j, ce schéma n'est pas structuré. Il n'existe aucun arbre joint couvrant les p_i , p_k, p'_{k1}, p_j. Dans ce déroulement nous montrons que rajouter une simple variable v° aux variables $\{a, b\}$ nous fait basculer d'un sous graphe structuré vers un sous arbre joint non structuré. Dans la suite, nous qualifions l'ensemble $\{a, b, v^\circ\}$ de vocabulaire joint non structuré minimal et v° de *variable de rupture*. De même nous constatons que le sous graphe restreint sur son vocabulaire joint non structuré minimal (ex $\{a, b, v^\circ\}$) fait apparaître une arête uniquement étiquetée par la variable de rupture (v° étiquette $p_i - p_j$) et un cycle contenant cette arête (p_i, p_j, p_k). C'est grâce à cette *arête de rupture* et à ce *cycle de rupture* que nous sommes en mesure de vérifier que pour tout schéma d'interactions non structuré, il existe au moins un système où la cohérence locale n'implique pas la cohérence globale.

4.5.0.1 Propriétés graphiques des systèmes d'interactions

Tout d'abord, définissons les notions de schéma et systèmes d'interactions.

Définition 4.6 (Schéma et système d'interactions) *Un schéma : $G((P, V), ACQ)$ est dit d'interactions ssi toute variable partagée entre deux pairs implique un lien d'accointances entre ces pairs c-à-d :*

$$V(p_i) \cap V(p_j) \neq \emptyset, (p_i, p_j) \in ACQ \tag{4.1}$$

Un système est dit d'interactions si son schéma est un schéma d'interactions.

Exemple. 4.6 *Le schéma de la Figure 4.13 est un système d'interactions reprenant les liens du schéma Figure 4.4 auquel il a été nécessaire d'ajouter un lien d'accointances afin de respecter la dépendance des pairs partageant la variable v.*

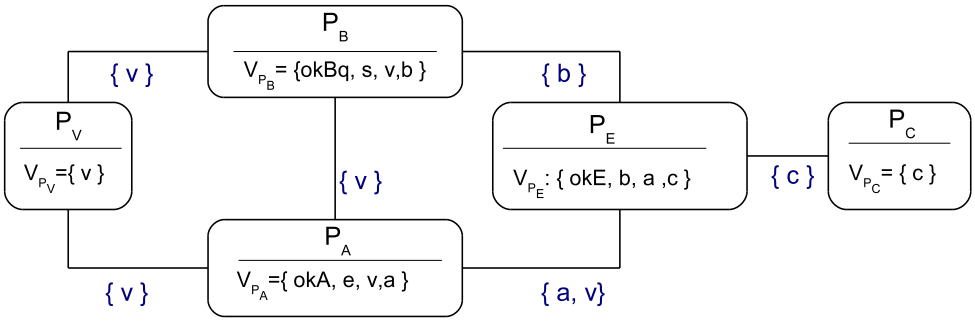


FIGURE 4.13 – Un schéma d'un système d'interactions

Nous ouvrons une parenthèse, ici, pour remarquer que ce que nous qualifions de schéma d'interactions correspond à la notion de graphe dual utilisé dans le cadre des CSPs pour modéliser des problèmes de contraintes (cf. Dechter [Dec03]). Dans un graphe dual chaque noeud est étiqueté par les variables d'une contrainte, il existe un lien entre deux noeuds s'ils partagent une variable. De même, on peut aussi rapprocher la notion de schéma d'interactions de

la notion d'hyper-graphe utilisé par Gottlob [GLS99] pour modéliser un problème. Dans ce cas, une variable étiquette un noeud et chaque hyper-arête regroupe les variables d'une contrainte. Nous soulignons, que ces dernières modélisations ne permettent pas de représenter des liens d'accointances indépendants de la répartition du vocabulaire à travers le système.

Les schémas d'interactions ont les propriétés suivantes :

Remarque 4.1 (Cliques d'un schéma d'interactions) *Tout schéma d'interactions est un graphe joint où le sous graphe induit par une variable v est une clique.*

Preuve. De la Définition 4.6 de schéma d'interactions nous déduisons que tout ensemble de pairs contenant une même variable v est lié à travers une clique. Ainsi pour toute variable v d'un schéma d'interactions, l'ensemble des pairs contenant v est connecté. □

Exemple. 4.7 *Reprenons le schéma de la Figure 4.13. Nous remarquons que l'ensemble de pairs contenant la variable v forme une clique. C'est aussi le cas pour toutes les autres variables.*

Définition 4.7 (Restriction de graphes) *Soit $G((P,V),ACQ)$ un schéma d'interactions. $G^\circ((P^\circ,V^\circ),ACQ^\circ)$ est une restriction de G induite par les variables V° ssi*

1. $P^\circ = \{p \in P \text{ t.q. } V(p) \cap V^\circ \neq \emptyset\}$.
   P° est le sous-ensemble des pairs de P contenant au moins une variable de V°.
2. $\forall p \in P^\circ$, $V^\circ(p) = V^\circ \cap V(p)$
   Chaque pair p de G° est étiqueté par l'intersection de V° et de $V(p)$ de G.
3. $V^\circ(p_i) \cap V^\circ(p_j) \neq \emptyset \Rightarrow (p_i,p_j) \in ACQ^\circ$
   La restriction d'un schéma d'interactions est un schéma d'interactions.

Définition 4.8 (Vocabulaire structuré) *Soit $G((P,V),ACQ)$ un schéma d'interactions et $V^\circ \subseteq \bigcup_{p\in P} V(p)$ un ensemble de variables.*

1. *V° est dit structuré ssi le graphe restreint induit par V° est structuré*
2. *V° est un vocabulaire non structuré minimal ssi V° est non structuré et pour tout vocabulaire $V^\triangle \subset V^\circ$, $V^\triangle$ est structuré.*

Nous remarquons que la restriction d'un graphe joint sur un sous ensemble de variables inclus dans V est joint.

Propriété 4.7 (Vocabulaire joint non structuré minimal d'un graphe joint) *Tout graphe joint non structuré contient un vocabulaire joint non structuré minimal.*

Preuve. Au pire un vocabulaire singleton $V^\triangle = \{v\}$ est structuré, le graphe induit représente une clique. □

Propriété 4.8 (Variable de rupture d'un schéma d'interactions) *Pour tout graphe joint non structuré il existe un ensemble de variables $V^\triangle \subset V$ et une variable $v^\circ \in V$ dite de rupture t.q.*

1. *$V^\triangle$ est structuré*
2. *$V^\circ = V^\triangle \cup \{v^\circ\}$ est non structuré minimal*

Preuve. D'après la propriété , il existe $V^\circ \subseteq V$ joint non structuré minimal. il existe donc $v^\circ \in V^\circ$ et $V^\triangle = V^\circ \setminus \{v^\circ\}$ □

Propriété 4.9 (Arête de rupture d'un schéma d'interactions) *Soit $G^\circ((P^\circ, V^\circ), ACQ^\circ)$ la restriction d'un schéma d'interactions $G((P,V), ACQ)$ sur un vocabulaire non structuré minimal $V^\circ = V^\triangle \cup \{v^\circ\}$ où v° est une variable de rupture.*
Il existe une arête $(p_i, p_j) \in ACQ^\circ$ dite de rupture tq :
$V^\circ(p_i) \cap V^\circ(p_j) = \{v^\circ\}$, $\{v^\circ\} \subset V^\circ(p_i)$ *et* $\{v^\circ\} \subset V^\circ(p_j)$

Afin de montrer que tout schéma d'interactions G restreint sur un vocabulaire non structuré minimal $V^\circ = V^\triangle \cup v^\circ$ contient une arête de rupture, nous comparons la restriction de G sur V° : G° par rapport à la restriction de G sur $V^\triangle$: $G^\triangle$. Nous montrons que G° contient une arête (p_i, p_j) n'apparaissant pas dans les arêtes de $G^\triangle$ t.q. $V^\circ(p_i) \cap V^\circ(p_j) = \{v^\circ\}$.
Preuve. Soient un schéma d'interactions $G((P,V), ACQ)$ et $V^\circ = V^\triangle \cup v^\circ$ un vocabulaire non structuré minimal. Soit $G^\circ((P^\circ, V^\circ), ACQ^\circ)$ la restriction de G induite par V°, G° est non structuré. Soit $G^\triangle((P^\triangle, V^\triangle), ACQ^\triangle)$ la restriction de G à $V^\triangle$, $G^\triangle$ est structuré. On a $P^\circ = P^\triangle \cup P^*$, où P° représente les pairs dont le vocabulaire se limite à $\{v^\circ\}$ dans G°. Par la propriété 4.4, nous avons que le sous graphe de G^* induit par les pairs P^* est structuré. Nous déduisons que si G est non structuré, c'est à cause du schéma induit par les pairs $P^\triangle$. Dans la suite nous noterons $G^\circ((P^\triangle, V^\circ), ACQ^\circ)$ le sous graphe représentant la restriction de G sur V° induite par les pairs $P^\triangle$. Soit $A^\triangle$ un arbre joint couvrant $P^\triangle$ dans $G^\triangle$. $A^\triangle$ n'est pas joint dans G° à cause de v° d'au moins une arête (p_i, p_j) (où $\{v^\circ\} \subset V^\circ(p_i)$, $\{v^\circ\} \subset V^\circ(p_j)$ de ACQ°) n'appartenant pas à $A^\triangle$ tq :

1. (p_i, p_j) est un nouveau lien de ACQ° non présent dans $ACQ^\triangle$, dans ce cas $V^\circ(p_i) \cap V^\circ(p_j) = \{v^\circ\}$ et la propriété est démontrée, ou :
2. (p_i, p_j) est un ancien lien de $ACQ^\triangle$, dans ce cas $\{v^\circ\} \subset V^\circ(p_i) \cap V^\circ(p_j)$.

(Hypothèse H1) Supposons que G° ne contient que des liens anciens (p_i, p_j) en provenance de $G^\triangle$. Notons $V^\circ(p_i) \cap V^\circ(p_j) = \{v^\circ\} \cup V^*$. $A^\triangle$ n'est pas joint dans G° à cause de (p_i, p_j) et est joint dans $G^\triangle$. Il existe donc un cycle C^* contenant (p_i, p_j) et suivant le chemin de $A^\triangle$ reliant p_i à p_j dans G° tq pour tout pair p du cycle $V^* \subseteq V^\circ(p)$. (Hypothèse H2) Supposons que le cycle C^* est un cycle joint et structuré. Dans ce cas, pour un tel cycle C^*, il existe au moins une arête (p_k, p_l) où $V^\circ(p_k) \cap V^\circ(p_l) = V^*$. (p_k, p_l) pourra être remplacée par (p_i, p_j) dans $A^\triangle$ de sorte que $A^\triangle$ restera un arbre joint pour le vocabulaire $V^\triangle$ et respectera la connectivité de (p_i, p_j). Si tout cycle C^* composé de (p_i, p_j) et d'un chemin de $A^\triangle$ est structuré, il existe donc un arbre joint couvrant $P^\triangle$ dans G°. Cette dernière remarque est en contradiction avec l'hypothèse de départ qui dit que G° n'est pas structuré. En supposant H1 vraie (et donc H2 faux) nous déduisons donc qu'il existe un cycle C^* joint et non structuré. Dans ce cas toutes les arêtes (p_k, p_l) de C^* sont t.q. $V \subset V^\circ(p_k) \cap V^\circ(p_l)$. Ainsi les pairs de C^* restent connectés entre eux par au moins une autre variable que V^*. Notons $C^\circledast$ la restriction de C^* sur $V^\circ \setminus V^*$, puisque C^* est non structuré, $C^\circledast$ ne l'est pas non plus. En effet seul le vocabulaire commun V^* entre tous les pairs a été supprimé. Nous en déduisons que la restriction de G sur $V^\circ \setminus V^*$ est aussi non structurée. Cette dernière remarque est en contradiction avec notre hypothèse initiale qui considère V° comme un vocabulaire non structuré minimal. Nous déduisons donc que l'hypothèse H1 est fausse, ACQ° contient donc un nouveau lien n'apparaissant pas dans $ACQ^\triangle$ t.q. $V^\circ(p_i) \cap V^\circ(p_j) = \{v^\circ\}$. □

Propriété 4.10 (Cycle de rupture d'un schéma d'interactions) *Soit (p_i, p_j) une arête de rupture de la restriction d'un schéma d'interactions G sur un vocabulaire non structuré minimal $V^\circ = V^\triangle \cup v^\circ$. (p_i, p_j) est couvert par un cycle joint non structuré dit cycle de rupture.*

Preuve. Soit $G^\circ((P^\circ, V^\circ), ACQ^\circ)$ la restriction d'un schéma d'interactions $G((P, V), ACQ)$ sur le vocabulaire non structuré minimal $V^\circ = V^\triangle \cup v^\circ$. Soient $G^\triangle((P^\triangle, V^\triangle), ACQ^\triangle)$ la restriction de G sur le vocabulaire structuré $V^\triangle$ et $A^\triangle$ un de ses arbres joints. Dans la preuve de la propriété 4.9, nous avons vu qu' une arête de rupture (p_i, p_j) était un nouvel arc ajouté entre les pairs p_i et p_j précédemment liés par un chemin joint $C^\triangle$ de $A^\triangle$ dans $G^\triangle$. G° est joint, le cycle formé par $C^\triangle$ et (p_i, p_j) dans G° est joint. □

Théorème 4.3 (Incohérence globale des systèmes d'interactions non structurés) *Un schéma d'interactions non structuré ne peut garantir la cohérence distribuée.*

Afin de faciliter la lecture de la preuve, à la Figure 4.14 nous illustrons le système d'interactions non structuré localement cohérent et globalement incohérent que nous construisons dans la preuve, par notre exemple introduit en début de section.

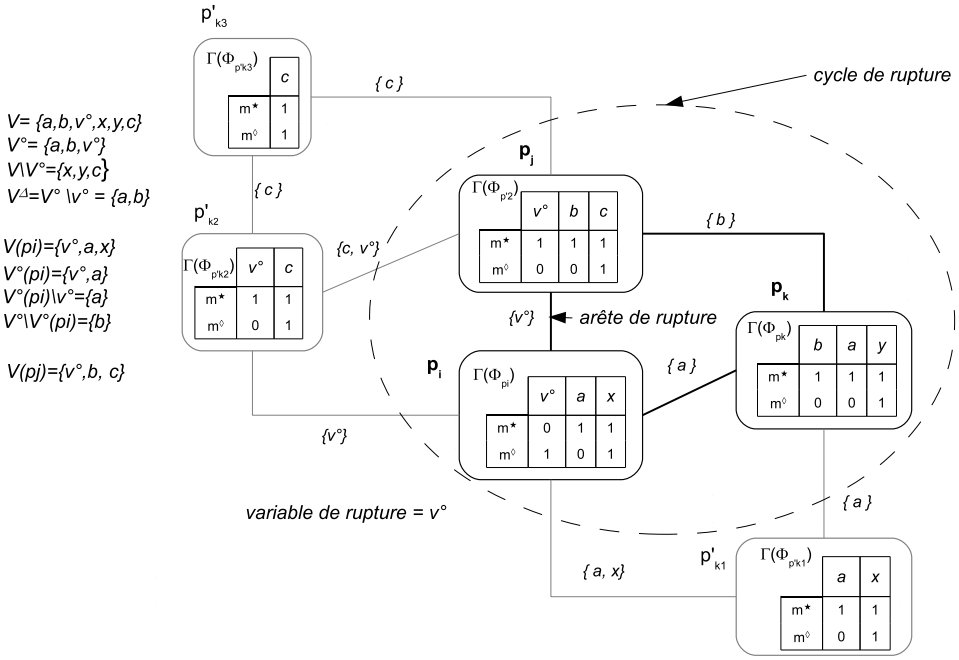


FIGURE 4.14 – Système d'interactions illustrant la preuve du th : 4.3

Preuve. Soient un schéma d'interactions $G((P, V), ACQ)$ et $G^\circ((P^\circ, V^\circ), ACQ^\circ)$ la restriction de G sur un vocabulaire non structuré minimal $V^\circ = V^\triangle \cup v^\circ$ où v° représente une variable de rupture. Notons (p_i, p_j) l'arête de rupture étiquetée par v°. Considérons la description accessible DS, acceptant G comme schéma, satisfaite pour chaque pair p par les modèles $m_p^\diamond$ et $m_p^\star$:

En entête de colonne, nous séparons tout d'abord le vocabulaire minimal non décomposable V° du reste des variables du système $V \setminus V^\circ$. Nous séparons ensuite le vocabulaire $V^\circ(p_i)$ des autres variables de V°. Nous rappelons que $V^\circ = \bigcup_{i=1\ldots n} V^\circ(p_i)$. Enfin, nous distinguons la

$v \in$		V°			$V \setminus V^\circ$
		v°	$V^\circ(p_i) \setminus \{v^\circ\}$	$V^\circ \setminus V^\circ(p_i)$	
$V(p_i)$	$m^\star_{p_i}$	Faux	Vrai	$\emptyset$	Vrai
	$m^\diamond_{p_i}$	Vrai	Faux	$\emptyset$	Vrai
$V(p_j)$	$m^\star_{p_j}$	Vrai	$\emptyset$	Vrai	Vrai
	$m^\diamond_{p_i}$	Faux	$\emptyset$	Faux	Vrai
$V(p_k)$, $k \notin \{i,j\}$	$m^\star_{p_k}$	Vrai	Vrai	Vrai	Vrai
	$m^\diamond_{p_k}$	Faux	Faux	Faux	Vrai

variable v° des autres variables de $V(p_i)$. En entête de ligne, nous distinguons le vocabulaire des pairs p_i, p_j de l'arête de rupture de celui des autres pairs p_k du système. Ensuite pour chaque pair, nous renseignons l'assignation de valeurs des modèles $m^\diamond_p$ et $m^\star_p$ en fonction de la variable en colonne. Par exemple, la première case en haut à gauche indique que l'assignation de la variable v° dans $m^\diamond_{p_i}$ est assignée à $Faux$. L'assignation d'une variable de $V^\circ(p_i) \setminus v^\circ$ dans $m^\diamond_{p_i}$ est à $Vrai$. Une assignation vide ($\emptyset$) indique que dans aucun cas le vocabulaire du pair ne contiendra le type de variables indiqué en colonne. À la ligne $V(p_j)$ le pair p_j ne partageant avec p_i que la variable v° dans V°, p_j assigne $\emptyset$ en face de la colonne $V^\circ_{p_i} \setminus v^\circ$. De même, le vocabulaire du pair $p_i : V(p_i)$ n'a en commun avec V° que $V^\circ_{p_i}$. Nous soulignons que G étant un graphe restreint de G, $V^\circ \subseteq V$.

A) Montrons que DS est localement cohérente.

Considérons les pairs $\{p_i, p_j\}$ de l'arête de rupture. La seule variable partagée entre p_i et p_j est v°. Dans le tableau des assignations on déduit que ($m^\star_{p_i}$ et $m^\diamond_{p_j}$) (resp ($m^\diamond_{p_i}$ et $m^\star_{p_j}$)) sont cohérents. Pour toute autre arête (p, p') différente de (p_i, p_j), on peut vérifier que $m^\diamond_p$ et $m^\diamond_{p'}$ (resp $m^\star_p$ et $m^\star_{p'}$) sont cohérents. Ainsi, tout couple de pairs voisins dans DS est cohérent, la description du système DS est donc localement cohérente.

B) Montrons que DS est globalement incohérente.

Nous avons constaté que pour l'arête de rupture (p_i, p_j), $m^\star_{p_i}$ est l'unique modèle de p_i cohérent avec $m^\diamond_{p_j}$. De même $m^\star_{p_j}$ est l'unique modèle de p_j cohérent avec $m^\diamond_{p_i}$. Par la propriété 4.10, nous savons qu'il existe un cycle de rupture C' couvrant (p_i, p_j) dans G°. C' est joint donc tout couple de pairs voisins dans C' se partagent au moins une variable de V°. Notons E' l'ensemble des couples de pairs voisins composant C' excepté $\{p_i, p_j\}$. Dans le tableau des assignations nous pouvons vérifier que toute variable $v \in V^\circ$ partagée entre voisins $\{p, p'\}$ dans E' est assignée de façon contradictoire dans $m^\diamond_p$ et $m^\star_{p'}$. Le modèle $m^\diamond_p$ est donc l'unique modèle de p cohérent avec $m^\diamond_{p'}$. Par un raisonnement similaire, $m^\star_{p'}$ est l' unique modèle de p' cohérent avec $m^\star_p$. Nous en déduisons qu'il existe au plus deux assignations : $M^\diamond_{E'}$ et $M^\star_{E'}$ satisfaisant l'ensemble des couples de pairs voisins de E' t.q. :

- $M^\diamond_{E'} = \bigcup_{\{p,p'\} \in E'} m^\diamond_p \cup m^\diamond_{p'}$
- $M^\star_{E'} = \bigcup_{\{p,p'\} \in E'} m^\star_p \cup m^\star_{p'}$

Ainsi, le modèle local $m^\star_{p_i}$ de p_i n'est cohérent qu'avec le modèle local $m^\diamond_{p_j}$ de p_j. Or $m^\star_{p_i}$ et $m^\diamond_{p_j}$ ne sont pas tous les deux cohérents avec $M^\diamond_{E'}$ (à cause de $m^\star_{p_i}$) et ne sont pas tous les deux cohérents avec $M^\star_{E'}$ (à cause de $m^\diamond_{p_j}$). De même, les assignations $m^\star_{p_j}$ et $m^\diamond_{p_i}$ sont contradictoires avec $M^\diamond_{E'}$ et $M^\star_{E'}$. Il n'existe donc pas d'assignation valide satisfaisant les descriptions du cycle C'. La description DS est globalement incohérente.

□

Corollaire 4.1 *Tout schéma d'interaction garantissant la cohérence distribuée est structuré.*

Preuve. Par l'absurde. Supposer qu'il existe un schéma G non structuré garantissant pour toutes les descriptions qu'il modélise que la cohérence locale implique la cohérence globale est en contradiction avec le théorème 4.3. □

Théorème 4.4 (Les schémas d'interactions garantissent la cohérence distribuée) *Un schéma d'interactions garantit la cohérence distribuée ssi il est structuré.*

Preuve.
<= Nous avons montré par le théorème 4.1 que tout système structuré garantit la cohérence distribuée. Un système d'interactions est un système particulier pour lequel s'applique le théorème. Ainsi nous savons que tout système d'interactions structuré garantit la cohérence distribuée. Nous déduisons a fortiori que tous les systèmes d'un schéma d'interactions structuré garantissent la cohérence distribuée. Un schéma d'interactions structuré garantit donc la cohérence distribuée.
=> est prouvé par le corollaire 4.1

□

En conclusion, dans ce chapitre nous nous sommes intéressés à caractériser par leurs topologies, les systèmes accessibles pour lesquels il est possible de prolonger avec certitude tout diagnostic localement cohérent par un diagnostic global d'un système. La première étape a été de réduire la vérification d'un système garantissant un diagnostic distribué correct à un système garantissant la cohérence distribuée (c-à-d où tout modèle local est globalement cohérent). Dans la seconde étape, nous avons montré que les systèmes accessibles structurés, c-à-d , qui contiennent un arbre couvrant respectant la running intersection jouaient un rôle prépondérant dans la caractérisation. Dans le chapitre suivant, nous montrons comment transformer le système de pairs vers un système accessible structuré tout en respectant les accointances des pairs et une politique de confidentialité visant à garder les variables locales secrètes.

Chapitre 5

Décomposition d'un système distribué vers un réseau structuré respectant sa confidentialité

Dans le chapitre précédent, nous avons vu que des descriptions accessibles structurées suffisent à garantir un diagnostic distribué correct. Dans le cas des systèmes d'interactions, nous avons montré que les descriptions accessibles structurées étaient nécessaires. Ainsi autant concentrer tout l'effort sur cette construction sur sa qualité, puisqu'elle est indispensable. C'est donc tout naturellement que nous chercherons dans ce chapitre à transformer la description d'un système vers un système structuré en vue de le diagnostiquer. Nous introduisons dans ce chapitre la décomposition arborescente de systèmes distribués. Nous avons vu que la décomposition arborescente, introduite par Robertson et Seymour [RS86] est une technique bien connue qui a permis d'accélérer la résolution de nombreux problèmes difficiles. Elle a pour but de transformer la représentation graphique d'un problème vers un arbre de clusters de telle sorte que toutes les variables liées dans le graphe initial apparaissent ensemble dans un des clusters de variables de l'arbre résultat. Une fois le système décomposé, le temps de calcul du diagnostic distribué pourra être borné pour de nombreux problèmes. Au delà de la nécessité de décomposer, c'est cette propriété qui explique pourquoi la décomposition arborescente a été largement étudiée dans un contexte centralisé (par exemple la théorie des graphes [RS86], l'optimisation de contraintes [KDLD05, Dec06, JNT07], la planification [GG02], les bases de données [GLS00, DGG$^+$08], ou encore la représentation des connaissances [HD07, SBH07, FdGJ09], mais aussi dans un contexte distribué (c-à-d l'optimisation de contraintes distribuées [PF05, EBBB08, LF11]). Intuitivement, la décomposition arborescente guide le mécanisme de raisonnement par l'intermédiaire du réseau, tout en bornant le nombre de messages. Dans notre contexte distribué, afin de garantir un temps de réponse, il est en effet essentiel de pouvoir borner le nombre maximal de messages.

Lorsqu'un système est décomposé en sous-systèmes, le temps et l'espace mémoire nécessaire à sa résolution sont bornés par une exponentielle dépendant du nombre maximal de variables du plus large sous problème (c-à-d la largeur de la décomposition). De nombreuses applications reposent sur de bonnes décompositions arborescentes, et de nombreuses classes polynomiales sont basées sur l'existence d'une bonne décomposition. En raison de son impact exponentiel sur la borne de complexité, même une petite amélioration dans la qualité de la décomposition peut conduire à des améliorations importantes en pratique.

Toutefois, lorsque le système est intrinsèquement distribué, ou soumis à une politique de confidentialité (aucun pair ne peut avoir une vue globale du système), de nouveaux algorithmes

doivent être explorés. Par exemple, l'ajout d'un lien d'accointance entre deux pairs peut ne pas être possible (seulement des liens d'accointances existant déjà peuvent être utilisés). De plus, en toute généralité, le contexte distribué ne se restreint pas aux systèmes d'interactions (c-à-d systèmes où deux pairs qui partagent une variable commune doivent être directement reliés par le réseau) comme c'est le cas dans les travaux de CSPs distribués.

Dans ce chapitre, nous proposons d'explorer le cas général, c'est à dire lorsque les liens entre les pairs ne sont pas dépendants du vocabulaire partagé. Aussi nous chercherons à satisfaire une politique d'accès particulière où les variables locales restent confidentielles aux pairs où elles apparaissent. Dans ce contexte, nous proposons un nouvel algorithme de décomposition arborescente distribuée, basé sur des élections locales avec passage d'un jeton. L'utilisation d'un jeton est nécessaire pour éviter que des décisions concurrentes soient prises. Nous concluons ce chapitre par une analyse expérimentale de notre algorithme sur les familles de réseaux petit monde, et montrons que les arbres joints produits sont nettement meilleurs que l'état de l'art des algorithmes distribués, tout en s'adressant à des réseaux plus généraux.

5.1 Décomposition Arborescente Distribuée (DAD)

Dans cette section nous rappelons tout d'abord la définition de décomposition arborescente définie par Robertson et Seymour. Nous montrons que cette définition ne peut s'appliquer à notre contexte distribué où il est nécessaire de respecter les liens d'accointances du système ainsi que la confidentialité du vocabulaire local.

5.1.1 La décomposition arborescente en centralisé

Introduit par Robertson et Seymour [RS86], la décomposition arborescente a été appliquée dans de nombreux problèmes. Par exemple, elle a été notamment appliquée au diagnostic de circuits électroniques par Fattah et Dechter [FD95]. Dans ce travail, un graphe modélise l'ensemble des équations décrit par un circuit. Chaque noeud du graphe est étiqueté par une variable, et chaque arête modélise un lien sémantique entre les variables apparaissant dans une même équation.

La Figure 5.1 gauche montre un problème décrit par une conjonction de formules Φ_i (une formule peut représenter par exemple une contrainte, une propriété, ou encore dans notre cadre du diagnostic la description d'un composant d'un sous-système). Sur la droite, nous montrons le graphe d'interactions correspondant où chaque noeud est étiqueté par une variable et où il existe un lien entre deux noeuds si les variables qui les étiquettent apparaissent dans une même formule. Par exemple, en gris clair, nous entourons les parties du graphe d'interactions représentant Φ_1, Φ_2 et Φ_3 du problème initial.

La décomposition arborescente d'un graphe d'interactions est un arbre de clusters de variables respectant la running intersection et préservant les liens de dépendances entre les variables. Nous rappelons ici la définition de Robertson et Seymour [RS86] en gardant à l'esprit qu'un sommet du graphe initial correspond à une variable.

Définition 5.1 (Décomposition arborescente[RS86]) *Une décomposition arborescente d'un graphe G est un couple (χ, T) pour laquelle $T = (CT, F)$ est un arbre où F modélise l'inter-dépendance entre les clusters $ct_i \in CT$. $\chi = \{\chi_{ct_i} : ct_i \in CT\}$ est un ensemble de sous-ensembles de sommets de (G) t.q. chaque χ_{ct_i} représente l'ensemble des sommets du cluster ct_i. La décomposition arborescente satisfait les propriétés suivantes :*

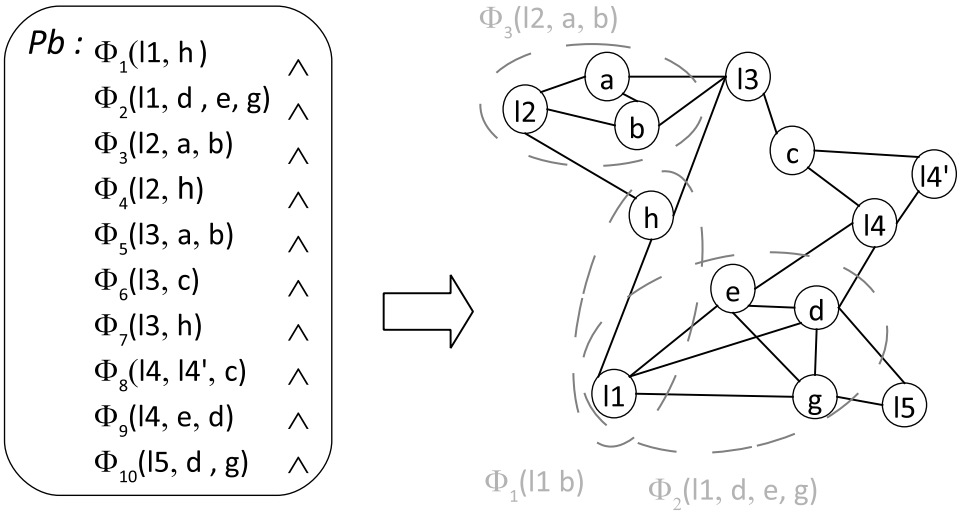


FIGURE 5.1 – Description d'un problème en centralisé (gauche), son graphe d'interactions (droite)

1. $\bigcup_{ct_i \in CT} \chi_{ct_i} = sommets(G)$,
   Tout sommet du graphe initial apparaît dans au moins un des clusters. Un cluster ne contient pas de nouveau sommet.
2. $\forall \{x, y\} \in G, \exists ct_i \in CT$ *t.q.* $\{x, y\} \in \chi_{ct_i}$
   Chaque paire de variables connectées par une arête dans le graphe initial se retrouve dans au moins un des clusters ***(conformité des dépendances)****.*
3. $\forall ct_i, ct_j, ct_k \in CT$, *si* ct_k *est sur le chemin de* ct_i *à* ct_j *dans* T, *alors* $\chi_{ct_i} \cap \chi_{ct_j} \subseteq \chi_{ct_k}$,
   Deux clusters contenant un même sommet sont connectés par d'autres clusters qui contiennent aussi ce sommet ***(running intersection)****.*

L'arbre de clusters résultant de la décomposition arborescente est aussi appelé arbre joint (il vérifie la propriété de running intersection). En plus, cet arbre vérifie (par la conformité des dépendances) que toutes les variables apparaissant dans une même formule et liées à travers le graphe d'interactions, apparaissent aussi dans un même cluster. Cette propriété nous permettra de redistribuer chaque formule du problème initial vers les clusters de l'arbre joint.

La Figure 5.2 représente la décomposition arborescente du graphe d'interactions de la Figure 5.1. On peut vérifier que chaque sommet étiqueté par une variable dans le graphe d'interactions appartient à au moins un des clusters de la décomposition arborescente. De plus on peut noter que l'ensemble des clusters $\{ct_2, ct_5, ct_7, ct_6, ct_3\}$ contenant $l1$ est connecté à travers un chemin. Ainsi la running intersection est satisfaite pour $l1$. Il est facile de vérifier que la running intersection est satisfaite pour toutes les autres variables.

La représentation graphique usuelle d'un problème par son graphe d'interactions et l'opération de décomposition évoquée ci-dessus ne modélisent que les interdépendances sémantiques de variables apparaissant dans une même formule. Cette représentation graphique n'est pas adaptée à notre contexte distribué. En effet, la décomposition arborescente telle que définie ci-dessus ne permet pas de vérifier que l'arbre de clusters résultat respecte les accointances des pairs ou

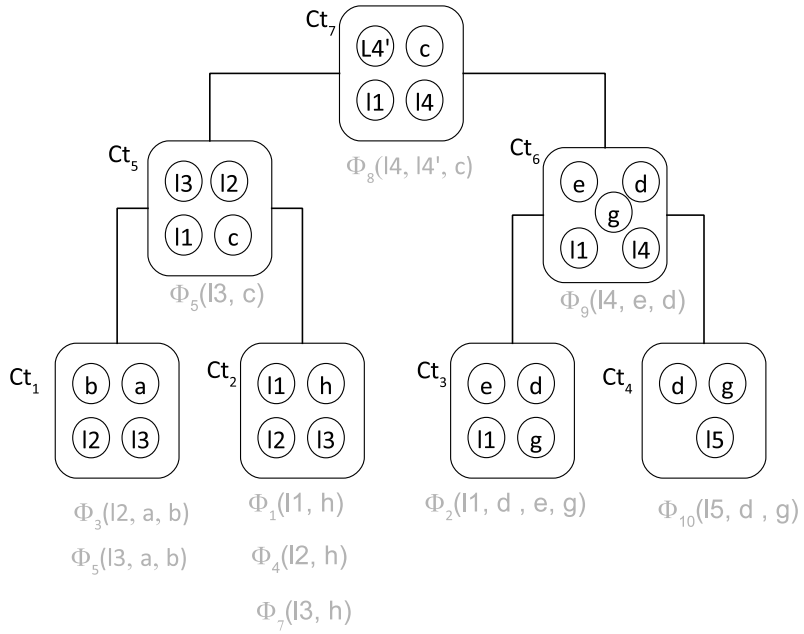


FIGURE 5.2 – Décomposition arborescente de la Figure 5.1

encore la confidentialité des variables partagées. Il est donc nécessaire d'adapter cette définition à notre contexte distribué.

5.1.2 La décomposition arborescente distribué avec confidentialité

Dans notre contexte distribué, chaque pair connaît un sous-ensemble de formules auxquelles il a accès et peut interagir avec ses voisins dans le but de résoudre un problème global. Nous rappelons que nous avons représenté par un système accessible une description du système global auquel chaque pair a accès. Lors de la caractérisation des systèmes accessibles garantissant un diagnostic distribué correct, nous avons modélisé à travers le schéma d'un système accessible à la fois les liens d'accointances et la répartition du vocabulaire auquel chaque pair a eu accès.

La Figure 5.3 (à droite) montre un exemple de schéma du système distribué (à gauche). On notera par exemple que p_1 et p_3 ont en commun la variable h sans pour autant la partager dans le réseau par un lien direct comme ce serait le cas dans un graphe d'interactions ou son graphe dual. Sans perte de généralité nous considérons que deux pairs voisins dans le schéma partagent au moins une variable.

C'est à partir du schéma d'un système que nous adaptons la définition de décomposition arborescente introduite par Robertson et Seymour [RS86] pour l'appliquer au contexte distribué.

Définition 5.2 (Décomposition arborescente distribué (DAD)) *Soit $G((P,V),ACQ)$ le schéma d'un système distribué, un arbre de cluster $T((CT,\chi),F)$ t.q. chaque cluster $ct \in CT$ est étiqueté par un ensemble de variables $\chi(ct)$, est une décomposition arborescente distribuée de G ssi il existe $\gamma : CT \to P$ (qui représente la provenance des clusters) t.q. :*

1. $\bigcup_{p\in P} V(p) = \bigcup_{ct\in CT} \chi(ct)$ ***(conformité du vocabulaire)***
2. $\forall p \in P, \exists ct \in CT$ *t.q.* $V(p) \subseteq \chi(ct)$ ***(conformité des dépendances)***.

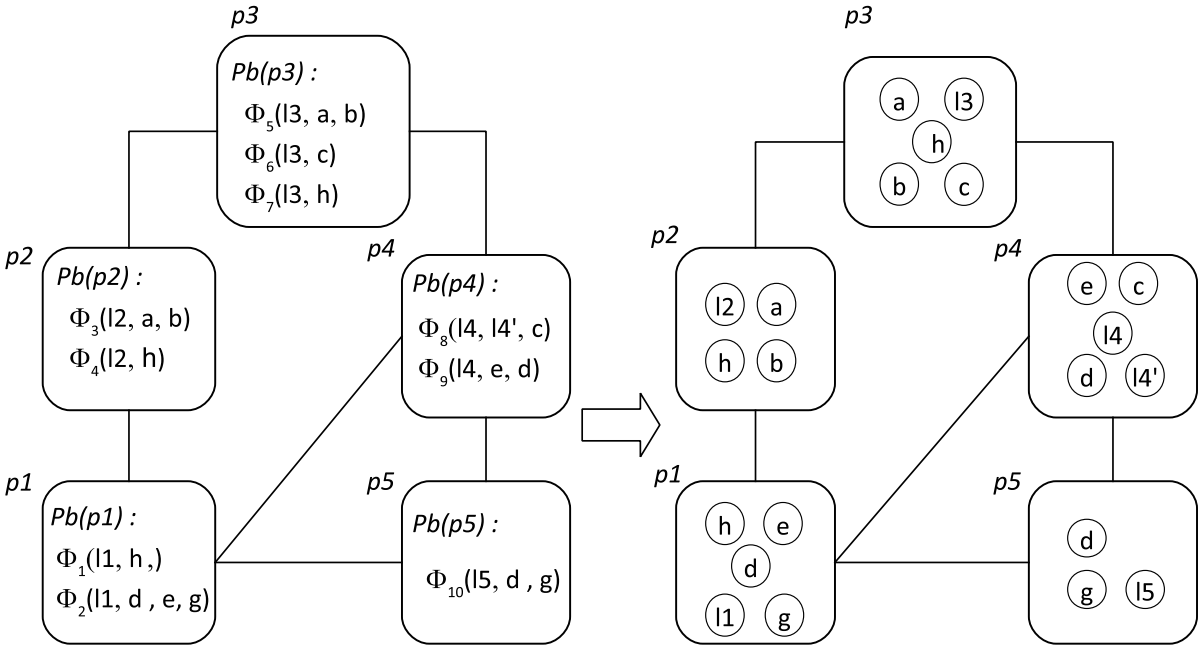


FIGURE 5.3 – Gauche : système distribué de la Figure 5.1. Droite : Son schéma

3. $\forall ct, ct', ct" \in CT$, *si* ct' *est sur le chemin de* ct *à* $ct"$ *dans* T, *alors* $\chi(ct) \cap \chi(ct") \subseteq \chi(ct')$ ***(running intersection)***.
4. $\forall \{ct, ct'\} \in F$:
   – $\gamma(ct) = \gamma(ct')$ *ou*
   – $\{\gamma(ct), \gamma(ct')\} \in ACQ$
   Chaque cluster est créé par un pair. Deux clusters voisins proviennent du même pair ou de pairs voisins dans le schéma. ***(conformité des accointances)***
5. $\forall ct \in CT, \forall v \in \chi_{ct} : K(\gamma(ct), v) = vrai$. *Les variables d'un cluster respectent la politique d'accès du pair contenant le cluster.* ***(conformité de la politique d'accès)***

Dans la définition ci-dessus Les trois premières propriétés de la décomposition arborescente distribuée sont similaires aux propriétés de la définition 5.1 de Robertson et Seymour. La décomposition arborescente distribuée d'un schéma conserve son vocabulaire, les dépendances des variables liées à travers le vocabulaire de chaque pair et la running intersection. Nous avons aussi besoin d'ajouter deux autres propriétés pour adapter la DAD à notre contexte distribué. La conformité des accointances correspond à la propriété suivante : un cluster est créé par un unique pair et un pair crée au moins un cluster (si sa formule n'est pas vide) et possiblement plusieurs. Ici un cluster n'est pas abstrait, il peut être vu comme un ensemble de variables localisées chez un pair. De plus, toute interaction entre clusters suit le graphe d'accointances du schéma. Notre politique d'accès est exprimée à travers la cinquième propriété. Dans notre modélisation, un cluster n'est pas une unité abstraite. C'est un ensemble de variable provenant possiblement de plusieurs pairs mais localisé chez un unique pair à partir duquel le cluster respecte la politique d'accès. Ainsi un cluster hébergé par un pair respecte la politique d'accès de ce pair. Dans la suite nous nous intéressons à une DAD suivant une politique d'accès particulière qui préserve la confidentialité des variables locales.

Définition 5.3 (Politique de confidentialité des variables locales) *Une politique d'accès $K : P, V \to \{\top, \bot\}$ respecte la confidentialité des variables locales ssi pour toute variable $v \in V$:*

1. *soit v est une variable locale (c-à-d il existe un unique pair p t.q. : $K(p, v) = \top$) ou*
2. *soit v est une variable partagée (c-à-d pour tout p : $K(p, v) = \top$)*

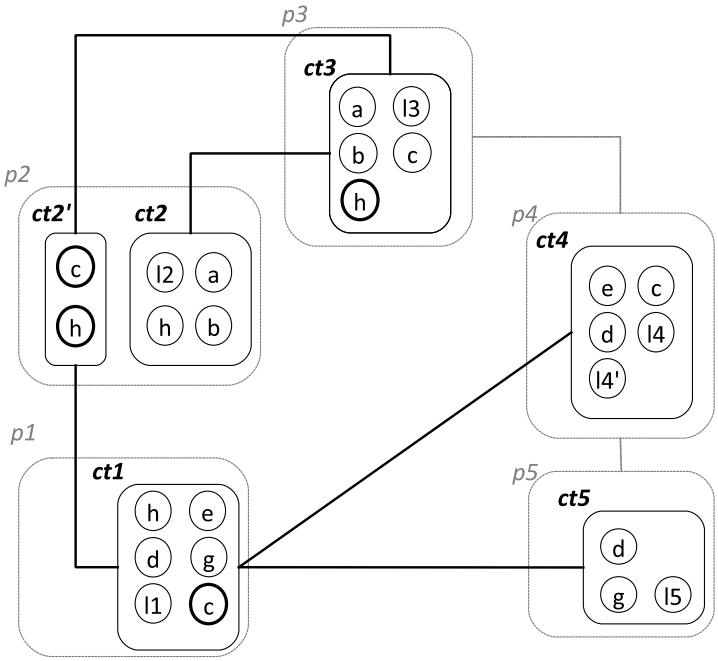


FIGURE 5.4 – Une décomposition arborescente distribuée avec confidentialité du schéma Figure 5.3

On remarquera que, dans la Figure 5.3, certaines variables particulières (notées l_i), apparaissent uniquement dans le vocabulaire de certains pairs. Nous utilisons cette notation pour représenter ces variables comme locales, les autres variables comme étant partagées. La propriété de confidentialité s'assure que les variables locales restent locales (c-à-d aucune variable locale ne peut être envoyée à un autre pair du système). Appliquée à la décomposition arborescente distribuée, notre politique de confidentialité vérifiera que tout cluster créé par un pair p pourra contenir n'importe quelle variable partagée, issue de n'importe quel pair du système mais ne pourra contenir que les variables locales du pair p.

Propriété 5.1 (DAD avec confidentialité des variables locales) *L'arbre joint $T((CT, \chi), F)$ est une décomposition arborescente distribuée respectant la confidentialité des variables locales de G ((P,V),ACQ) ssi*

1. *T est une décomposition arborescente distribuée de G où les clusters créés par chaque pair p sont représentés la fonction $\gamma(p)$.*
2. *$\forall p$, $\forall v \in V_p$:*
   (a) *si $v \notin V_{p'}$ $(p \neq p')$ alors $\forall ct$ $\gamma(ct) \neq p$ et $v \in \chi(ct)$*
   Si v est une variable locale à un pair p, tout cluster non créé par p ne contient pas la variable v.
   (b) *si $v \in V_{p'}$, $p \neq p'$ alors $\exists ct$ $v \in \chi(ct)$*
   Si v est une variable partagée entre p et p', elle apparaît dans un cluster quelconque.

La Figure 5.4 est une décomposition arborescente distribuée respectant la confidentialité et les accointances de la Figure 5.3. Pour tout lien entre clusters de l'arbre joint distribué, il existe un lien entre les pairs ayant créé ces clusters. Les variables l_i locales au pair p_i ne figurent que dans le vocabulaire d'un cluster créé par p_i. À l'opposé, les variables partagées t.q. c et f garantissent la running intersection. La Figure 5.4 respecte la confidentialité alors qu'une décomposition arborescente classique comme c'est le cas de la Figure 5.2 ne respecte pas la propriété de confidentialité (les variables locales de pairs distincts apparaissent dans un même cluster). De plus, tout comme pour la décomposition arborescente, on pourra vérifier pour une DAD respectant la confidentialité que toutes les formules d'un système sont couvertes par le vocabulaire des clusters de l'arbre joint distribué résultat.

5.2 Décomposer à travers un arbre couvrant distribué

À notre connaissance, il n'existe pas ou peu de travaux s'étant attaqués à la problématique de décomposition arborescente distribuée respectant à la fois les accointances des pairs et la confidentialité des variables locales. Toutefois, il est facile de constater que le respect des accointances du schéma d'un système peut être obtenu par un arbre distribué couvrant l'ensemble des pairs du système. À partir de cet arbre couvrant, on propagera certaines variables partagées afin de respecter la running intersection tout en garantissant la confidentialité des variables locales. C'est ce processus (la propagation Xor) que nous décrivons dans cette section. Tout d'abord nous présentons les protocoles de construction d'arbre couvrant distribué qui ont servi à la résolution de problèmes difficiles tq (DCSP, DCOPS [PF05]) et qui implicitement construisent un arbre joint distribué. Ensuite, à partir d'un arbre couvrant distribué nous présentons un nouvel algorithme la Propagation XOR capable de construire un arbre joint distribué respectant la confidentialité du vocabulaire et les accointances des pairs.

5.2.1 Construction d'un arbre couvrant distribué

Il existe un lien étroit entre la recherche d'une bonne décomposition arborescente d'un problème et la recherche d'un bon ordre d'élimination de variables, ou encore une bonne triangulation [Gol80, JJ94, JNT05]. Cependant, dans un contexte distribué, seulement un ordre d'élimination basé sur des méthodes d'exploration simples t.q. DFS (recherche en profondeur d'abord) [PF05] et BFS (recherche en largeur d'abord) [EBBB08] ont été proposées. Dans la suite nous présentons différents protocoles distribués qui à partir d'un graphe d'accointances $G(P, ACQ)$ construisent un arbre couvrant distribué $T(P, E)$ t.q pour tout arc $(parent, fils)$ de E il existe une arrête $\{parent, fils\} \in ACQ$. On dira que p' est le *voisin* de p ssi $\{p', p\} \in ACQ$. On dira que p est le *parent* de f ssi $(p, f) \in E$. L'arbre couvrant distribué $T(P, E)$ est induit par un ensemble de liens parents de chaque pair de P. Nous illustrons chaque protocole de construction d'arbre distribué par un arbre joint. Cet arbre joint est le résultat de l'exécution du protocole distribué Propagation Xor (présenté section 5.2.2) sur l'arbre couvrant $T(P, E)$.

5.2.1.1 Le flooding

L'algorithme de *flooding* est la fois la méthode de construction d'arbre couvrant la plus simple mais aussi la plus rapide. Initialement un pair initiateur diffuse un message $EXPLORATION$ à tous ses voisins. Un pair qui reçoit un message $EXPLORATION$ pour la première fois d'un voisin le diffuse à son tour à ses autres voisins. La diffusion progressive du message d'$EXPLORATION$ inonde le réseau et atteint tous les pairs du système. Dans le but de

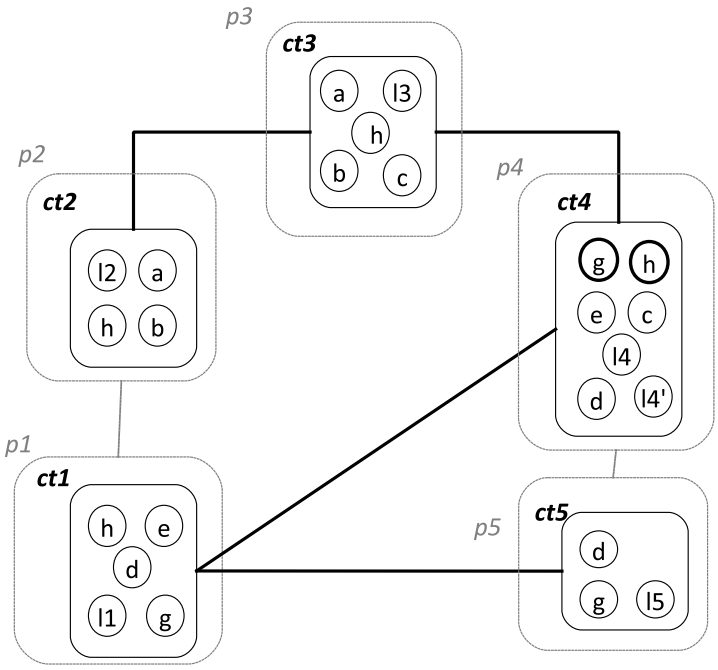


FIGURE 5.5 – Arbre joint distribué du schéma Figure 5.3 (Flodding et Propagation Xor)

construire un arbre couvrant, tout pair différent du pair initiateur enregistre le premier voisin dont il reçoit un message comme parent. Lorsque le flooding aura atteint tout le système, chaque pair aura enregistré un parent. L'ensemble des liens parents enregistrés par chaque pair formera ainsi un arbre couvrant distribué enraciné au pair initiateur.

Algorithme 1: Flooding
Entrée: un graphe d'accointances $G(P, ACQ)$
Sortie: un arbre distribué $T(P, E)$ induit par les liens parents
(1) p **reçoit** $EXLORATION$ **d'un voisin** p'
(2) **if** $parent = null$
(3) $parent \leftarrow p'$
(4) envoie $EXLORATION$ aux autres voisins

En toute généralité, sous l'hypothèse d'une transmission de message en temps uniforme, un parcours par flooding permet une exploration rapide du réseau, la complexité en temps est de l'ordre du diamètre D du réseau. La complexité en nombre de messages est de l'ordre du nombre de liens $|ACQ|$ du réseau. Si un pair recevant un message $EXPLORATION$ répond à l'expéditeur, la terminaison pourra être détectée lorsque chaque pair aura reçu un moins un message de tous ses voisins.

La Figure 5.5 représente une possible décomposition arborescente induite par une exploration par inondation et par la propagation XOR que nous présentons par la suite (section 5.2.2). Dans cette figure les arêtes mises en gras mettent en relief l'arbre construit à partir du pair p_1. Nous reviendrons dans la suite sur la construction d'un arbre joint à partir d'un arbre couvrant. L'arbre couvrant issu d'une exploration par inondation est très sensible au temps de communication des différents canaux constituant les liens d'accointances du réseau. Dans la Figure 5.5 comme le canal de communication entre p_1 et p_2 est lent, le parcours par inondation a le temps d'explorer les liens (p_1, p_4), (p_4, p_3), (p_3, p_2) avant que le message d'exploration lancé sur

le lien (p_1, p_2) n'arrive à p_2. Si le temps de communication des liens d'accointances est identique alors l'exploration par inondation est similaire à un parcours distribué de BFS à travers le réseau. Nous détaillerons ce parcours à la section suivante. La première qualité d'une construction d'un arbre joint par un protocole d'inondation est la rapidité. C'est pourquoi nous avons suivi ce protocole dans nos premiers travaux sur le diagnostic de systèmes distribués (cf. Armant et al [ADS08]). Toutefois, nous nous sommes vite rendu compte que la qualité de la décomposition arborescente ne nous permettait pas de diagnostiquer des systèmes de pairs de taille importante. En effet, bien que rapide, l'arbre induit par une exploration par inondation ne tient pas du tout compte de la largeur arborescente de l'arbre joint distribué en cours de construction. Or la largeur arborescente un paramètre déterminant qui nous permet de passer à l'échelle.

5.2.1.2 BFS distribué

Dans le cadre de la résolution de COP distribués, les travaux de Elzahir et al [EBBB08] orientent la recherche de solutions par la construction d'un arbre distribué grâce à un parcours en largeur d'abord. Dans ce cas, la décomposition arborescente est implicite et conduit le raisonnement distribué des feuilles vers la racine pour le calcul d'une solution optimale. D'autre part, les travaux de GrunBach et Wu [GW10] appliquent la décomposition arborescente à la vérification de programmes, toujours en se basant sur un parcours BFS.

Dans la littérature des algorithmes distribués, l'algorithme BFS se généralise par la recherche du plus court chemin à partir d'une racine. Nous rappelons les principes de la recherche du plus court chemin proposé par Dijkstra [Dij59]. Nous détaillons ensuite l'algorithme de Bellman-Ford [Bel58, RB62] qui est plus rapide mais nécessite plus de messages que Dijkstra.

Les algorithmes distribués dérivés de l'approche de Dijkstra, de construction d'arbres sont gloutons et itératifs. Initialement un pair est choisi comme racine. À la première itération les voisins de la racine l'ajoutent comme parent. À l'itération d, sont ajoutés à l'arbre en construction les pairs à distance d de la racine. En distribué, le passage du niveau d à $d+1$ se traduit par un retour arrière vers la racine.

L'algorithme de Bellman-Ford calcule la distance la plus courte à la racine par inondation sur tous les chemins. Le meilleur prédécesseur devient le parent dans l'arbre. Chaque pair maintient une variable d_{min} enregistrant la distance minimum reçu. d_{min} est initialisée à ∞. Initialement le pair racine envoie 1 a tous ses voisins.

Algorithme 2: BFS Distribué
Entrée: un graphe d'accointances $G(P, ACQ)$
Sortie: un arbre distribué $T(P, E)$ induit par les liens parents
(1) p **reçoit** d **d'un voisin** p'
(2) **if** $d < d_{min}$
(3) $d_{min} \leftarrow d$
(4) $parent \leftarrow p'$
(5) envoie $d_{min} + 1$ aux autres voisins

À la figure 5.6 nous illustrons la construction d'un arbre joint induit par un parcours BFS et la Propagation Xor (cf section 5.2.2). Les liens de l'arbre couvrant construit sont mis en gras. Nous constatons que les chemins de l'arbre représentent pour chaque pair la plus courte distance d'un pair à la racine (p_1). L'algorithme Bellman-Ford s'exécute en O(diamètre). Le diamètre d'un graphe est la plus grande distance séparant deux sommets du graphe. Ce temps d'exécution est moindre que l'algorithme DFS Distribué (que nous présenterons à la section suivante) et est particulièrement rapide pour les réseaux petit monde qui se caractérisent par

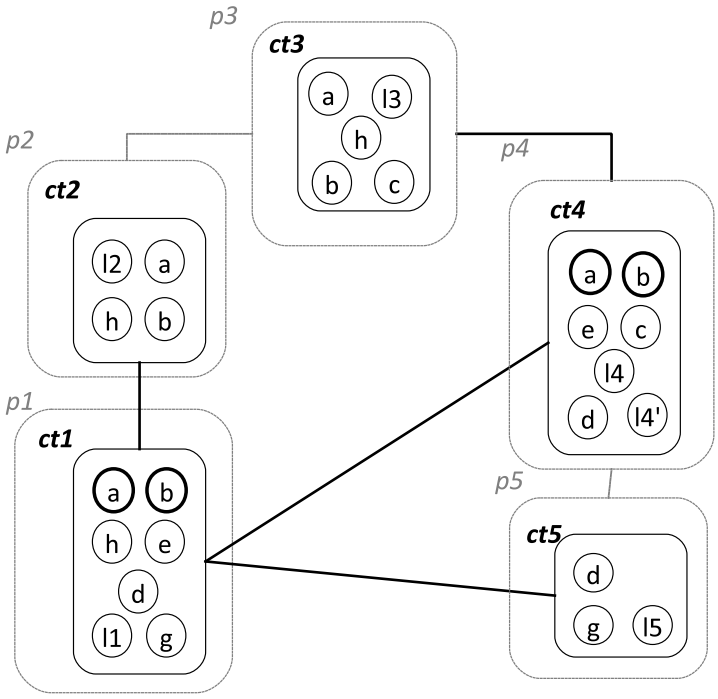


FIGURE 5.6 – Arbre joint distribué du schéma Figure 5.3 (BFS et Propagation Xor)

un faible diamètre. Néanmoins la complexité en nombre de messages est en $O(|V| * |E|)$. Tout comme l'exploration par inondation, le parcours BFS distribué construira très rapidement une décomposition arborescente à travers un arbre distribué. Cependant, nous avons constaté expérimentalement que la qualité des décompositions arborescentes induite par BFS pouvait être significativement améliorée.

5.2.1.3 DFS distribué

Toujours dans le domaine de l'optimisation de contraintes distribué, l'algorithme DFS distribué est la référence permettant d'ordonner les différents agents chargés de la résolution des problèmes d'optimisation. Il est utilisé dans [PF05, FLP08] afin de structurer le réseau de contraintes. Ce protocole visite un à un tous les pairs du réseau. Initialement un pair est choisi comme *racine* de l'arbre. Il est *visité*. La racine choisit un voisin et lui envoie un message $FORWARD$. À la réception d'un message $FORWARD$, si le pair n'a pas encore été *visité* (c-à-d : n'a jamais reçu de message $FORWARD$ ligne 3), il enregistre l'expéditeur comme parent et transmet le message $FORWARD$ à un voisin non encore visité présent dans la liste *voisinsNonVisités*. Si le pair a déjà été visité il envoie un message *Return* à son parent. L'ensemble *voisinsNonVisités* comprend tous les voisins p' de p pour lesquelles p n'a reçu ni de messages $FORWARD$ ni de messages $RETURN$.

Contrairement aux méthodes de Flooding et BFS présentées précédemment, il est possible d'orienter le protocole distribué de DFS afin d'améliorer la qualité de l'arbre joint. Cette orientation s'effectue par la primitive *prochain* qui sélectionne parmi les voisins non visités le prochain noeud à visiter. Nous avons testé différentes heuristiques, celle qui retourne de meilleurs résultats est connue sous le nom de MCS (*Max Cardinality Set*). Appliquée à notre contexte distribué elle consiste à orienter l'exploration vers le pair ayant le plus large vocabulaire.

Algorithme 3: DFS Distribué
Entrée: un graphe d'accointances $G(P, ACQ)$
Sortie: un arbre distribué $T(P, E)$ induit par les liens parents
(1) p **reçoit** $FORWARD$ **d'un voisin** p'
(2) retire p' de *voisinsNonVisités*
(3) **if** non visité
(4) $parent \leftarrow p'$
(5) **if** $voisinsNonVisits \neq null$
(6) envoie $FORWARD$ à *prochain(voisinsNonVisités)*
(7) **else**
(8) envoie $RETURN$ à p'
(9) **else**
(10) envoie $RETURN$ à p'

(12)p **reçoit** $RETURN$ **de** p'
(13) retire p' de *voisinsNonVisités*
(14) **if** $voisinsNonVisits \neq null$
(15) envoie $FORWARD$ à *prochain(voisinsNonVisités)*
(16) **else**
(17) **if** p est racine
(18) fin
(19) **else**
(20) envoie $RETURN$ à p'

Nous illustrons à la Figure 5.7 un arbre couvrant construit par un parcours distribué DFS orienté par l'heuritique MCS. L'exécution du protocole distribué de DFS passe par tous les liens du réseau et est plus lente que les méthodes BFS ou Flooding vues jusqu'alors. Il a une complexité en temps et en nombre de messages de $2*|ACQ|$. Nous rappelons que dans un système distribué le temps se mesure en nombre de messages non concurrents. Dans une version optimisée, Sharma et al [SMI89] puis Makki et Havas [MH96] ont proposé d'enregistrer dans le message $FORWARD$ les pairs déjà visités (les ancêtres). Cette optimisation permet à DFS Distribué de s'exécuter en $O(2*|V|)$. Compte tenu de la qualité de son arbre joint nous considérons que l'algorithme *DFS Distribué* est la meilleur alternative pour la construction d'arbres joints couvrant distribués.

5.2.2 De l'arbre couvrant distribué à l'arbre joint distribué

Supposons maintenant qu'un arbre distribué $T(P, E)$ a été construit par l'un des protocoles vu précédemment (Flooding, BFS Distribué, DFS Distribué). Chaque pair p a la connaissance de son parent et des fils par les primitives $parent(p) \in P$ et $Fils(p) \subseteq P$.

Dans un contexte centralisé, il existe différents protocoles qui, étant donné un ordre sur les variables, sont capables de construire une décomposition arborescente d'un système. C'est par exemple le cas de l'algorithme Bucket Elimination[Dec99] ou plus particulièrement du protocole JTC de Dechter [Dec06] qui construit une décomposition arborescente des feuilles vers la racine. Cet algorithme que nous détaillerons dans la section suivante élimine une à une chaque variable du problème et construit simultanément les clusters de l'arbre joint résultat. Une variable éliminée n'apparaît pas dans les nouveaux clusters créés. Cette décomposition arborescente se base sur un graphe d'interactions pour lequel il a été possible de lier chaque variable du problème initial à toutes les autres variables avec qui elle apparaît dans une même fonction. Or, dans

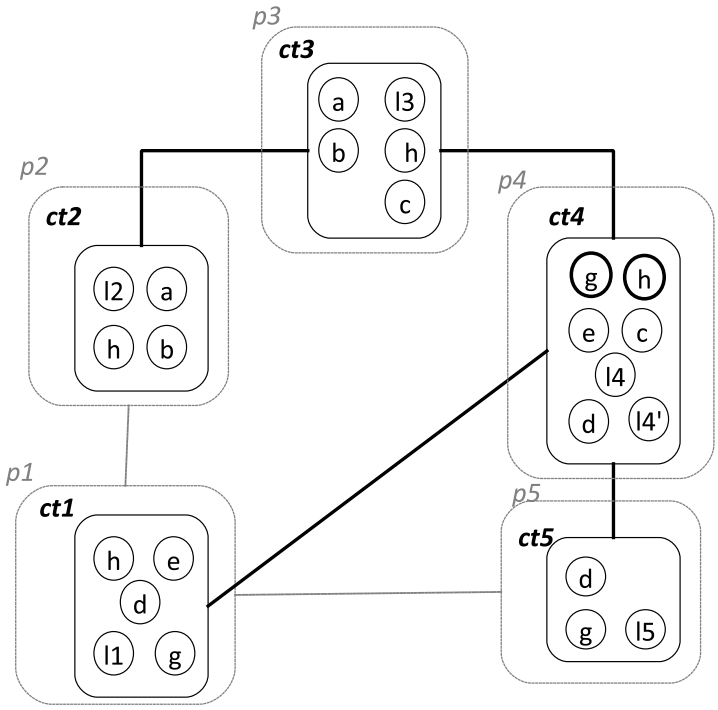


FIGURE 5.7 – Arbre joint distribué du schéma Figure 5.3 (DFS - MCS et Propagation Xor)

un contexte distribué, une même variable pourra figurer dans la description de pairs différents n'étant pas nécessairement connectés par un lien d'accointance. En raison des accointances et de la confidentialité, il ne sera pas toujours possible de construire le graphe d'interactions des pairs. Pour contourner cet obstacle nous introduisons un nouvel algorithme : la $ProjectionXor$ qui, à partir du vocabulaire partagé et des accointances de chaque pair d'un schéma $G((P, V), ACQ)$, construit un arbre joint distribué $T((P, \chi), E)$ à partir de l'arbre distribué $T(P, E)$ tout en respectant la confidentialité et les accointances de G. Nous rappelons que le schéma $G((P, V), ACQ)$ d'un système reprend les informations de son graphe d'accointances $G((P, ACQ)$ pour lequel chaque pair $p \in P$ est associé à un vocabulaire $V(p)$.

De façon générale, la Propagation XOR consiste à propager uniquement des variables des feuilles vers la racine d'un arbre déjà construit ou en construction afin que la répartition des variables respecte la *running intersection* et que l'on puisse obtenir un arbre joint. Ce protocole garde les variables locales locales. Afin d'arrêter automatiquement la propagation des variables partagées (car aucun pair n'a la vue du schéma global) nous introduisons le XOR de couples (variable, arête) : $(v, \{p, p'\})$ où v est une variable partagée entre p et un voisin p' t.q. $a = \{p, p'\}$. Intuitivement, le XOR entre deux ensembles de couples (variable, arête) $sh\ sh'$ ne retourne que les couples apparaissant uniquement dans l'un ou l'autre des ensembles sh et sh'. Les autres couples sont filtrés. C'est par cette opération que nous pouvons arrêter la propagation des variables partagées. Plus formellement, initialement chaque pair p maintient une structure $sh(p)$, "shared de p", définie par un ensemble de couples (variable, arête) t.q. $sh(p) = \{(v, \{p, p'\})$ t.q. $v \in V(p) \cap V(p')), \{p, p'\} \in ACQ\})$. En pratique une arête a pourra être encodée par un numéro unique afin de préserver la confidentialité des pairs auxquelles elle est rattachée. Notons $vars(sh) = \{v | (v, a) \in sh\}$ l'ensemble des variables de sh. Initialement $vars(sh)$ représente aussi l'ensemble

des variables partagées de p. Lors de l'exécution de la propagation Xor et de la structure $sh(p)$ sera enrichie d'autres variables partagées de façon à représenter les variables partagées du sous-arbre enraciné en p. Les variables locales à p ne sont partagées par aucun lien d'accointance (p, p').

À toutes les étapes de la propagation XOR aucune variable locale n'apparaitra dans $sh(p)$.

C'est cette dernière remarque qui nous permettra de garantir que la construction de la décomposition arborescente respecte la confidentialité des variables locales.

Toutefois, dans le but de satisfaire la *running intersection*, il sera nécessaire de propager certaines variables partagées à travers les liens de l'arbre couvrant $T(P, E)$ précédemment construit par l'un des algorithmes vu à la section précédente. Pour cela, nous introduisons la Propagation Xor de couples (variable, arête). Comme son nom l'indique cette propagation se base sur une opération Xor (différence symétrique) que nous définissons tout de suite.

Définition 5.4 (Xor de couples (variable, arête)) *Soient sh et sh′ deux ensembles de couples (variable, arête) on définit le Xor de sh et de sh′ par l'opération suivante :*

$$sh \oplus sh' = \{(v, a) \in (sh \cup sh') \setminus (sh \cap sh')\} \tag{5.1}$$

L'opération Xor entre sh et sh' retient tous les couples (v, a) provenant exclusivement de sh ou de sh'. Il supprime de l'ensemble résultat tous les couples (v, a) apparaissant à la fois dans sh et sh'.

Nous illustrons Figure 5.8 l'opération Xor appliquée aux structures "sh" (variable, arêtes) des pairs $p2$ et $p3$. Supposons ici que le pair $p2$ est le fils du pair $p3$ dans l'arbre T. $sh(p2)$ (resp. $sh(p3)$) dénote l'ensemble de couples (variables, arêtes) de $p2$ (resp. $p3$). $vars(sh(p2))$ représente l'ensemble des variables partagées de $p2$, de même pour $vars(sh(p3))$. $sh^T(p3)$ dénote la mise à jour des couples (variables, arêtes) de $p3$ décrivant l'opération $sh(p3) \oplus sh(p2)$. Nous remarquons par exemple que les variables $\{a, b\}$ disparaissent alors que la variable h contient encore une arête "ouverte" $(p1, p3)$ qui se fermera lors de la rencontre avec le premier ancêtre commun entre $p1$ et $p2$.

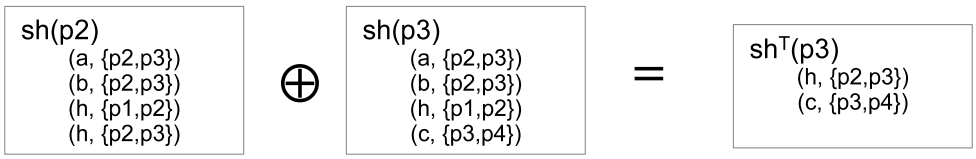


FIGURE 5.8 – Opération Xor

Maintenant que nous avons défini l'opération de base entre ensembles (variable, arête) nous décrivons la Propagation Xor permettant la construction d'un arbre joint. Initialement les feuilles (c-à-d les pairs p t.q. $Fils(p) = \emptyset$) font remonter les couples (variable, arête) vers la racine par leurs liens parents (ligne 4-5). Un pair p qui reçoit un ensemble $sh(p')$ d'un fils p' le compose par l'opération Xor avec son ensemble $sh(p)$ courant (ligne 8-9). Lorsqu'un pair a reçu l' ensemble $sh(p')$ de tous ses fils (ligne 10-11), il fait remonter son ensemble $sh(p)$ mis à jour par son lien parent (ligne 13). À cette étape, si p est l'ancêtre commun des pairs p' et p", tous les couples $(v, \{p', p"\})$ sont écartés de $sh(p)$, c'est le but de la propagation XOR(ligne 9). Les autres couples en provenance de p et de ses descendants ont été conservés. La Propagation Xor se termine localement lorsqu'un pair a reçu un message $sh(p')$ de tous ses fils p'. Elle se termine globalement lorsque la racine a reçu un message de tous ses fils.

Algorithme 4: Propagation Xor (de couples de (variable, arête))
Entrée: un arbre couvrant distribué $T(P, E)$ construit sur le schéma $G((P, V), ACQ)$
Sortie: un arbre joint distribué $T((CT, \chi), E)$
(1) *// initialisation, sh(p) lie chaque variable partagée v de p à chaque couple de pair* $\{p, p'\}$ *où elle est présente*
(2) $sh(p) \leftarrow \{(v, \{p, p'\})$ t.q. $v \in V(p) \cap V(p')), \{p, p'\} \in ACQ\}$
(3) **if** p **est une feuille de** T
(4) $\chi(p) \leftarrow V(p)$
(5) envoie $sh(p)$ à *parent*
(6)
(7) *// remontée des feuilles vers la racine*
(8) **if** p **reçoit** $sh(p')$ **d'un fils** p'
(9) $sh(p) \leftarrow sh(p') \oplus sh(p)$
(10) **if** p a reçu un msg de tous ses fils
(11) $\chi(p) \leftarrow V(p) \cup vars(sh(p))$
(12) **if** p n'est pas la racine
(13) p envoie $sh(p)$ à son *parent*

Nous montrons maintenant que le protocole Propagation Xor traverse l'arbre des feuilles vers la racine et construit la décomposition arborescente d'un schéma G à l'aide de l'arbre T, de la structure $sh(p)$, et de l'ensemble de variables $\chi(p)$. Conservons la notation $sh(p)$ pour désigner l'ensemble des couples (variable, arête) créé initialement dans chaque pair p. Notons $sh^T(p)$ la mise à jour de $sh(p)$ induite par le Xor des $sh(p')$ reçus des fils p' de p (ligne 9). On a :

$$sh^T(p) = \begin{cases} sh(p) \text{ si p est une feuille} \\ sh(p) \bigoplus\limits_{f \in Fils(p)} sh^T(f) \text{ sinon} \end{cases}$$

De même, à partir de sh^T et de la ligne 11 de l'algorithme de Propagation Xor, nous déduisons que le vocabulaire porté à la connaissance de chaque pair p se modélise par la fonction :
$\chi(p) = V(p) \bigcup\limits_{f \in Fils(p)} vars(sh^T(f))$.
$\chi(p)$ associe à chaque pair p son vocabulaire initial $V(p)$ ainsi que les variables partagées du sous-arbre de T enraciné en p.

Propriété 5.2 (*Running intersection* induite par la Propagation Xor) *Soient $T(P, E)$ un arbre couvrant distribué d'un schéma $G((P, V), ACQ)$, un couple de $(v, \{p, p'\})$ t.q. $\{p, p'\} \in ACQ$ et $v \in V(p) \cap V(p')$. v est propagée par la Propagation Xor sur le chemin reliant p à p' dans T et nul part ailleurs.*

Preuve. Soit $T(P, E)$ un arbre couvrant distribué d'un schéma joint $G((P, V), ACQ)$. Soit un couple $(v, \{p, p'\})$ t.q. $\{p, p'\} \in ACQ$ et $v \in V(p) \cap V(p')$. Soit p" le premier ancêtre commun entre p et p' dans T. Montrons que pour tout pair p_i sur le chemin de p à p", et de p' à p" on a $v \in \chi(p_i)$.
(1) Si $p_i = p$:$V(p) \subseteq \chi(p)$ d'où $v \in \chi(p)$. De façon symétrique si $p_i = p'$, $v \in \chi(p')$.
(2) Si p_i est sur le chemin de p à p' tout en étant différent de p et de p" : dans ce cas p_i est un ancêtre de p dans l'arbre orienté T. Puisque le premier ancêtre commun entre p et p' est p", aucun pair de p à p_i n'a pu effacer le couple $(v, \{p, p'\})$ par l'opération Xor. $(v, \{p, p'\})$ est donc contenu dans $sh^T(p_i)$. Or $vars(sh^T(f)) \subseteq \chi(p)$ d'où $v \in \chi(p_i)$. Par un raisonnement similaire on sait que v est contenu dans le vocabulaire de tout pair p_i sur le chemin de p' à p".
(3) Si $p_i = p$" : p" est l'ancêtre commun entre p et p' : nous avons vu en (2) que le couple

$(v, \{p, p'\})$ n'a pu être éliminé ni sur le chemin de p à p" ni sur le chemin de p' à p". p" contient donc deux fils f_1 et f_2 t.q. $(v, \{p, p'\}) \in sh^T(f_1) \cap sh^T(f_2)$. $(v, \{p, p'\})$ est donc éliminé de $sh^T(p")$ et n'apparaitra pas dans les ancêtres p". Toutefois, $vars(sh^T(f_1)) \cup vars(sh^T(f_2)) \subseteq \chi(p)$, d'où $v \in \chi(p_i)$. □

Nous venons de montrer que, par la Propagation Xor, des variables partagées entre pairs voisins d'un système sont dupliquées sur un chemin de l'arbre T de façon à ce que la *running intersection* soit respectée. Nous montrons maintenant que la propagation Xor construit un arbre joint.

Théorème 5.1 (Arbre joint induit par la Propagation Xor) *Soient $G((P, V), ACQ)$ le schéma d'un système joint distribué dont $T(P, E)$ est un arbre couvrant et χ la fonction associant à chaque pair p son ensemble de variables rencontrées lors de la Propagation Xor.*
$T((P, \chi), E)$ est une décomposition arborescente distribuée de $G((P, V), ACQ)$ respectant sa confidentialité et ses accointances.

Preuve. Soient $G((P, V), ACQ)$ le schéma d'un système joint distribué dont $T(P, E)$ est un arbre couvrant et χ la fonction associant à chaque pair p son ensemble de variables rencontrées lors de la Propagation Xor. Montrons que $T((P, \chi), E)$ est une DAD distribuée respectant la confidentialité de $G((P, V), ACQ)$. Par la ligne 4 et 11 on déduit que pour tout pair p $V(p) \subseteq \chi(p)$. Nous avons donc que $T((P, \chi), E)$ conserve tout le vocabulaire (cf. propriété 1 Définition 5.2) ainsi que les dépendances entre les variables (cf. propriété 2 Définition 5.2) de G. Nous déduisons par la propriété 5.2 et par l'hypothèse que G est joint que $T((P, \chi), E)$ est un arbre joint couvrant de G. Il respecte donc à la fois la *running intersection* (cf. propriété 3 Définition 5.2) et les accointances des pairs (cf. propriété 1 Définition 5.2). De plus, nous avons remarqué pour chaque pair p l'ensemble de couples (v,a) de $sh(p)$ ne contient que des variables partagées et aucune variable locale, $T((P, \chi), E)$ respecte aussi la confidentialité du système (pté 2 déf 5.1). $T((P, \chi), E)$ est donc une décomposition arborescente distribuée de $G((P, V), ACQ)$ respectant la confidentialité des variables locales. □

5.3 Les bonnes propriétés de la décomposition par élimination

Dans la problématique de diagnostic distribué, l'ensemble des pairs d'un système distribué a pour mission d'expliquer le comportement global du système à partir de diagnostics locaux. Dans ce cadre, nous avons montré que les schémas structurés caractérisent les systèmes de diagnostiqueurs garantissant un diagnostic distribué correct. Dans notre contexte distribué, il est donc indispensable de pouvoir décomposer un système de pairs vers un système structuré en vue de le diagnostiquer. Or, dans le cadre classique où la décomposition arborescente a été étudiée, l'objectif premier était de découper un problème initial vers un ensemble de sous-problèmes plus petits, organisés de telle façon à ce que la résolution des sous-problèmes permettent la résolution du problème initial tout en garantissant une borne maximale sur le temps et l'espace mémoire requis pour le calcul. À notre connaissance, les techniques de décomposition les plus efficaces (ayant la plus petite largeur arborescente) se basent sur un ordre d'élimination des variables du problème initial ([DGG+08, GLS00]). Même si nous avons vu précédemment que la décomposition arborescente classique ne peut garantir le respect des accointances d'un système, dans cette section, nous cherchons à comprendre les mécanismes qui font le succès de la décompo-

sition arborescente par élimination afin de s'en inspirer pour notre décomposition arborescente distribuée.

5.3.1 Décomposer à l'aide d'un ordre d'élimination

Il existe un lien étroit entre les problèmes de décomposition arborescente, la recherche d'un ordre d'élimination de noeuds et les problèmes de triangulation de graphes. Les techniques de décomposition arborescente se basant sur l'ordre d'élimination des variables tirent parti des techniques efficaces de l'ordre d'élimination pour offrir généralement des décompositions de petite largeur arborescente. Un ordre d'élimination d'un graphe G est un ordre total des noeuds du graphe induit par un processus d'élimination. Le processus d'élimination prend en entrée le graphe d'interactions des variables d'un problème $G(V, E)$ et élimine une à une toutes les variables du graphe en itérant les étapes suivantes :

1. une variable v est choisie parmi les noeuds restant du graphe,
2. des arêtes sont ajoutées entre tous les voisins non adjacents de v,
3. un cluster χ_v est créé à partir de v et de ses voisins,
4. la variable v est éliminée du graphe.

La qualité de l'ordre d'élimination est mesurée par la taille du plus grand cluster χ_v induit lors de l'élimination des variables (étape 3). On remarquera que tout lien ajouté entre deux voisins v' et v" de v à l'étape 2, entrainera aussi une augmentation de la taille des clusters $\chi_{v'}$ et $\chi_{v"}$ lorsque ces dernières seront éliminées à leur tour. Un ordre d'élimination est parfait si à l'étape 2 aucune arête n'est ajoutée. Un ordre d'élimination est optimal si parmi tous les autres ordre d'élimination le nombre d'arêtes ajoutées induit par l'élimination est minimal. Trouver un ordre d'élimination optimal est un problème NP-Difficile. Ainsi, de nombreuses heuristiques ont été proposées pour guider le choix de la prochaine variable à éliminer (étape 1) et ainsi se rapprocher de l'ordre d'élimination optimal. Les heuristiques les plus connues pour guider le choix de la prochaine variable à élimer (étape 1) sont :

- (*Max Cardinality Set*) cette heuristique consiste à éliminer en premier les noeuds de degré maximal.
- (*Min Cardinality Set*) cette heuristique consiste à éliminer en premier les noeuds de degré minimal.
- (*Min Fill In*) cette heuristique consiste à éliminer en premier les noeuds auxquels il faut ajouter un nombre d'arêtes minimum entre voisins non adjacents afin qu'ils forment une clique.

Nous remarquerons que le nombre minimal d'arêtes à ajouter entre les voisins non adjacents de v afin qu'ils forment une clique correspond aussi au nombre d'arêtes à ajouter à l'étape 2. Comme le montre les travaux de Jégou et al [JNT05] en pratique, l'heuristique *Min Fill In* retourne de meilleurs ordres d'élimination que les autres heuristiques dans la plupart des cas.

À la Figure 5.9 nous décrivons le processus d'élimination guidé par l'heuristique (*Min Fill In*). Initialement chaque noeud du graphe d'interactions est valué par le nombre d'arêtes à ajouter à ses voisins non adjacents afin que le noeud et ses voisins forment une clique. Les noeuds l_3 et l_5 ont la valuation minimale : 0. On élimine tout d'abord l_3 et on met à jour la valuation de son unique voisin e. Le cluster induit par l'élimination de l_3 est représenté à droite. Il se compose de l_3 et de e.

À la Figure 5.10, le noeud l_5 a maintenant la valuation minimale du graphe. On élimine l_5 et on met à jour les valuations de ses voisins a et b. Le cluster induit par l'élimination de l_5 est composé de l_5 et de son voisinage l_5, a, b.

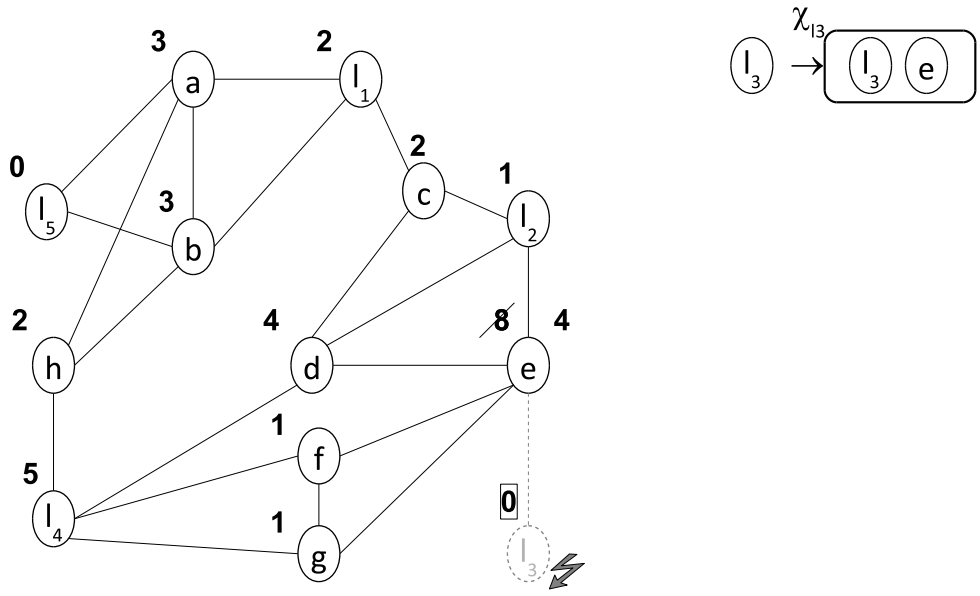


FIGURE 5.9 – Ordre d'élimination - heuristique Min Fill In, élimination de l_3

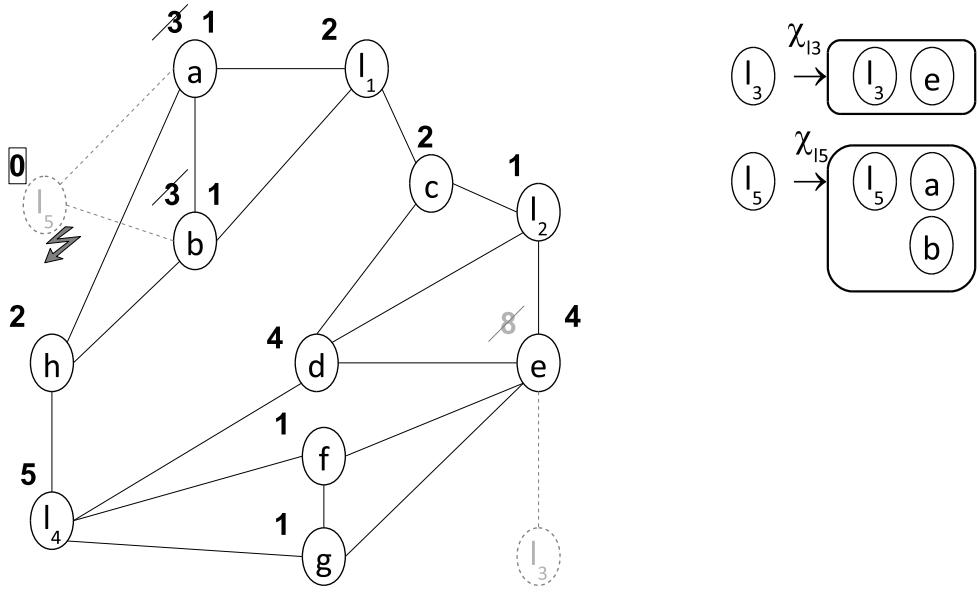


FIGURE 5.10 – Ordre d'élimination - heuristique Min Fill In, élimination de l_5

À la Figure 5.11 le noeud a a maintenant la valuation minimale du graphe. On ajoute l'arête entre les noeuds h et l_1 qui sont des voisins non adjacents de a. On élimine a et on met à jour les valuations de ses voisins l_1 et b. Le cluster induit par l'élimination de a est composé de a et de son voisinage a, b, h, l_1.

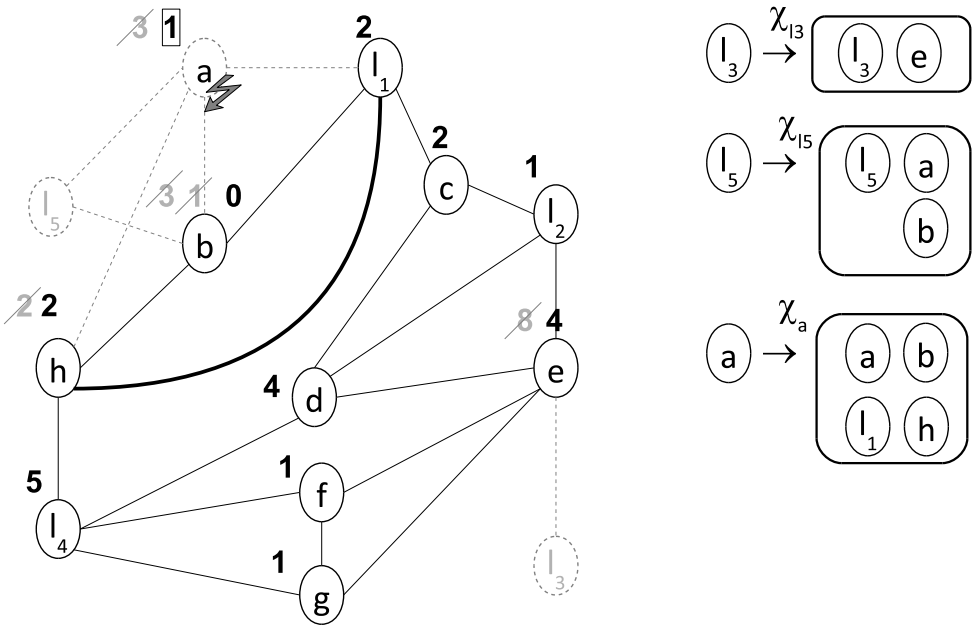
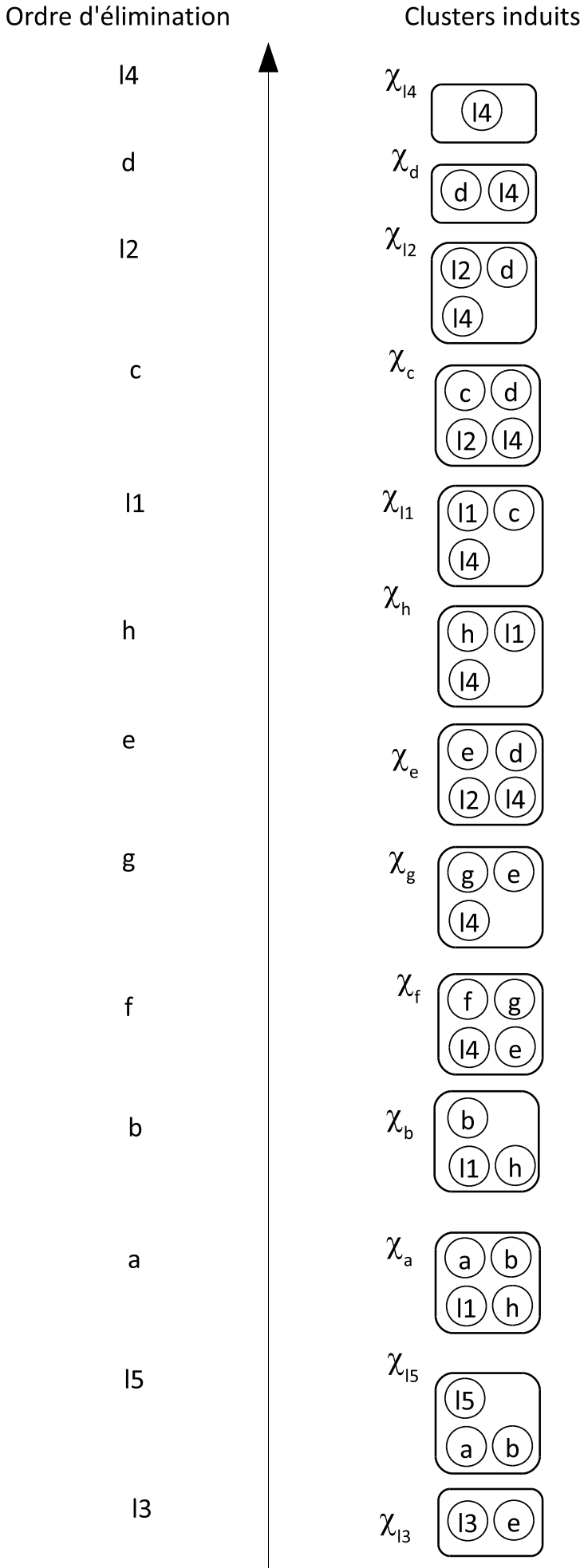


FIGURE 5.11 – Ordre d'élimination - heuristique Min Fill In, élimination de a et ajout de (h, l_1)

En continuant le processus d'élimination, les variables du graphe sont éliminées une à une en fonction de leur score. À la Figure 5.12 (de bas en haut), nous illustrons l'ordre d'élimination des variables à gauche, et le cluster induit par leur élimination à droite.


Ordre d'élimination
Clusters induits
l4
d
l2
c
l1
h
e
g
f
b
a
l5
l3
χ_{l4} l4
χ_{d} d l4
χ_{l2} l2 d l4
χ_{c} c d l2 l4
χ_{l1} l1 c l4
χ_{h} h l1 l4
χ_{e} e d l2 l4
χ_{g} g e l4
χ_{f} f g l4 e
χ_{b} b l1 h
χ_{a} a b l1 h
χ_{l5} l5 a b
χ_{l3} l3 e

5.3.2 D'un ordre d'élimination à l'arbre joint

Les travaux de Jensen [JJ94] ont montré qu'il était possible de relier les clusters induits par l'ordre d'élimination afin qu'ils puissent former un arbre joint. Ils ont montré la propriété suivante :

Propriété 5.3 (Décomposition arborescente et ordre d'élimination optimaux) *La taille de la décomposition arborescente optimale de G correspond à la taille du cluster induit par un ordre d'élimination optimal.*

Par ailleurs, dans le but de résoudre efficacement un réseau de contraintes, l'algorithme TC [DP89] de Dechter et Pearl, et plus récemment l'algorithme JTC [Dec06] (Join Tree Clustering) de Dechter, se basent sur les clusters construits lors d'un processus d'élimination de variable (ou encore de triangulation) pour décomposer le schéma d'un réseau de contraintes vers un schéma acyclique. Nous illustrons l'arbre joint construit par JTC à la Figure 5.13. Dans un premier temps, seuls les clusters maximaux sont conservés. Par exemple, le cluster χ_b induit lors de l'élimination de b n'est pas maximal. Il se compose de l'ensemble de variables $\{g, l_4, e\}$. L'ensemble des variables de χ_g est couvert par l'ensemble des variables du cluster χ_f, le cluster χ_g ne figurera donc pas dans l'arbre joint résultat. Ensuite, JTC parcourt les clusters maximaux restants, dans l'ordre de leur création (c-à-d l'ordre d'élimination des variables auxquelles ils correspondent). Lors de la visite d'un cluster χ_v, JTC connecte ce dernier à un autre cluster non encore visité avec qui χ_v partage le plus grand nombre de variables. Par exemple, χ_b, induit par l'ordre l'élimination de b, ne partage aucune variable avec le cluster χ_f construit juste après lui. χ_b ne sera donc pas connecté à χ_f mais plutôt à χ_h avec qui il partage $\{l_1, h\}$. Durant son parcours, et par le jeu des connections des clusters, JTC construit un arbre joint des feuilles vers la racine.

L'algorithme de BE (Bucket Élimination) utilise le même mécanisme que [Dec99] pour construire un arbre joint à partir d'un ordre total sur les variables.

La bucket élimination est l'une des méthodes qui se base sur l'ordre d'élimination pour construire une décomposition arborescente.

Pour ces deux algorithmes, la running intersection est assurée à la fois à l'étape de construction des clusters lors de l'élimination de variables mais aussi à l'étape de connexions où ne sont éliminées que les variables locales au sous-arbre joint en construction. L'efficacité de l'algorithme JTC ou encore BE pour le domaine de la programmation dynamique s'explique en grande partie grâce à l'ordre d'élimination des variables en entrée à partir desquelles sont construites les clusters de l'arbre joint résultat. Dechter montre la propriété suivante.

Propriété 5.4 (Décomposition arborescente induit par un ordre d'élimination optimal) *Si l'ordre des variables en entrée de la bucket élimination représente un ordre d'élimination optimal, alors la bucket élimination retourne une décomposition arborescente optimale [Dec03].*

Nos expérimentations sur les réseaux petit monde confirment la suprématie de BE par rapport aux algorithmes de décomposition comme DFS envisagé dans un contexte distribué. Pour guider l'ordre d'élimination, nous avons utilisé les heuristiques Min-Fill ou MCS. Malheureusement, ni l'algorithme JTC, ni la Bucket élimination ne peuvent s'appliquer à notre contexte où les liens d'accointances du réseau initial et la confidentialité des variables locales doivent être respectés. En effet, dans la méthode que nous venons de présenter, plusieurs mécanismes violent les contraintes de notre cadre distribué.

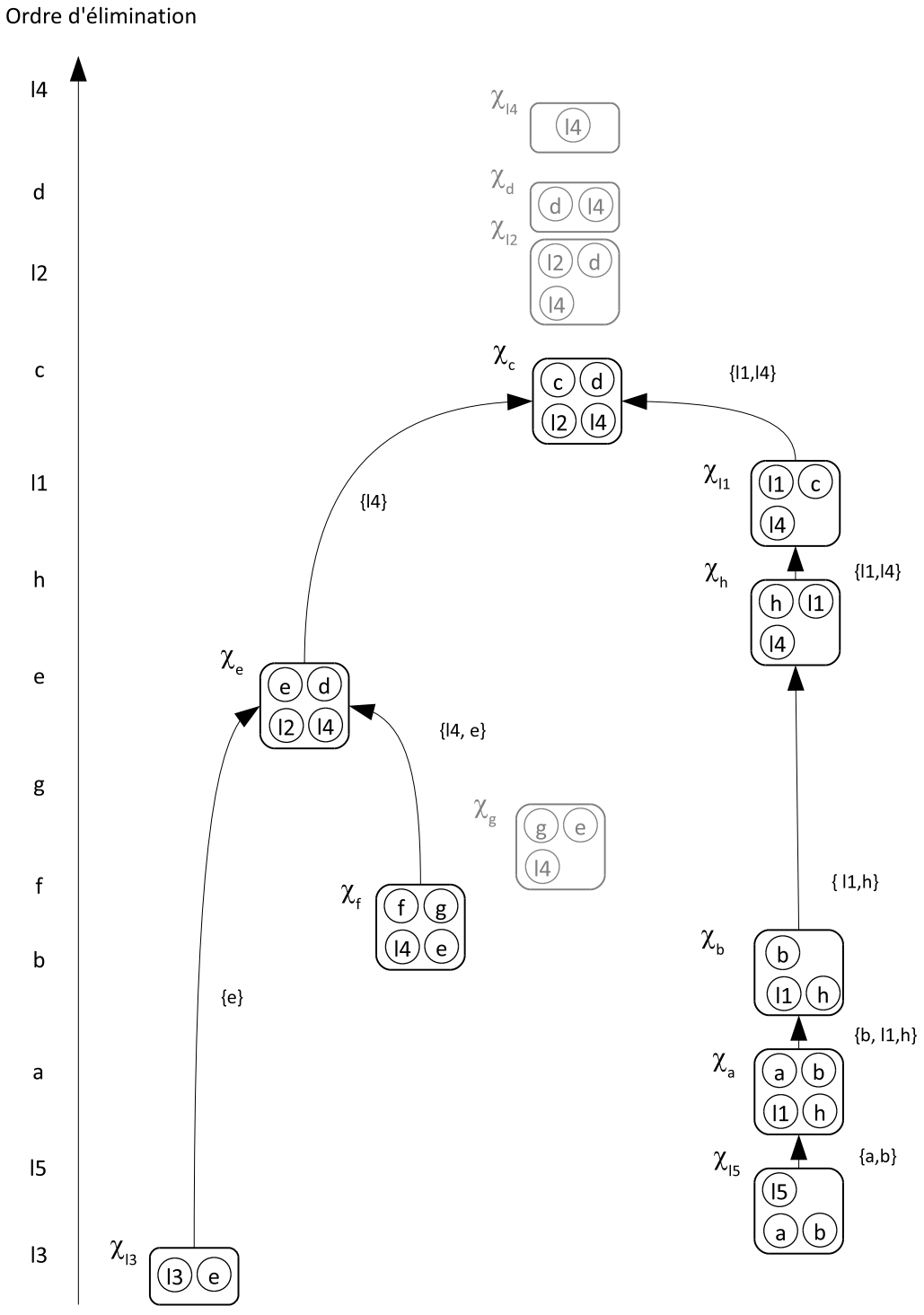


FIGURE 5.13 – Décomposition arborescente induite par JTC et les clusters Figure 5.12

Toutefois, dans la solution que nous présentons, nous nous inspirons des mécanismes d'élimination des variables et de la construction d'un arbre joint distribué des feuilles vers la racine tout en respectant la confidentialité du vocabulaire des pairs.

5.4 Pourquoi des éliminations concurrentes nous éloignent de l'optimal ?

On pourrait supposer que le contexte distribué aurait pu aider à l'exécution parallèle du processus d'ordre d'élimination, mais ce pas toujours le cas. Considérons le chemin suivant où chaque pair contient exactement une variable.

$$a - b - c - d - e - f - g - i - j$$

Considérons maintenant une exécution où les variables a et j sont éliminées de façon concurrente. Cette élimination entrainerait la création des clusters $\{a, b\}$ lors de l'élimination de la variable a et la création du cluster $\{i, j\}$ lors de l'élimination de la variable j. La taille des clusters induits par l'élimination de a ou de j est 2. De façon analogue, en poursuivant le processus d'élimination et en prenant soin d'éliminer de façon concurrente les extrémités du chemin restant, la taille des clusters induits par ce processus d'élimination ne dépassera pas 2. Parmi tous les ordres d'élimination possibles (concurrents ou non), 2 est la plus petite largeur des décompositions arborescentes (c'est la largeur arborescente du chemin).

Maintenant considérons l'exécution où trois variables a, e, et j sont éliminées de façon concurrente. L'élimination de a (resp. j) entrainerait la création du cluster $\{a, b\}$ (resp. $\{i, j\}$) de taille 2, alors que l'élimination de e entrainerait la création du cluster $\{d, e, f\}$ de taille 3. Malheureusement, nous pouvons vérifier que toute élimination simultanée d'au moins trois variables sur cette exemple conduira à la création de clusters de taille 3, ce qui nous éloigne de la largeur de décomposition optimale (qui est 2 dans ce cas). Cet exemple peut facilement se généraliser aux graphes ayant une structure d'arbre ou de graphe acyclique. Par cet exemple, nous avons montré que juste quelques décisions concurrentes peuvent nous éloigner de l'optimal. Dans notre algorithme (expliqué à la section suivante), dans le but d'éviter la prise de décisions concurrentes, seul le pair détenant l'unique jeton circulant à travers le réseau sera autorisé à s'éliminer après des élections locales. Le protocole que nous présentons dans la section suivante représente une vraie nouveauté dans l'état de l'art.

5.5 Élimination distribuée guidée par des élections locales et le passage jeton (TE)

Notre protocole distribué TE (*Token Elimination*) tire parti des méthodes de décomposition arborescente basées sur un ordre d'élimination tout en respectant la confidentialité des variables locales et les accointances du système initial. Nous construisons une décomposition arborescente distribuée des feuilles vers la racine par l'idée suivante. Tout d'abord nous nous prémunissons des éliminations concurrentes par l'utilisation d'un jeton. Un pair ne pourra s'éliminer que s'il détient le jeton. Le jeton initialement détenu par un pair quelconque, circule à travers le réseau en suivant le vote local de ces derniers. Par l'intermédiaire de messages, chaque pair vote pour un de ses voisins (y compris lui même) qu'il souhaite voir s'éliminer en premier. Un pair qui détient le jeton et qui a été choisi par tout son voisinage s'élimine et crée un nouveau cluster. Les premiers clusters créés par les pairs éliminés seront aux feuilles de l'arbre joint final. Les autres clusters représenteront des racines de sous-arbres joints déjà créés. Un cluster nouvellement créé est orphelin (il ne connaît pas encore son parent). Dans le but de le connecter à un autre cluster et de respecter la running intersection, nous ajoutons les variables partagées entre les clusters de son sous-arbre joint dans le jeton. Les variables partagées sont calculées par le mécanisme de

la Propagation Xor sur la structure (variable, arête) *sh* vue section 5.2.2. Après s'être éliminé, un pair passe le jeton à un voisin. Une élimination d'un pair n'est pas synonyme de passivité définitive. Un pair continuera à participer aux élections locales jusqu'à ce que tous ses voisins soient éliminés, après quoi il ne se chargera que de rediriger les messages. L'élimination successive des pairs induit par les élections locales et le passage du jeton dessine un arbre joint distribué des feuilles (les pairs s'étant éliminé en premier) vers la racine (le dernier pair à s'être éliminé). D'autre part, parce que le graphe dirigé induit par le choix des meilleurs voisins n'est pas toujours fortement connexe, nous avons recours à une stratégie de secours. Nous supposons qu'il existe un cycle qui traverse tous les pairs du réseau. Dans notre approche, ce cycle est construit lors d'un parcours DFS des pairs. Le protocole doit aussi reconnecter les différents clusters entre eux. Parce que nous devrons suivre les liens d'accointances du schéma initial, chaque pair éliminé recherche ses fils et questionne le réseau. Lorsqu'un fils se reconnaît, un ensemble de clusters de connexions sont créés sur le chemin le menant à son père. Les clusters de "connexions" ont toutes les variables requises pour assurer la running intersection.

5.5.1 Déroulement de l'algorithme TE

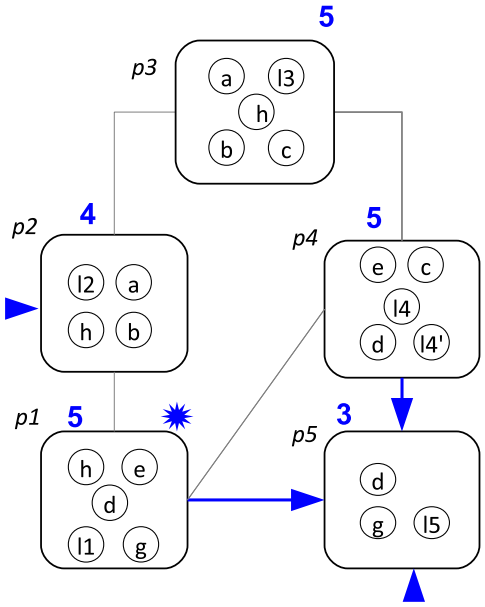


FIGURE 5.14 – TE - Min Cluster, $p1$ organise une élection locale et passe le jeton à $p5$

Avant de mieux comprendre notre algorithme, nous illustrons les principes de la Token Elimination à l'aide de l'exemple introduit au début de ce chapitre. Supposons que le jeton soit initialement dans $p1$ (cf. Figure 5.14). Ce dernier provoque des élections locales. Afin de guider le vote de chaque pair et indirectement l'ordonnancement des éliminations, chaque pair estime, par l'heuristique Min-Cluster, la taille du cluster que le pair produirait s'il était le prochain à s'éliminer. Par exemple, si $p4$ s'éliminait tout de suite, il produirait un cluster de taille 5. Le pair dans le voisinage du jeton ayant le score minimum reçoit le jeton. Le jeton est donc passé à $p5$

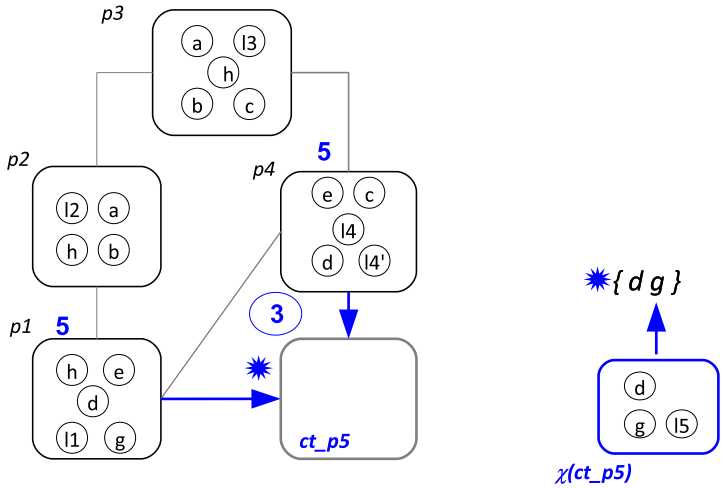


FIGURE 5.15 – TE - Min Cluster, p_5 organise une élection locale et s'élimine ct_{p5}

À la réception du jeton, $p5$ réorganise une élection locale (cf. Figure 5.15). L'information concernant le calcul du score de chaque pair est identique. $p5$ a le score minimum, tous ses voisins le choisissent comme prochain pair à éliminer : il s'élimine. Lors de son élimination, $p5$ crée un cluster ct_{p5} construit à partir de son vocabulaire V_{p5}. ct_{p5} est un cluster qui sera une feuille de l'arbre joint distribué final. Pour le moment, ct_{p5} est un cluster orphelin (il n'a pas encore de cluster parent). Afin qu'il puisse être adopté ultérieurement, $p5$ projette dans le jeton l'ensemble de ses variables partagées $\{d, g\}$ avec le reste du réseau. Dans notre algorithme, ces variables partagées sont calculées à partir de l'ensemble (variable, arête) : $sh(p5)$ vu à la section 5.2.2. Nous soulignons que les variables partagées placées dans le jeton représentent la projection de sous arbres par lesquelles les prochains pairs éliminés pourront se brancher.

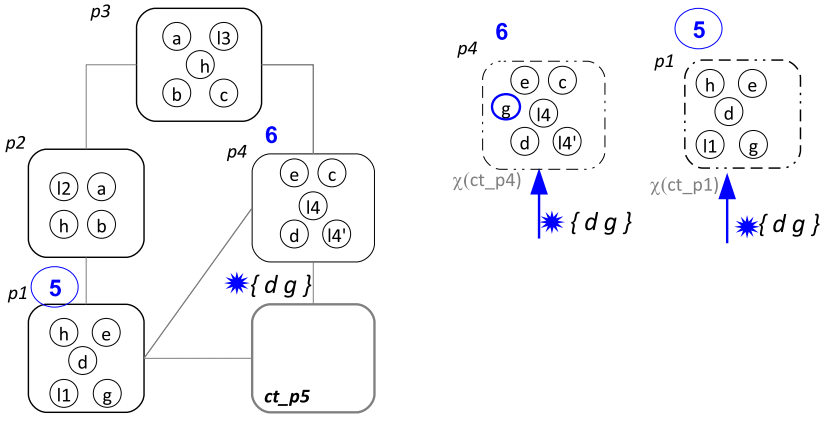


FIGURE 5.16 – TE - Min Cluster, $p3$ organise une élection locale et passe le jeton à $p1$

Après son élimination, $p5$ réorganise des élections locales (cf. Figure 5.16). Le but est de choisir le prochain voisin parmi $p1$ et $p4$ à qui passer le jeton. Lors de cette élection, et compte tenu de la projection $\{d, g\}$ placée dans le jeton, $p1$ estime que si son élimination avait lieu maintenant, il produirait un cluster de taille 5. Il n'aurait aucune variable à ajouter à son

cluster actuel qui contient déjà d et g. Ce n'est pas le cas de $p4$, qui estime que si son élimination avait lieu maintenant il produirait un cluster de taille 6. Afin de respecter la running intersection et parce que $p1$ et $p4$ contiennent la variable d, le cluster que chacun d'entre eux produirait se placerait comme parent du cluster orphelin ct_p5. Nous remarquerons ici que seule la projection des variables des clusters orphelins est nécessaire pour respecter la running intersection. $p1$ a un meilleur score que $p4$, le jeton est passé à $p1$.

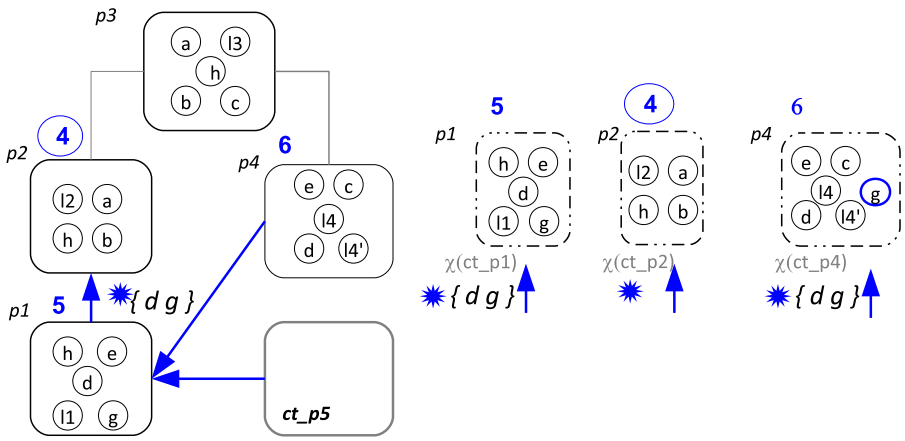


FIGURE 5.17 – TE - Min Cluster, $p1$ organise une élection locale et passe le jeton à $p2$

À la réception du jeton, $p1$ organise une élection locale (cf. Figure 5.17). C'est la deuxième élection organisée par $p1$. Cette fois ci $p1$ (resp $p2$, $p4$) estime que s'il devait être le prochain à s'éliminer, alors il produirait un cluster de taille 5 (resp 4, 6). Le faible score de $p2$ s'explique par le fait que contrairement à $p1$ ou $p2$, il ne partage aucune variable avec la projection contenue dans le jeton. Le jeton est passé à $p2$.

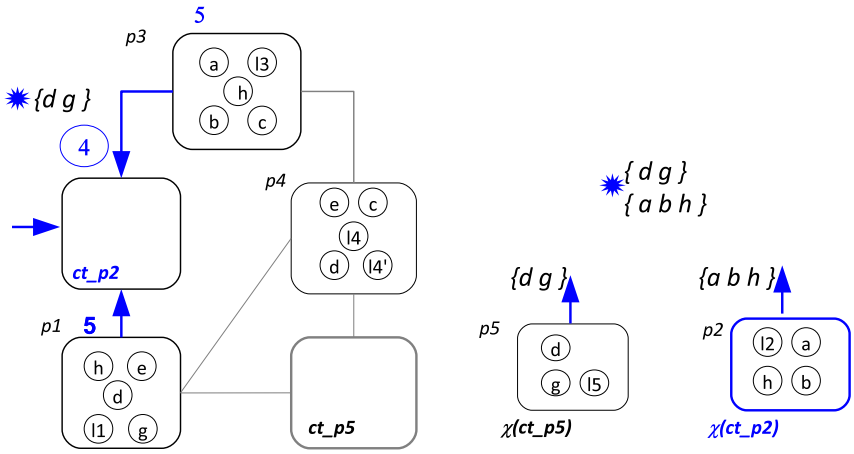


FIGURE 5.18 – TE - Min Cluster, $p2$ organise une élection locale, s'élimine et passe le jeton à $p3$

À son tour, à la réception du jeton, $p2$ organise une élection locale (cf Figure 5.18). $p2$ a le score local minimum, tous ses voisins le choisissent comme prochain pair à éliminer : il s'élimine donc. Lors de son élimination $p2$ crée un cluster ct_{p2} construit à partir de son vocabulaire V_{p2}. $p2$ ne partage aucune variable avec les projections contenues dans le jeton. Il crée le cluster ct_{p2} qui

sera aussi en feuille de l'arbre joint distribué final. ct_{p2} est un cluster orphelin, afin qu'il puisse être adopté, $p2$ place dans le jeton la projection des variables $\{a, b, h\}$ partagées entre le cluster ct_{p2} et le reste du réseaux. À ce stade le jeton contient deux ensembles de variables partagées correspondant aux projections de variables partagées déposées par des clusters orphelins. À la suite de nouvelles élections locales $p2$ passe le jeton à $p3$.

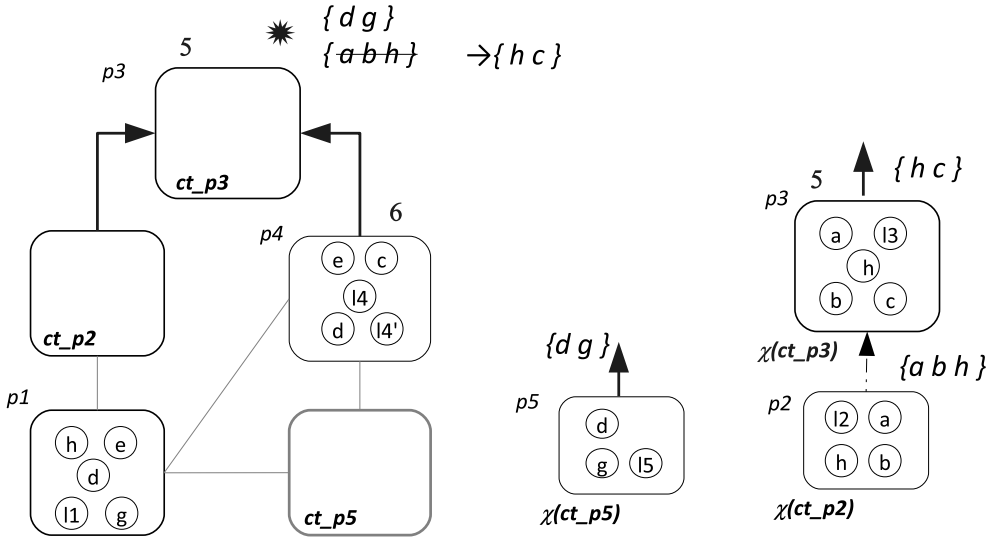


FIGURE 5.19 – TE - Min Cluster, $p3$ s'élimine et adopte un orphelin

Lorsque p_3 reçoit le jeton, il est le minimum local de son voisinage (cf Figure 5.19). Il s'élimine et crée le cluster ct_{p3} à partir de son vocabulaire et de la projection $\{a, b, h\}$ contenue dans le jeton. Le cluster ct_{p3} et la projection $\{a, b, h\}$ du pair p_2 s'intersectent. Il supprime $\{a, b, h\}$ du jeton et ajoute sa projection de variables partagées $\{c, h\}$. Afin de respecter la *running intersection*, ct_{p3} initie un processus de connexions visant à le relier à son fils ct_p2. ct_{p3} devient lui-même un cluster orphelin, à la tête d'un sous-arbre joint dont les clusters sont en cours de connexion.

Par le jeu des élections locales, des éliminations et du passage de jeton, un arbre distribué est construit implicitement des feuilles vers la racine. Avant la phase de connexions des clusters parents aux clusters orphelins que nous détaillons par la suite, nous illustrons la fin du processus d'éliminations successives à la Figure 5.20. À droite nous modélisons l'arbre joint induit par l'élimination successive des clusters et la relation parent. On peut remarquer que la confidentialité des variables locales de l'exemple introductif est respectée. À gauche nous représentons l'arbre induits par les liens parents-enfants entre clusters.. On peut constater que le lien entre le cluster ct_{p3} et ct_{p1} ne suit pas les liens d'accointances du schéma initial. Dans la phase de reconnexion de clusters (cf section 5.5.7), nous réglons ce problème.

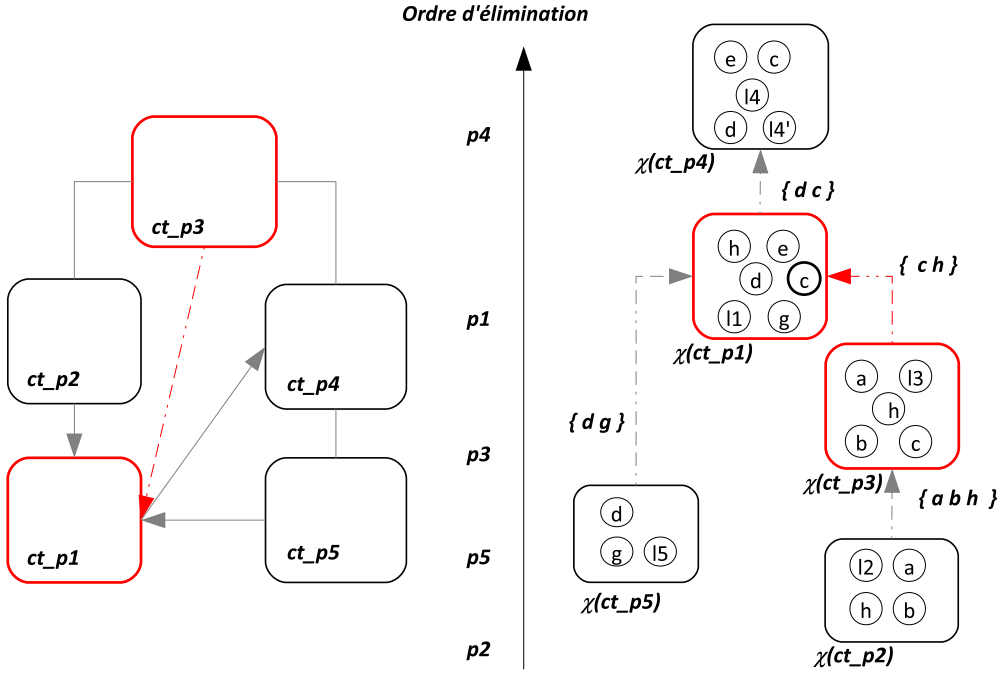


FIGURE 5.20 – TE - Min Cluster, arbre joint distribué implicite

5.5.2 Structures de données et notations

Chaque pair dans le voisinage du déteneur du jeton calcule un *score* par l'une ou l'autre de nos heuristiques Min Cluster ou Min Projection. Selon l'heuristique Min Cluster, le score d'un pair est estimé par la taille du cluster que provoquerait l'élimination de ce dernier. Selon l'heuristique Min Projection, le score d'un pair est estimé par la taille de la projection des variables partagées que provoquerait l'élimination du pair. Par un mécanisme d'élections locales, l'algorithme choisit parmi le voisinage du détenteur du jeton le pair ayant un score minimal. Chaque pair maintient en plus un tableau *scores []* résumant le score de ses voisins. Le calcul du score prend en compte les nombre de variables partagées des sous-arbres joints orphelins. C'est grâce à ces projections qu'il est possible d'estimer les heuristiques évoquées ci dessus.

-Durant tout le protocole distribué un cluster ct_p est créé par un unique pair p. Avant la phase de reconnexion chaque pair ne contient qu'un seul cluster principal.

-*Jeton* est le jeton qui circule à travers le réseau. Il est également décoré de précieuses informations. Il contient l'ensemble des clusters orphelins *ClustOrph* qui sont à la tête de sous-arbres joints en cours de construction ainsi que leurs projections de variables partagées qui serviront à les connecter. Les variables partagées sont contenues dans les structures (variable, arête) $sh(ct_f)$.

-*voisins*$*$ est l'ensemble de tous les voisins d'un pair, y compris les pairs éliminés, et lui-même. *voisins*+ est l'ensemble des voisins de tous les voisins d'un pair, y compris lui-même, mais non compris les pairs éliminés. *meilleur* est le "meilleur" voisin d'un p par les pairs, c-à-d par les pairs ayant la plus petite valeur dans le tableau *score*[], y compris p.

-$Fils(ct)$ est l'ensemble des clusters orphelins adoptés à ce jour. Dans la phase de reconnexion, cet ensemble évoluera et ne contiendra que des clusters de pairs voisins dans le graphe d'accointances.

5.5.3 L'algorithme distribué d'élimination par passage d'un jeton

Nous présentons l'algorithme distribué d'élimination par passage d'un jeton : Token Elimination TE à la page 113. Initialement le jeton est détenu par un pair quelconque. À la réception du jeton, le traitement suivi par un pair comporte trois étape : l'élection locale, une éventuelle élimination du pair élu, le passage du jeton.

5.5.4 Les élections locales

À chaque fois qu'un pair p_{jeton} reçoit (ou détient initialement) le jeton (ligne 1), il organise des élections locales (ligne 2) et attend le vote de tous ses voisins (y compris ceux déjà éliminés). La réception de ce message est traité à la ligne 30. En appliquant l'heuristique Min Cluster, le du score permet d'estimer la taille du cluster que produirait p, le pair ayant reçu le message $ELEC$ (ligne 30) s'il était le prochain pair éliminé. L'heuristique simule ainsi la création d'un cluster, lignes $9-12$. Le score de p est renvoyé à p_{jeton} mais aussi à tous ses voisins, ligne 33, et reçu ligne 37. Ainsi, tous les voisins de p_{jeton} mettent à jour leur tableau de score et sont prêts à voter. Ils informent p_{jeton} par le msg $VOTE$ si oui ou non il représente un minimal local. Lorsque le pair p_{jeton} a reçu le vote de tous ses voisins (y compris lui même) à la ligne 4, il a maintenant trois possibilités, se retirer lui-même (cf section 5.5.5), donner le jeton à son meilleur voisin ou passer par le cycle pour chercher un autre pair (voir section 5.5.6).

On remarquera ici que le tableau de score n'a été mis à jour que pour dans le voisinage de p_{jeton} et donc les votes du voisinage de p_{jeton} sont basés sur une information partielle. Si p est un voisin de p_{jeton}, p met seulement à jour le score de ses voisins qui sont aussi connectés à p_{jeton}. Tous les autres pairs sont évalués selon une ancienne valeur de $score[]$ qui peut être basée sur des sous-arbres joints anciens. Nous devons l'accepter, il s'agit juste d'une heuristique, et essayer d'avoir une meilleur estimation seraît trop coûteux. Cependant, nous remarquons ici que si aucun pair n'est élu, le jeton reste le même et suit le "meilleur" lien ou encore le cycle global, parcourant éventuellement tous les pairs. L'algorithme termine parce qu'une fois que le pair a mis à jour ses propres scores en tenant compte de l'information du jeton, si le jeton ne change pas (aucun pair n'est retiré), alors son score reste le même. Si le jeton passe par tous les pairs, toutes les valeurs du tableau $score[]$ seront mises à jour en fonction des mêmes sous-arbres joints orphelins. Cette remarque est importante pour la terminaison : nous avons en effet la garantie de trouver un noeud à supprimer après au plus un cycle et dans aucun cas après avoir vu au plus deux fois chaque arête.

5.5.5 Élimination du pair élu

Lorsque p détient le jeton et est un minimum local, *c-à-d* tous ses voisins, y compris lui-même, l'ont choisi, alors il a deux étapes à accomplir lors de son élimination. Premièrement, il crée un cluster ct_p de la ligne 8 à la ligne 16 qui sera à la tête d'un sous-arbre joint orphelin. Ce cluster contient les variables partagées entre le sous-arbre joint dont ct_p va prendre la tête et le reste du réseau. Ces variables sont calculées à partir des variables partagées de $sh(ct_p)$ (ligne 9), mais aussi des variables partagées des clusters orphelins $sh(ct_f)$ que ct_p a adopté comme fils (ligne 10) et desquelles on a retiré par la propagation XOR, les variables partagées devenues locales (ligne 11). La seconde étape consiste en la mise à jour des clusters orphelins du jetons. (ligne 14) p retire du jeton tout cluster orphelin qu'il a adopté comme fils (ligne 10) et ajoute son nouveau cluster ct_p muni de ses variables partagées $\chi(ct_p)$. Ensuite (ligne 18 - 19), il cherche à se connecter à ses fils. Enfin, p notifie ses voisins de son élimination en envoyant et accusant la réception du message $ELIM$. Durant cette étape, il est aussi important de noter que tous

les voisins de p y compris p lui-même, mettent à jour leur meilleur score. Nous reportons les explications des lignes concernant la reconnexion des sous-arbres joints à la fin de la section.

Algorithme 5: Élimination distribuée par passage d'un jeton et élections locales
Entrée: un schéma d'accointances distribué $G((P,V),ACQ)$
Sortie: un arbre joint distribué $T((CT,\chi),E)$
(1) p **reçoit** $JETON(ClustOrph)$ **de** p'
(2) // *élection locale*
(3) envoyer $ELEC(ClustOrph)$ aux $voisins^*$
(4) attendre $VOTE(b_{p'})$ des $voisins^*$
(5)
(6) **if** (p a reçu $VOTE(\top)$ de $p' \in voisins^*$)
(7) // *Éliminer* p (voir section 5.5.5)
(8) // Créer cluster ct_p
(9) $sh(ct_p) \leftarrow \{(v,\{p,p'\})$ t.q. $v \in V(p) \cap V(p')), \{p,p'\} \in ACQ\}$
(10) $Fils(ct_p) \leftarrow \bigcup_{(ct_f, sh(ct_f)) \in ClustOrph} f$ t.q. $Vars(sh(ct_p)) \cap Vars(sh(ct_f)) \neq \emptyset$
(11) $sh(ct_p) \leftarrow sh(ct_p) \bigoplus_{f \in Fils} sh(ct_f)$
(12) $\chi(ct_p) \leftarrow V_p \bigcup_{f \in Fils} Vars(sh(ct_f))$
(13) // *Mise à jour des sous-arbres orphelins*
(14) $ClustOrph \leftarrow ClustOrph(\cup ct_p, sh(ct_p) \setminus \bigcup_{f \in Fils} (ct_f, sh(ct_f)$
(15) envoyer $ELIM$ aux $voisins^*$
(16) attendre $AckELIM$ des $voisins^*$
(17) // *connexion des fils au parent (voir section 5.5.7)*
(18) **foreach** $ct_f \in Fils$
(19) envoyer $ConnecteFILS(ct_f)$ aux $voisins$
(20)
(21) // *passage du jeton* (voir section 5.5.6)
(22) **if** ($meilleur \neq p$)
(23) **if** ($voisins^+ \neq \emptyset$)
(24) envoyer $JETON(ClustOrph)$ au $meilleur$
(25) **else**
(26) envoyer $JETON(ClustOrph)$ au $succ_{DFS}(p')$
(27) **else**
(28) envoyer $JETON(ClustOrph)$ au p" t.q. p a reçu $VOTE(\bot)$ de p"

(30)p **reçoit** $ELEC(ClustOrph)$ **de** p'
(31) $score[p] \leftarrow eval(ClustOrph)$
(32) envoyer $SCORE(score[p])$ à tous les $voisins$
(33) attendre $AckSCORE$ de tous les $voisins$
(34) $meilleur \leftarrow min(score[])$
(35) envoyer $VOTE(meilleur == p')$ à p'

(37)p **reçoit** $SCORE(i)$ **de** p'
(38) $score[p'] \leftarrow i$
(39) envoyer $AckSCORE$ à p'

(41)p **reçoit** $ELIM$ **de** p'
(42) remove p' de $score$ and $voisins^+$
(43) $meilleur \leftarrow min(score[])$
(44) envoyer $AckELIM$ à p'

5.5.6 Passer le jeton

L'idée de la ligne 23 à la ligne 28 est de laisser le jeton suivre les meilleurs liens dans le graphe et se base sur un ordre statique de DFS pour s'échapper. Si p n'est pas le meilleur choix (c-à-d ce n'est pas un minimum local), alors si p a encore des voisins, il envoie le jeton au meilleur de ses voisins non éliminés. Sinon on se repose sur le circuit induit par un parcours DFS pour s'échapper. Si p est son propre meilleur choix, nous sommes dans un minimum local, et nous avons aussi besoin d'en sortir. Sinon au moins un des voisins en vie ne considère pas p comme son meilleur choix, le jeton est passé à l'un de ces voisins ligne 28.

5.5.7 Étape finale : reconnexions des clusters orphelins

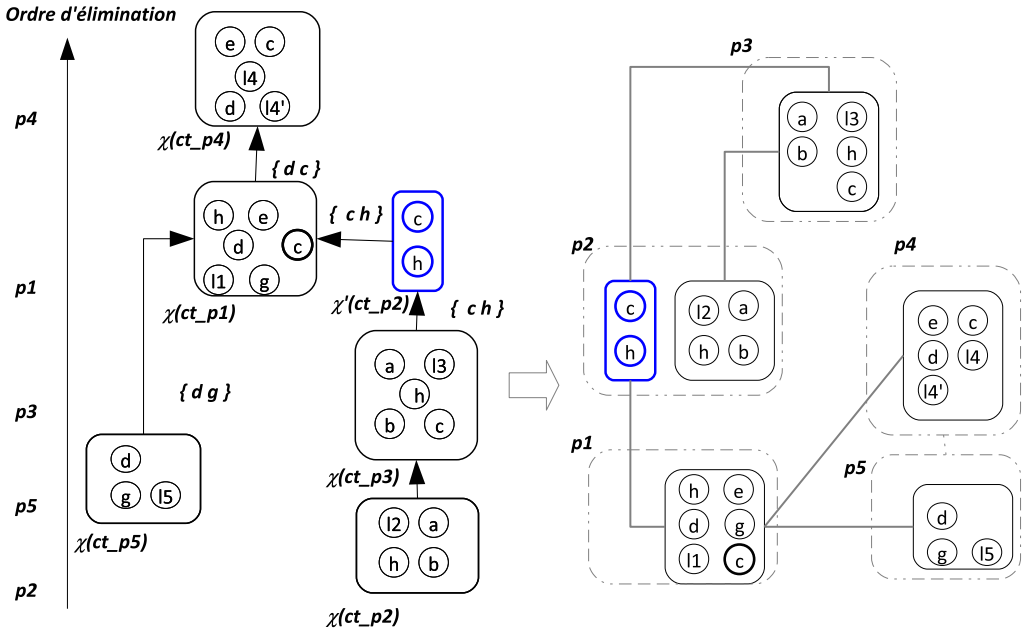


FIGURE 5.21 – TE - Min Cluster, $p3$ s'élimine, adopte un orphelin

Lorsque l'algorithme 113 arrive à la ligne 19, un pair cherche à se connecter au cluster orphelin qu'il a adopté (ligne 10). Comme nous l'avons vu au schéma 5.20, pour le cluster ct_{p1}, il se peut que le cluster orphelin adopté (par exemple ct_{p3}) ne se soit pas voisin direct. Dans ce cas il est nécessaire de connecter les parents à leur fils en suivant les liens d'accointances du réseau. Un pair cherche à se connecter à son fils f en inondant le réseau du message $ConnecteFils(f)$ (ligne 3 - 9). Afin de pouvoir remonter au pair parent, un pair recevant le message $ConnecteFils(f)$ pour la première fois enregistre l'expéditeur comme $parent(f)$ (ligne 4 - 5). De même, afin d'arrêter l'inondation, un pair diffuse le message $ConnecteFils(f)$ à ses voisins uniquement si c'est la première fois qu'il le reçoit (ligne 4 et 9). Un fils qui se reconnait (ligne 6), initie la remontée du chemin vers le parent (ligne 11 -21) en créant à chaque pair intermédiaire un cluster contenant les variables partagées du cluster fils. Lorsque la remontée arrive au parent, celui-ci met à jour ses fils en y remplaçant le fils recherché par le cluster d'un pair voisin contenant toutes ses variables partagées. À la fin de l'algorithme, nous avons les propriétés suivantes : chaque pair contient exactement un cluster principal créé lors de l'élimination du pair et de 0 à n cluster de

Algorithme 6: Connection des clusters parents à leurs fils

(2) p **reçoit** $ConnectFILS(ct_f)$ **de** p'
(3) *// recherche par innondation*
(4) **if** $(parent(ct_f) \neq null)$
(5) $parent(ct_f) \leftarrow p'$
(6) **if** p a créé ct_f
(7) envoyer $CreerCHEMIN(ct_f, ct_f, Vars(sh(ct_f)))$ à p'
(8) **else**
(9) envoyer $ConnecteFILS(ct_f)$ aux $voisins - p'$
(10)
(11) p **reçoit** $CreerCHEMIN(ct'_f, ct_f, \chi_{ct_f})$ **de** p'
(12) **if** $f \notin Fils(p)$
(13) *// sur le chemin d'un cluster fils à son parent*
(14) Créer cluster $ct"_f$
(15) $Fils(ct"_f) \leftarrow ct'_f$
(16) envoyer $CreerCHEMIN(ct"_f, ct_f, \chi_{ct_f})$ à parent $parent(ct_f)$
(17) envoyer $PARENT(ct"_f, ct'_f)$ à p'
(18) **else**
(19) *// arrivée au parent*
(20) $Fils(ct_p) \leftarrow Fils(ct_p) \cup ct'_f \setminus ct_f$
(21) envoyer $PARENT(ct"_f, ct'_f)$ à p'
(22)
(23) p **reçoit** $PARENT(ct"_f, ct'_f)$ **de** p'
(24) $parent(ct'_f) \leftarrow ct"_f$ *//pour une utilisation future use*

"connexion". Nous illustrons à la Figure 5.21 la création du cluster $ct'2$ reliant le cluster ct_{p3} au cluster ct_{p1} par le pair $p2$.

5.5.8 Analyse de complexité

À la fin de l'algorithme, n pairs ont été éliminés. Pour chaque élimination, le jeton a potentiellement visité l'ensemble de toutes les arêtes ACQ dans les deux sens (dans le pire des cas) faisant une élection locale à chaque pair p. Après quoi, durant la phase de reconnexion des sous arbres, chaque pair sera impliqué. Puisque chaque élection locale requiert un nombre constant de messages non concurrents, la complexité global est approximée par $O(n * ACQ)$ c.à.d $O(n^3)$.

Nous avons implémenté et analysé les performances de notre algorithme sur une plate forme centralisé simulant le contexte distribué.

5.6 Évaluation de la largeur arborescente sur les graphes petit monde

Dans la section précédente, nous avons présenté l'algorithme TE qui transpose le principe des techniques basées sur l'ordre d'élimination dans un contexte distribué. Il élimine un à un les pairs élus par leur voisinage et en possession du jeton. Un pair est élu s'il a le meilleur score de son voisinage. Le score de chaque pair est calculé par l'heuristique Min Cluster (la taille du cluster que produirait un pair lors de son élimination) ou Min Proj (la taille de la projection des variables partagées que produirait un pair lors de son élimination).

Nous comparons les performances et la qualité des décompositions arborescentes produites par notre algorithme TE par rapport à la bucket élimination (BE) guidée par des heuristiques d'ordonnancement issues du contexte centralisée (Min-Fill) présentée à la section 5.3.2 page 104.

Nous comparons aussi notre approches par rapport des décompositions arborescentes envisagées dans un contexte distribué (DFS, DFS MCS) présentée à la section 5.2.1.3 page 94.

Dans le but de pouvoir comparer la qualité des décompositions arborescentes produites par TE, par rapport à d'autres méthodes du contexte distribué et du contexte centralisé, nous avons généré des benchmarks de graphes d'interactions de modèles petit monde. Les graphes d'interactions où une variable étiquette un noeud sont nécessaires à l'application des méthodes du contexte centralisé. Afin de réutiliser cette modélisation dans un contexte distribué, nous avons apporté certaines adaptations à TE et considéré que le vocabulaire associé à un pair se constituait de sa variable ainsi que de ses variables voisines non encore éliminées. Dans ce cas une variable partagée du vocabulaire associé à un pair est propagée jusqu'à ce que son pair initial s'élimine.

5.6.0.1 Les graphes petits mondes

Dans le but de générer des problèmes simulant des applications réelles, plusieurs travaux de Newman [NG04], [New03] sur les modèles aléatoires, ont montré que le graphe d'interactions d'une grande partie des problèmes réels avait une structure dite petit monde et sans échelle. Par exemple, dans le domaine du diagnostic de circuits électroniques les expériences de Provan et Wuang [PW07] [WP09] ont montré que la topologie de circuits électroniques ISCAS suivait effectivement une topologie de graphe petit monde de type WS. L'étude de Walsh [Wal99] compare la satisfaction de problèmes de coloration de graphes et de planification par différentes techniques de restart pour des problèmes suivant cette même structure WS. Afin de tester notre approche sur spectre de réseaux petits monde plus large, nous avons aussi testé le modèle de génération de graphes proposé par Barabassi et Albert [BA99] (BA Graph) reconnu pour sa capacité à représenter un grand nombre d'applications du monde réel (le World Wide Web, des citations scientifiques, des échanges de mail, ...). Comparativement à un réseau aléatoire classique assez homogène (la plupart des noeuds ont approximativement le même nombre de voisins comme c'est le cas dans WS), les réseaux de type BA sont extrêmement hétérogènes (la majorité des noeuds ont peu de voisins, très peu de noeuds sont connectés à un grand nombre de voisins). Nous avons implémenté un générateur aléatoire pouvant générer des graphes de type BA ou WS en java.

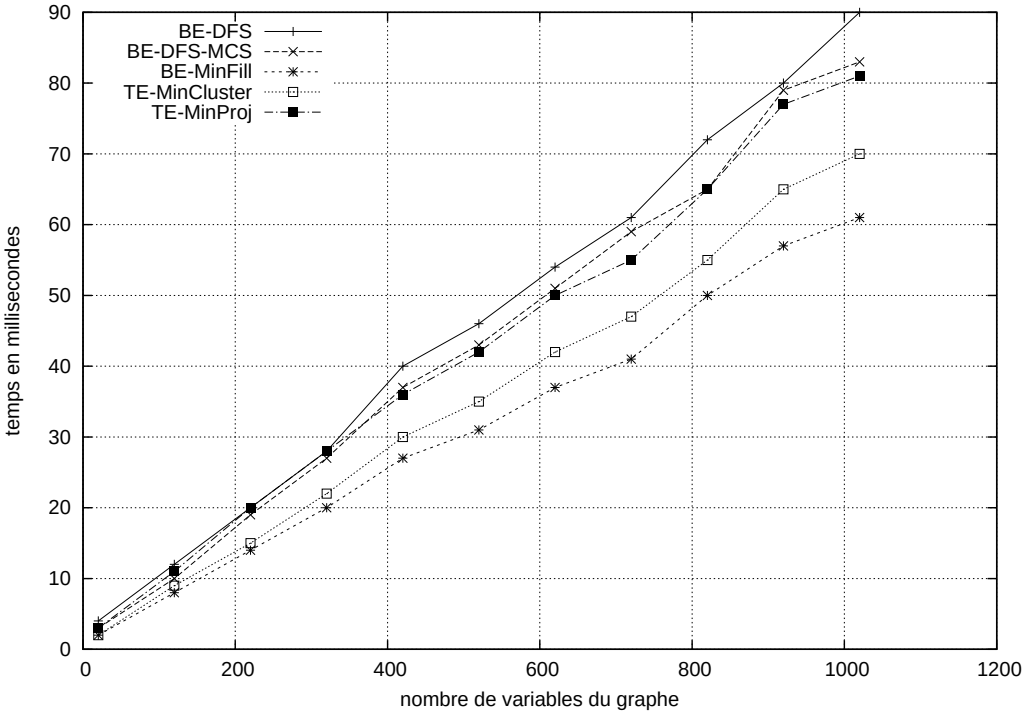


FIGURE 5.22 – Largeur arborescente de BA graphes

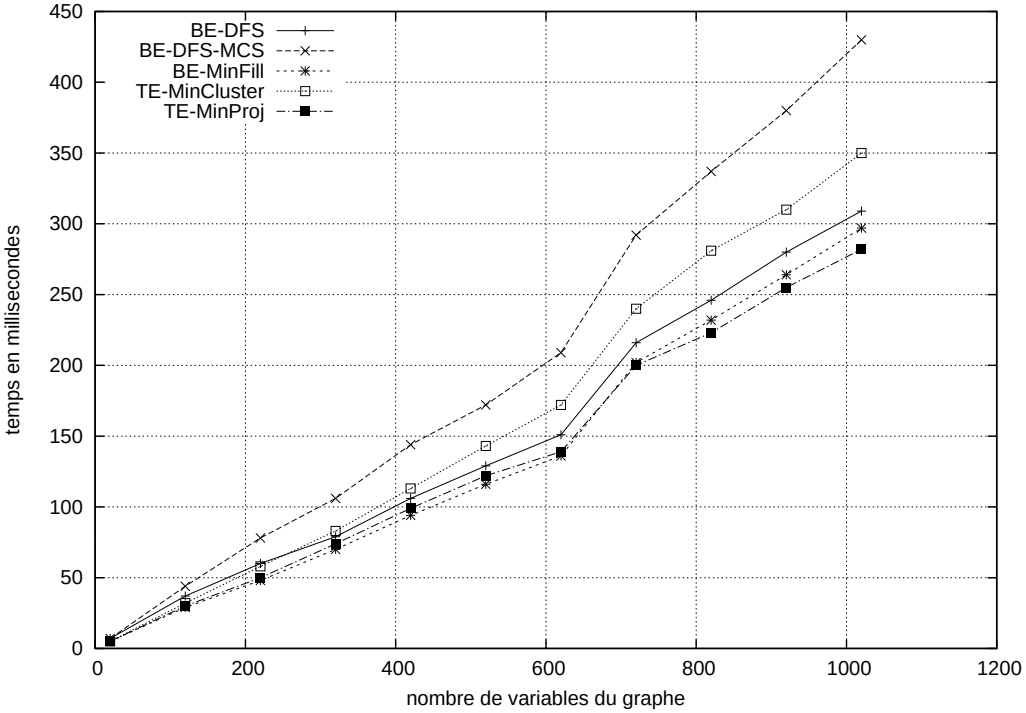


FIGURE 5.23 – Largeur arborescente de WS graphes

5.6.1 Qualité des décompositions arborescentes

Aux figures 5.22 et 5.23, nous mesurons la la moyenne des largeurs arborescentes (en ordonnée) par rapport à la taille du graphe(en abscisse) sur 10 instances.. Plus petite est la largeur arborescente, plus rapide et plus succinct sera le raisonnement. Nous rappelons qu'une faible diminution de la largeur arborescente peut traduire un gain exponentiel en temps ou en espace pour le raisonnement. Nous comparons notre approche la Token Elimination TE guidée par deux heuristiques TE-MinCluster TE-MinProj par rapport à la Bucket Elimination BE guidée par différent heuristiques efficaces utilisées en centralisées BE-MinFill et en distribué BE-DFS, BE-DFS-MCS. Nous rappelons que le Bucket élimination suppose un ordre total sur ses variables. Cette ordre total est donné par l'heuristique d'exploration. Nous rappelons la signification de chaque heuristique :

- **DFS** : choix local guidé parcours en profondeur d'abord.
- **DFS-MCS** : choix local guidé parcours en profondeur d'abord privilégiant l'exploration des noeuds non visités ayant une nombre de voisins maximal.
- **MinFill** : choix global guidé par les noeuds pour lesquelles le voisinage se rapproche d'une clique.
- **MinCluster** choix local guidé par les noeuds pour lesquelles la taille du cluster qu'induirait l'élimination est minimale.
- **MinProj** choix local guidé par les noeuds pour lesquelles la taille de la projection (c-à-d du séparateur) qu'induirait l'élimination est minimal.

Toutes les méthodes et heuristiques ont été implémentées en java.

Nos expérimentations montrent que la qualité des décompositions de TE surpassent les méthodes homologues du contexte distribué (BE-DFS, BE DFS-MCS) sur les instances WS et BA. De façon surprenante (fig 5.23), TE produit de meilleures décompositions que BE Min-Fill pour de larges instances (> 700 variables) de réseaux de types WS. Pour des réseaux de type BA (fig 5.22), nous constatons que la qualité de la décomposition produite TE est comparable sans pour autant être meilleure que celle fournie par BE Min-Fill. De même, de façon inattendue, alors que DFS-MCS surpasse DFS sur les instances de graphes irréguliers, DFS surpasse de loin DFS-MCS qui affiche les plus mauvais résultats.

5.6.2 Rapidité des décompositions arborescentes

Les figures 5.24 et 5.25reportent la moyenne des temps CPU (en ordonnée) nécessaire à chaque méthode pour décomposer un graphe dont la taille est indiquée en abscisse. La moyenne a été calculée sur 10 instances. Nous constatons que la performance CPU de la construction de l'arbre joint implicite de TE est comparable aux algorithmes DFS pour des réseaux WS (cf. Figure 5.25), TE obtient les moins bonnes performances sur les réseaux de type BA (cf. Figure 5.24). Nous notons aussi que BE Min-fill-in semble suivre une pente cubique sur les instances WS.

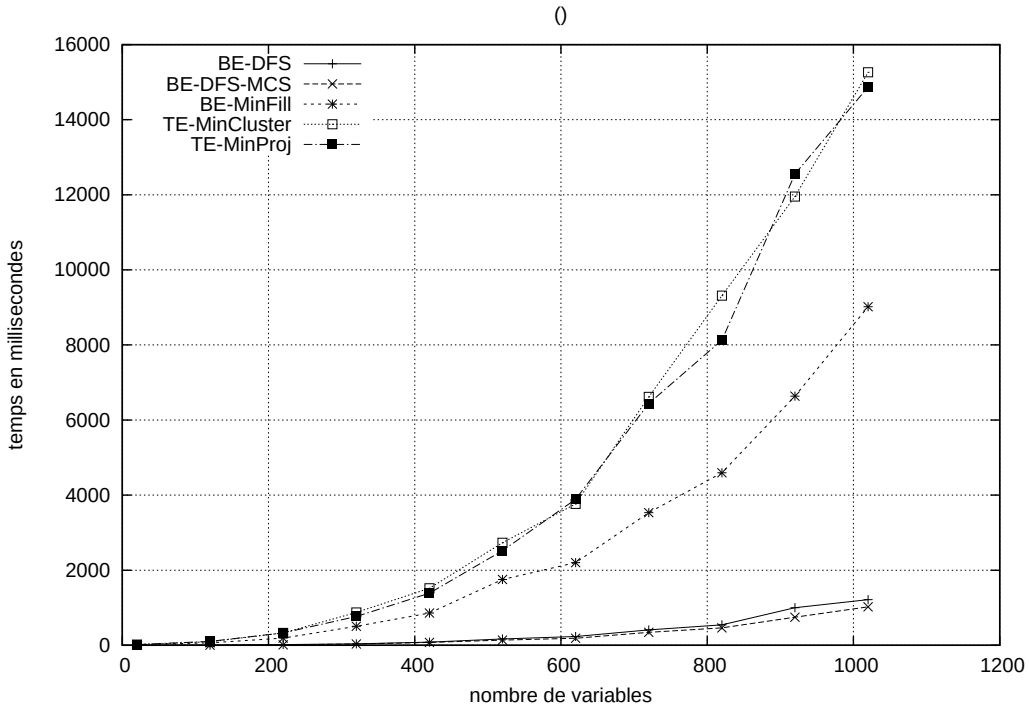


FIGURE 5.24 – Temps de décomposition de BA graphes en ms

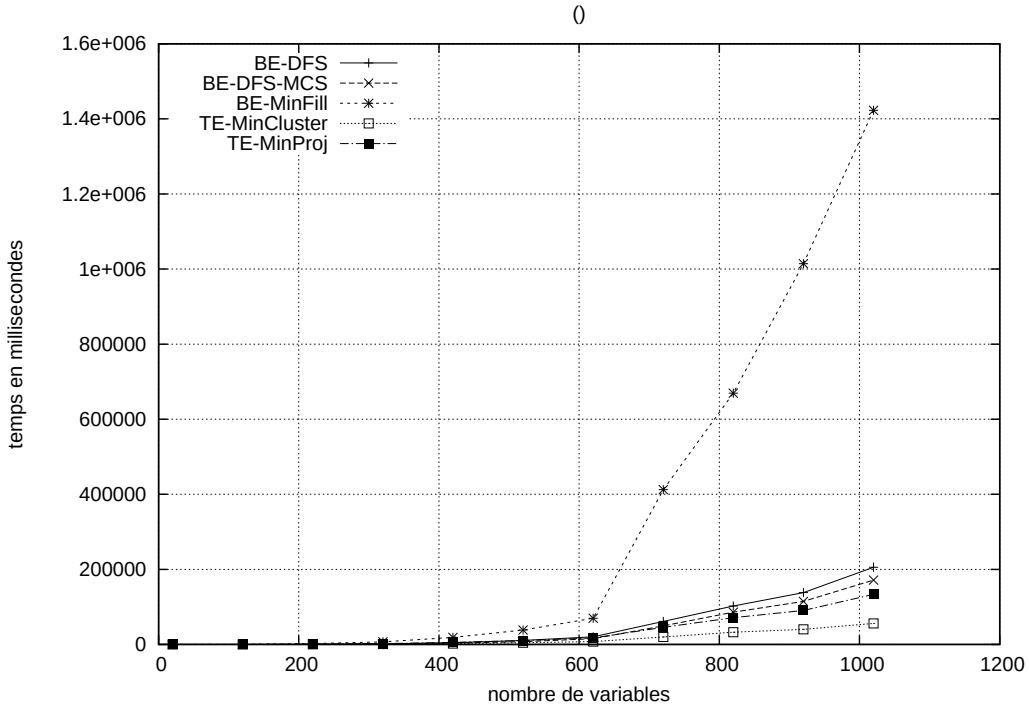


FIGURE 5.25 – Temps de décomposition de WS graphes

5.6.3 Conclusion

En résumé, notre algorithme TE se propose de construire une décomposition arborescente distribuée respectant la confidentialité des variables locales ainsi que les liens d'accointances du réseau initial. Nous avons montré que cet algorithme est capable de décomposer de larges instances de graphes de type petit monde. La qualité des décompositions arborescentes produites surpasse les méthodes de référence du contexte distribué et est comparable aux méthodes du contexte centralisé. Nous avons vu qu'à partir d'un réseau décomposé nous pourrons garantir un diagnostic distribué correct. Dans le chapitre suivant, à partir de la décomposition arborescente calculée par TE, nous transformons la description d'un système vers une description compilée pour le diagnostic distribué.

Chapitre 6

Diagnostic distribué d'un réseau structuré de FNDs

Nous rappelons que la problématique de diagnostic distribué consiste à expliquer le comportement d'un système global, par l'ensemble de ses pairs, chaque pair ayant une visibilité et un accès limité à la connaissance de la description du système global. Dans ce cadre, les pairs ne pourront qu'expliquer le comportement de descriptions auxquelles ils ont accès par des diagnostics localement cohérents. Nous rappelons que notre objectif est de compiler un système initial vers un système accessible où :

1. les diagnostics localement cohérents de chaque pair sont en mesure d'expliquer le comportement global du système (c-à-d le système de pairs devra garantir un diagnostic distribué correct).

2. tout diagnostic global d'un système devra se retrouver à travers un ensemble de diagnostics locaux (c-à-d le système de pairs devra garantir un diagnostic distribué complet).

À la fin du chapitre 3 nous avons montré qu'expliquer le comportement global d'un système initial par un système accessible équivalent (c-à-d où la sémantique du système initial globale reste inchangée mais où la sémantique des descriptions locales à chaque pair a évolué) garantit un diagnostic distribué complet. De même dans le chapitre 4 , nousavons montré qu'expliquer le comportement global d'un système initial par un système équivalent et structuré (c-à-d contenant un arbre joint couvrant tous ses pairs) garantit un diagnostic distribué correct. Dans le chapitre 5, nous avons exhibé une méthode de décomposition arborescente distribué TE qui permet de transformer tout système cherchant à préserver ses accointances et la confidentialité du vocabulaire local à chaque pair vers un système structuré. Dans ce chapitre, nous repartons de l'arbre joint distribué construit par TE à travers le réseau de pairs et redistribuons les descriptions des pairs à travers les clusters. Nous transformons alors la description locale de chacun des clusters vers une FND locale globalement cohérente. Ensuite, nous vérifions que le système obtenu est compilé pour le diagnostic distribué. Pour cela nous mettons en évidence un algorithme où chaque pair sera en mesure de vérifier que chacun de ses diagnostics locaux fait partie d'un diagnostic global minimal. Nous nous basons sur nos précédents travaux sur le diagnostic à base de cohérence [ADS08] qui montrent que toute FND d'un système représente aussi ses diagnostics minimaux.

6.1 Un réseau de FNDs est-ce raisonnable ?

La plupart des approches s'attaquant à la problématique du diagnostic à base de modèles considèrent que la description du système initial est proche d'une Forme Normale Conjonctive (c-à-d un ensemble de clauses Σ). Dans notre modélisation des comportements distribués, nous avons fait la même hypothèse. Nous considérons que la description $DS_p : \langle V_p, L_p, \Sigma_p \rangle$ de chacun des pairs p d'un système de pairs diagnostiqueurs $S : \langle P, G, K, DS \rangle$, se modélise sous la forme d'un sous-système par la formule FNC : Σ_p. Dans l'état de l'art (section 2.3.2), nous avons remarqué que certaines approches telles que le système de maintien de vérité distribué [MJ89] ou encore Somewhere [ACG+06] seraient en mesure de calculer en premier lieu l'ensemble des conflits (rappel : un conflit est une clause contenant un ensemble de littéraux de mode ne pouvant pas être corrects ensemble). C'est seulement après le calcul de tous les conflits que seront calculés les diagnostics minimaux. Cependant, cette méthode n'a que peu d'espoirs d'aboutir lors d'un diagnostic en ligne, dans la mesure où tous les conflits doivent être connus avant que le premier diagnostic ne puisse être retourné. Par exemple, en présence d'une politique de confidentialité restrictive où même les variables de modes devront rester locales à un pair, il ne sera pas possible d'exprimer tous les conflits d'un système chez un pair. Par contre si l'hypothèse de la transcription FND de la théorie globale peut être considérée comme non réaliste, des pairs de petite taille, eux, peuvent admettre une petite théorie FND qui doit être en pratique de taille raisonnable. C'est ce que propose les travaux de Feldman et al [FvG06] où est trouvé un compromis entre temps de compilation et temps de réponse à une requête en compilant dans une première étape un réseau hiérarchique de FNCs vers un réseau hiérarchique de FNDs. Comparativement aux travaux de Feldman et al [FvG06] qui s'intéressent à la recherche de diagnostics de cardinalité minimale dans un contexte centralisé, nous cherchons à représenter tous les diagnostics minimaux (pour l'inclusion ensembliste) à travers l'ensemble des pairs d'un système intrinsèquement distribué. Un réseau hiérarchique de FNDs est une représentation très proche mais plus succincte qu'un réseau structuré de contraintes extensionnelles utilisé par Dechter [FD95]. Un tel réseau est illustré Figure 3.8 à la fin du chapitre 3. De plus nous pouvons espérer qu'un réseau structuré de FNDs soit donc plus succinct qu'un OBDD par exemple. En effet, le corollaire 24 de [Bod98] montre que la largeur de la décomposition du problème initial en un arbre est toujours plus succincte que sa largeur de décomposition en un chemin. Nous noterons que les réseaux structurés de FNDs offrent une représentation au pire exponentielle en la largeur de la décomposition du problème initial en un arbre alors que les OBDDs de leurs côtés offrent une représentation au pire exponentielle en la largeur de la décomposition du problème initial en un chemin.

Exemple. 6.1 *À la Figure 6.1 nous supposons que TE a structuré notre exemple jouet en propageant la variable v dans le cluster ctE. Nous considérons que TE a associé à chaque pair p_i un cluster ct_i. Dans cet exemple, chaque formule encodant une description locale accessible à un cluster a été récrite en FND.*

6.2 Garantir un diagnostic distribué correct par un réseau structuré de FNDs globalement cohérentes

Nous rappelons qu'à l'issue de l'exécution de l'algorithme TE, un arbre joint distribué de clusters $T((CT, \chi), E)$ est construit à travers le réseau de pairs $S : \langle P, G, K, DS \rangle$. Ce arbre joint est implicitement représenté à travers les liens $parent(ct)$ et $Fils(ct)$ d'un cluster ct. Un arbre joint distribué respecte la conformité des dépendances. Pour chaque pair p il existe un cluster

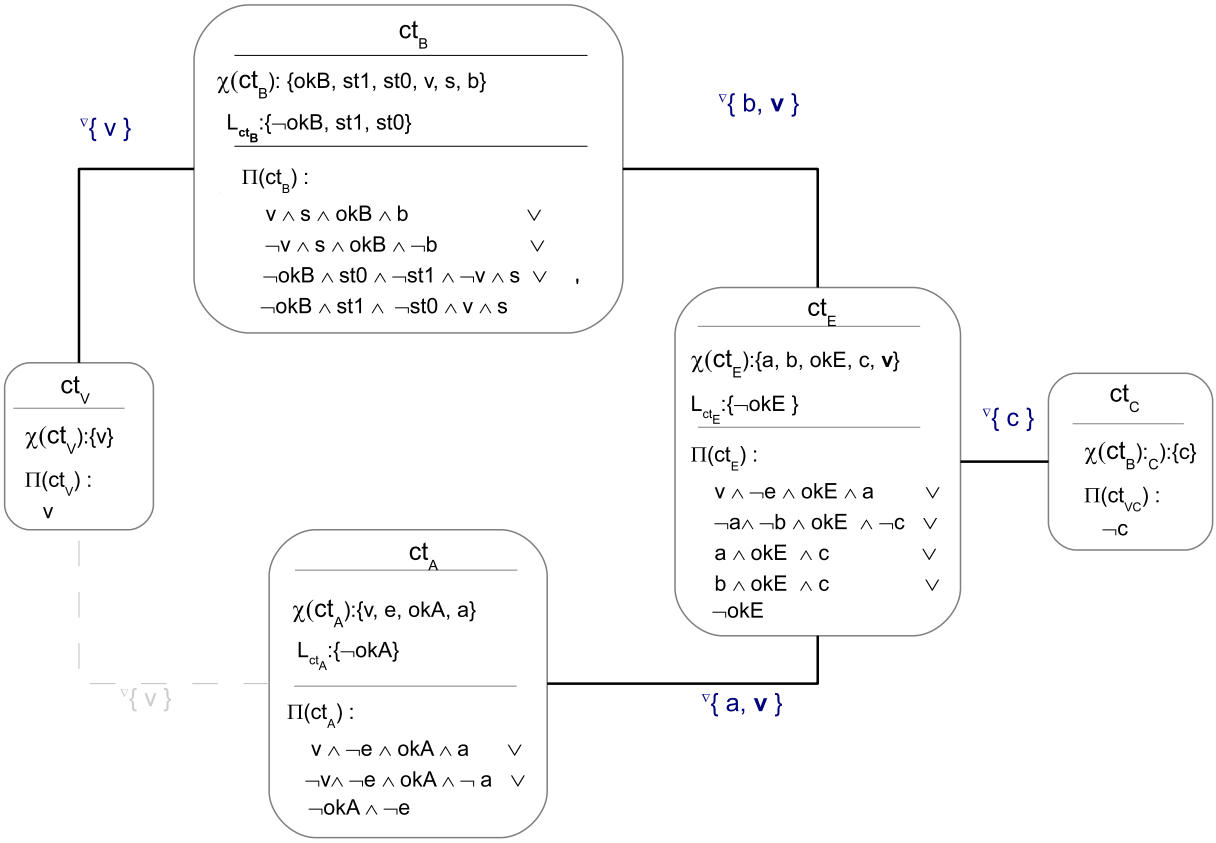


FIGURE 6.1 – Système Figure 3.4 p.54 structuré par TE et traduit en un système de FNDs

$ct \in CT$ t.q. $V_p \subseteq \chi_{ct}$. Dans l'algorithme TE, nous avons vu que le premier cluster ct créé par un pair p préserve la conformité des dépendances. Ainsi, le vocabulaire V_p utilisé par le pair p pour décrire un sous-système est donc inclus dans le vocabulaire $\chi(ct)$ du cluster ct ($V_p \subseteq \chi(ct)$). Nous déduisons que chaque formule Σ_p décrivant le comportement du pair p pourra être associée à la formule $\Sigma(ct)$ du cluster ct t.q $\Sigma(ct) = \Sigma_p$. Concernant les autres clusters construits par un pair p, si leur vocabulaire ne peut couvir V_p, nous associons à $\Sigma(ct)$ le symbole $Vrai$.

Nous remarquerons que la conjonction des formules encodées par les clusters conserve la sémantique du système de pairs initial (c-à-d $\bigwedge_{ct \in CT} \Sigma(ct) \Leftrightarrow \bigwedge_{p \in P} \Sigma_p$). En quelque sorte le système structuré $T((CT, \chi, \Sigma), E)$ décrit un système accessible équivalent à S. C'est de ce système structuré dont nous partons pour construire un réseau structuré de FNDs globalement cohérent. Nous ouvrons une parenthèse ici pour préciser que nous avons décrit un système de pairs formellement à l'aide de liens d'accointances et une politique d'accès. Dans le chapitre précédent nous avons montré que l'algorithme TE construisait un décomposition arborescente distribuée respectant à la fois les liens d'accointances et une politique de confidentialité. Ici nous utiliserons exclusivement un raisonnement structuré basé sur les clusters et les liens entre clusters. À partir de là, nous veillons scrupuleusement à ce que les échanges de messages entre les clusters définis par TE respectent le vocabulaire partagé entre ces derniers. Cette précaution nous permettra de respecter la politique de confidentialité à travers les protocoles que nous proposerons. Ainsi, nous ne parlerons plus de politique de confidentialité ou encore de respect des liens d'accointances qui sont en principe respecté par l'arbre joint où se situe le raisonnement.

Nous supposons que la description FNC $\Sigma(ct)$ de chaque cluster est traduite vers une FND

locale $\Pi(ct)$ par la primitive $cnf2dnf$.

Afin de respecter la politique de confidentialité, un cluster n'échangera avec un voisin que l'expression de ses impliquants locaux restreint sur le vocabulaire partagé avec le voisin. Le schéma du système dessiné par TE restera inchangé. La confidentialité sera donc préservée. Ainsi, nous utilisons l'opération de restriction *restrict* que nous avons appliqué aux modèles, aux monômes d'une FND.

Définition 6.1 (restriction d'une FND sur un vocabulaire, sur un ensemble de littéraux) *Soit* Π *une formule FND :*
La restriction de Π *sur un ensemble de variables* V *se définit par :*

$$restrict(\Pi, V) = \{I' | \exists I \in \Pi \text{ } t.q. \text{ } vars(I') = vars(I) \cap V\}$$

La restriction de Π *sur un ensemble de littéraux* L *cohérents se définit par :*

$$restrict(\Pi, V) = \{I \cap L | I \in \Pi\}$$

Exemple. 6.2 (restriction des FNDs sur le vocabulaire partagé) *Reprenons la description FND du cluster* ct_B *:* $\Pi(ct_B)$ *décrite Figure : 6.1. Considérons* $\Pi^1(ct_B) = \Pi(ct_B) \wedge v$ *et* $\chi(ct_B) : \{okB, st1, st0, v, s, b\}$ *le vocabulaire du cluster* ct_B*.* $\Pi^1(ct_B)$ *représente la mise à jour de* $\Pi(ct_B)$ *où a été prise en compte l'observation* v *(cf Figure : 6.3). Dans notre algorithme,* ct_B *transmet à* ct_E *l'expression de sa FND mise à jour (*$\Pi^1(ct_B)$*) restreinte sur le vocabulaire partagé* $\chi(ct_B) \cap \chi(ct_E)$

$$\Pi^1_{ct_B} = \begin{array}{l}(v \wedge s \wedge okB \wedge \neg st1 \wedge \neg st0 \wedge b \vee) \\ (v \wedge s \wedge \neg okB \wedge st1 \wedge \neg st0)\end{array} \quad restrict(\Pi^1_{ct_B}, \chi(ct_B \cap \chi(ct_E)) = \begin{array}{l}(v \wedge b \vee) \\ v\end{array}$$

Afin de préserver la confidentialité du vocabulaire local, dans l'algorithme $ArbreJoint - \Sigma 2\Pi$*, les messages échangés entre pairs contiendront uniquement l'expression de FNDs locales restreintes sur le vocabulaire partagé.*

D'autre part, une FND locale ne suffit pas, il faut la mettre à jour et la transformer en un sous-système globalement cohérent. Du fait que l'on se situe dans un système structuré, nul besoin de vérifier l'arc cohérence. Nous transformons les descriptions locales en descriptions locales globalement cohérentes par un algorithme en 2 phases. Pour ce faire, nous introduisons l'opérateur de distribution $\otimes$ entre $FNDs$ qui distribue le $\wedge$ de la conjonction de FNDs locales sur le $\vee$ des impliquants d'une FND et retourne une nouvelle FND. Plus formellement on a :

Définition 6.2 (distribution ($\otimes$) de FNDs) *La distribution de deux FNDs* Π_{ct_1} *et* Π_{ct_2} *se définit par :*

$$\Pi_{ct_1} \otimes \Pi_{ct_2} = \{I_1 \wedge I_2 | I_1 \in \Pi_1, I_2 \in \Pi_2, I_1 \wedge I_2 \not\models \bot\}$$

Nous illustrons la distribution de l'expression de la FNDs sur ses variables partagées d'un cluster *cte* issu du pair p_E et du cluster *cta* issue du pair p_A FNDs Figure 6.2.

L'opération de distribution représente la composition cohérente d'impliquants de Π_{ct_1} avec ceux de Π_{ct_2}. Si le résultat est minimisé, cet opérateur est exactement l'opérateur de *clause-distribution* de [CS01], mais appliqué aux FNDs et aux monômes.

Dans un soucis de représentation succincte mais aussi afin de sélectionner les diagnostics les plus intéressants, nous cherchons à conserver les diagnostics ou encore les impliquants minimaux par rapport à l'inclusion ensembliste. Afin d'avoir une seule définition pour l'opération de minimalité, nous considérons une FND comme une base, c-à-d un ensemble d'ensemble de littéraux.

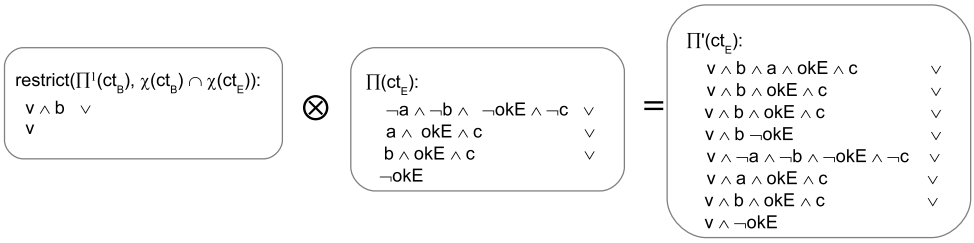


FIGURE 6.2 – Distribution de FNDs

Définition 6.3 (Impliquants minimaux d'une FND) *Soit une FND* Π *représentée par un ensemble d'ensemble de littéraux* $B = \{E_1, ..., E_n\}$. E_i *représente un impliquant de* Π . $min_{\subseteq}(B)$ *représente tous les ensembles* $E_i \in B$ *non sous-sommés dans* B.

$$min_{\subseteq}(B) = \{E_i \in B t.q. \forall E_j \in B \ si \ E_j \subseteq E_i \ alors \ E_j = E_i\}$$

Minimisons la FND Π'_{ct_E} vue précédemment À la Figure : 6.2.

$$min_{\subseteq}(\Pi'_{ct_E}) = \begin{array}{ll} v \wedge a \wedge okE \wedge c & \vee \\ v \wedge b \wedge okEc \wedge c & \vee \\ v \wedge \neg okE \wedge c & \end{array}$$

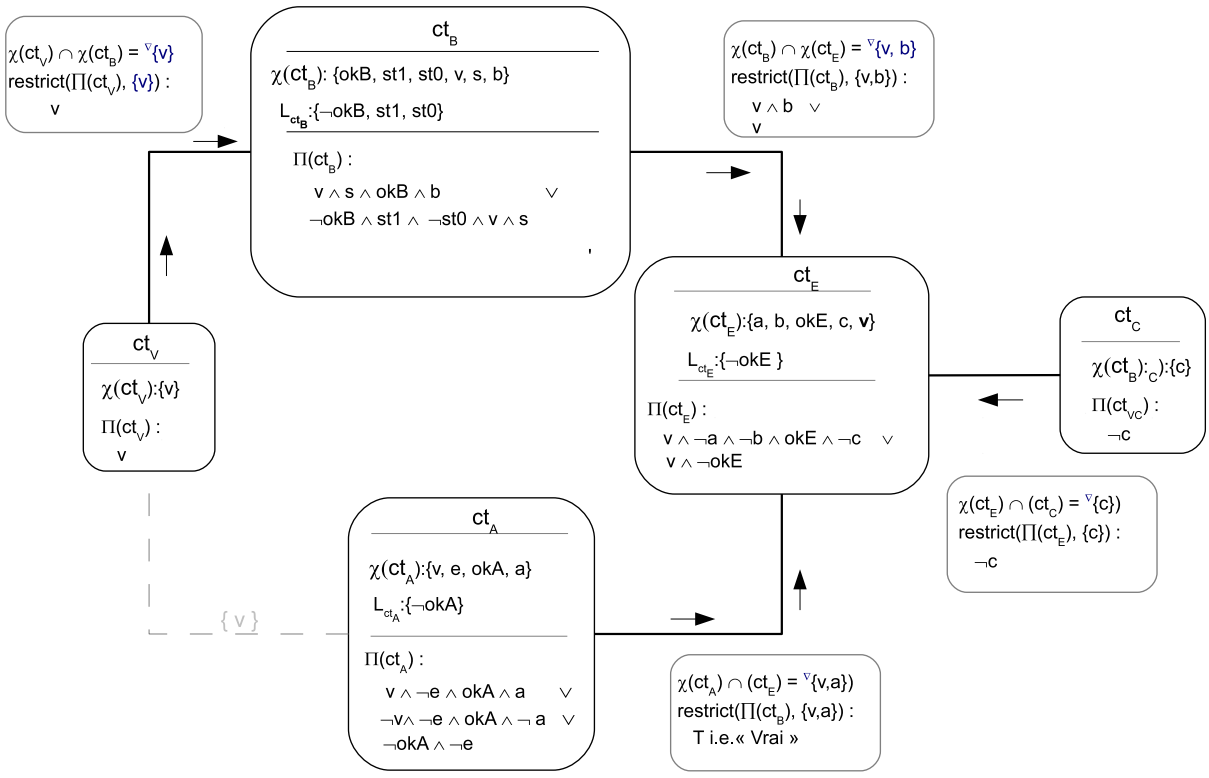


FIGURE 6.3 – Système structuré de FNDs à la fin de la phase de remonté de $ArbreJoint - \Sigma 2\Pi$

Maintenant, nous présentons l' algorithme : $ArbreJoint - \Sigma 2\Pi$ qui transforme un arbre joint de CNFs vers un arbre joint de FNDs globalement cohérentes. Nous convertissons la description de chaque cluster par la fonction $fnc2fnd(\Sigma(ct))$ (ligne 1). Dans une première phase, un

parcours de T est initié à partir de ses feuilles à la racine (ligne 2 - 3). Un clusteur envoie à son parent une FND restreinte sur le vocabulaire partagé. Lorsque qu'un cluster ct reçoit les impliquants restreints Π' d'un de ses fils (ligne 6), il met à jour sa description (ligne 7). Lorsque ct a reçu un message de tous ses fils il envoie à son tour l'expression de sa FND mise à jour à son parent ligne(10). La première phase de remonté termine lorsque le pair racine a reçu un message de tous ses fils (ligne 13). À l'issue de la phase de remonté, la formule FND de ct issue des mises à jour (cf ligne 7) avec les FNDs de ses clusters fils restreint sur le vocabulaire partagé se formalise de la façon suivante :

$$\Pi^1(ct) = \begin{cases} fnc2fnd(\Sigma(ct)), \text{ si } ct \text{ est une feuille} \\ fnc2fnd(\Sigma(ct)) \bigotimes\limits_{ct_f \in Fils} restrict(\Pi^1(ct_f), \chi(ct) \cap \chi(ct_f)) \text{ sinon.} \end{cases}$$

Intuitivement, la phase de remonté aura transformé le système structuré de FNCs $T((CT, \chi, \Sigma), E)$ vers un système structuré de FNDs : $T((CT, \chi, \Pi^2), E)$ où chaque FND $\Pi^1(ct)$ locale sera cohérente avec l'union des FND des clusters de son sous-arbre dans T. En d'autres termes, lorsque le cluster ct aura mis à jour sa FND $\Pi(ct)$ avec les FNDs reçues de ses fils et restreintes sur le vocabulaire partagé, il pourra prolonger de façon cohérente chacun de ses impliquants par une union d'impliquants locaux apparaissant dans chacunes de FNDs du sous-arbre de ct. La cohérence d'un impliquant du cluster ct avec son sous-arbre peut se montrer récursivement sur la profondeur de T et se rapproche du théorème 3 de notre travail [ADS08]. Ce théorème montre que propager les impliquants restreints sur l'expression sur les variables partagées à travers un arbre joint suffit à garantir la cohérence d'un impliquant à la racine avec les impliquants de son sous-arbre.

Algorithme 7: $ArbreJoint - \Sigma 2\Pi$
Entrée: $T((CT, \chi, \Sigma), E)$ un système structuré de FNCs
Sortie: $T((CT, \chi, \Pi), E)$ un système structuré de FNDs globalement cohérent
(1) $\Pi(ct) \leftarrow fnc2fnd(\Sigma(ct))$
(2) **if** ct **est une feuille de** T
(3) envoie $restrict(\Pi(ct), \chi(ct) \cap \chi(parent))$ à $parent$
(4)
(5) *// remonté des feuilles vers la racine*
(6) **if** ct **reçoit** Π' **d'un fils** ct_{fils}
(7) $\Pi(ct) \leftarrow \Pi(ct) \otimes \Pi'$ *// /màj FND locale par l'expression partagée des FNDs fils*
(8) **if** ct a reçu un msg de tous ses fils
(9) **if** ct n'est pas la racine
(10) envoie $restrict(\Pi(ct), \chi(ct) \cap \chi(parent(ct)))$ à $parent$
(11) **else**
(12) *// initialisation de la descente*
(13) envoie $restrict(\Pi(ct), \chi(ct) \cap \chi(ct_f))$ à tout $ct_f \in Fils(ct)$
(14)
(15) *// descente de la racine aux feuilles*
(16) **if** ct **reçoit** Π' **de son parent** $parent(ct)$
(17) $\Pi(ct) \leftarrow \Pi(ct) \otimes \Pi'$ *// /màj FND locale par l'expression partagée de la FND parent*
(18) **if** ct n'est pas une feuille
(19) envoie $restrict(\Pi(ct), \chi(ct) \cap \chi(ct'_f))$ à tout $ct'_f \in Fils(ct)$

Exemple. 6.3 *À la Figure 6.3 nous décrivons le déroulement de l'algorithme où la phase de remontée est terminée en la racine ct_E. On peut vérifier que chaque impliquant est cohérent par*

ses variables partagées avec un impliquant des clusters de son sous-arbre. Les impliquants de ct_E sont globalement cohérents.

La deuxième phase est similaire à la première phase et consiste à remettre à jour, de la racine vers les feuilles, la description FND de chaque pair de façon à ce qu'elle soit cohérente avec tout le système. Nous remarquerons qu'à la fin de la phase 1, la description FND du cluster racine est cohérente avec l'union des descriptions des clusters de son sous-arbre, nous déduisons que la FND de la racine est cohérente avec tout le système, elle est donc globalement cohérente. C'est d'ailleurs la racine (ligne 13) qui initie la seconde phase en partageant avec ses fils l'expression de sa FND globalement cohérente sur ses variables partagées. À la réception de la restriction de la FND de son parent, un fils met à jour sa FND locale (ligne 18) et propage à son tour l'expression de sa FND locale devenue globalement cohérente avec ses fils (ligne 18 -19). À l'issue de la descente, la FND de chaque cluster ct issue de la mise à jour (cf ligne 17) avec la FND de son parent peut se formaliser de la façon suivante : $\Pi^2(ct) = \begin{cases} \Pi^1(ct), \text{ si } ct \text{ est la racine} \\ \Pi^2(ct) \otimes restrict(\Pi^2(parent), \chi(ct) \cap \chi(parent)) \text{ sinon.} \end{cases}$
La phase de descente aura transformé l'arbre joint $T((CT, \chi, \Pi^1), E)$ vers l'arbre joint $T((CT, \chi, \Pi^2), E)$ où chaque FND $\Pi^2(ct)$ locale sera globalement cohérente avec l'union des FND du système. Cette propriété peut se démontrer grâce à la running intersection et à l'application de l'opération de distribution qui cette fois-ci prend en entrée une FND globalement cohérente (c-à-d la restriction de la FND reçue du parent) qu'il distribue avec une FND cohérente avec son sous-arbre (ligne 17). La FND en sortie est donc globalement cohérente.

En résumé, par l'algorithme $ArbreJoint - \Sigma 2\Pi$, nous transformons un système structuré de CNFs $T((CT, \chi, \Sigma), E)$ construit à partir de l'algorithme de TE vers un système structuré $T((CT, \chi, \Pi), E)$ où chaque formule locale $\Pi(ct)$ est une FND globalement cohérente. Par le théorème 4.1 du chapitre 4, nous déduisons que $T((CT, \chi, \Pi), E)$ garantit un diagnostic distribué correct.

D'autre part l'application successive de l'opération de distribution $\otimes$ qui met à jour les FNDs de clusters sans changer la sémantique du système initial global. Le système structuré $T((CT, \chi, \Pi), E)$ est équivalent au système initial $S : \langle P, G, K, DS \rangle$. Par la propriété 3.2 du chapitre 3 nous déduisons que $T((CT, \chi, \Pi), E)$ garantit un diagnostic distribué complet.

$T((CT, \chi, \Pi), E)$ garantit un diagnostic distribué correct et complet, nous vérifions dans la section suivante que le système $T((CT, \chi, \Pi), E)$ est un système compilé pour le diagnostic distribué.

À la Figure 6.4 nous décrivons le résultat final de l'algorithme $ArbreJoint - \Sigma 2\Pi$

6.3 Un système structuré de FNDs globalement cohérentes est un système compilé pour le diagnostic distribué

Nous avons vu qu'un réseau structuré de FNDs est une représentation compacte de la description d'un système. En dehors de ces qualités, nous avons montré dans [ADS08] qu'une FND avait un grand intérêt pour le diagnostic de systèmes. Dans un premier temps, nous avons constaté qu'il suffit juste de lire un impliquant d'une FND pour avoir un diagnostic. Autrement dit chaque impliquant contient dans son expression un diagnostic. De plus nous avons aussi montré que parmi les diagnostics contenus par les impliquants d'un système sont présents tous les diagnostics minimaux. Notre objectif est de montrer qu'à partir d'un arbre joint structuré de FNDs globalement cohérentes, il est possible de prolonger un diagnostic local par un diag-

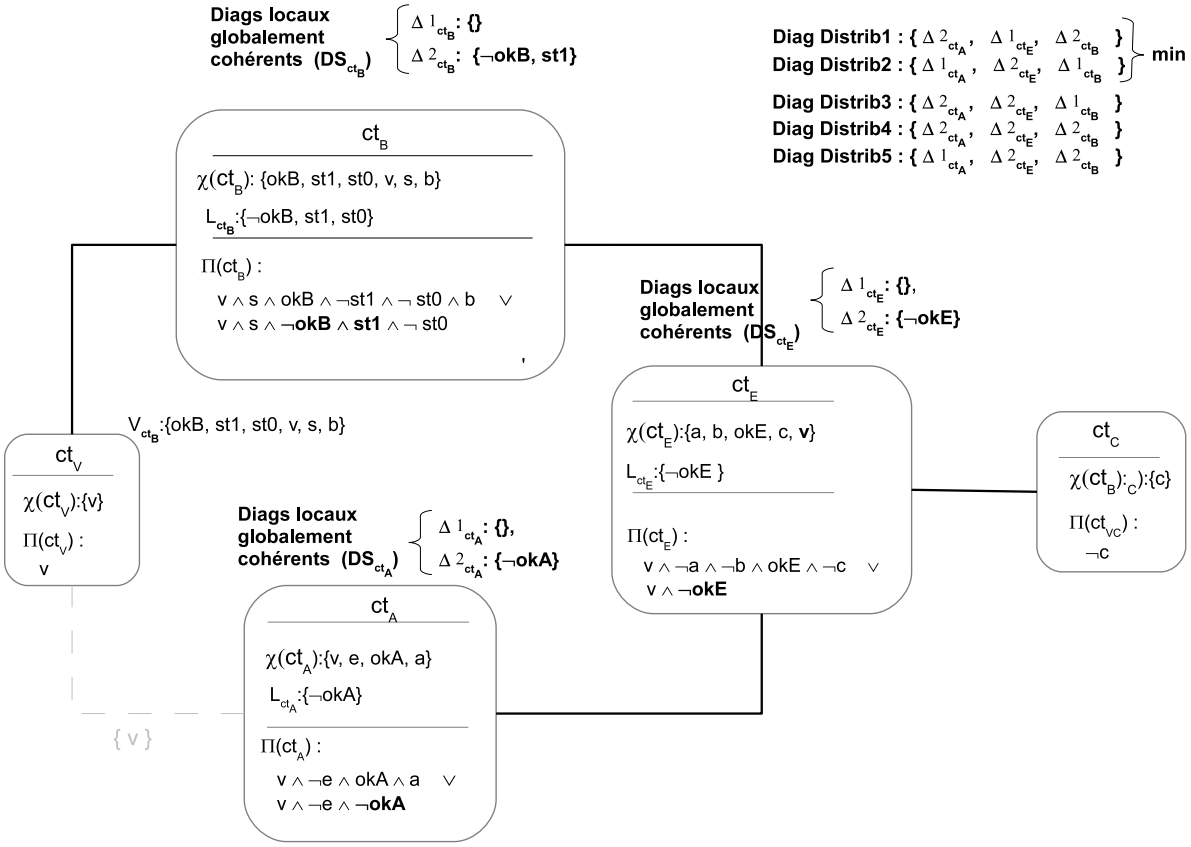


FIGURE 6.4 – Système structuré de FNDs globalement cohérentes

nostic minimal du système global. Dans cette section nous présentons nos résultats théoriques [ADS08] sur les FNDs avant de les appliquer aux réseaux de FNDs respectant la confidentialité des variables locales.

6.3.1 Un réseau de FNDs représente tous les diagnostics minimaux d'un système

Dans une première étape nous montrons que chaque impliquant d'une FND représente un diagnostic du système. Ensuite, nous montrons que n'importe quelle FND (y compris une FND qui serait induite par la distribution de FNDs locales) contient tous les diagnostics minimaux d'un système. Notons Π, la FND encodant le comportement d'un pair ou d'un système, et L l'ensemble des littéraux de mode cohérent à partir desquels sont élaborés les diagnostics. Notons $Diag(L, \Pi)$ son ensemble de diagnostics définis sur L et $min_{\subseteq}(Diag(L, \Pi))$ son ensemble de diagnostics minimaux par rapport à l'inclusion ensembliste. Nous constatons dans un premier temps que si on s'assure que L est un langage de littéraux de mode, chaque monôme de la projection $restrict(\Pi, L)$ est un diagnostic.

Exemple. 6.4 (Extraire les diagnostics d'une FND) *Reprenons l'exemple introductif et intéressons nous à la description observé du pair P_B modélisant le comportement du service de la banque. Considérons le langage cible : $L_{P_B} = \{\neg okB, st1, st0\}$. Nous décrivons le service par la formule FND suivante :*

$$\Pi_{P_B} = \begin{array}{l}(v \wedge s \wedge okB \wedge \neg st1 \wedge \neg st0 \wedge b) \vee \\ (\neg v \wedge s \wedge okB \wedge \neg st1 \wedge \neg st0 \wedge \neg b) \vee \\ (v \wedge s \wedge \neg okB \wedge st1 \wedge \neg st0) \vee \\ (\neg v \wedge s \wedge \neg okB \wedge \neg st1 \wedge st0)\end{array} \quad restrict(\Pi_{P_B}, L_{P_B}) = \begin{array}{l}\{\} \\ \{\neg okB \wedge st1\} \\ \{\neg okB \wedge st0\}\end{array}$$

Prenons le premier impliquant $I_1 : (v \wedge s \wedge okB \wedge \neg st1 \wedge \neg st0 \wedge b)$. *Si on le restreint sur le langage cible* $L_{P_B} = \{\neg okB, st1, st0\}$ *on obtient donc* $restrict(I_1, L_{P_B}) = \{\}$ *qui est un diagnostic vide. Les autres diagnostics obtenus par restriction sur le langage cible sont* $\{\neg okB \wedge st1\}$ *et* $\{\neg okB \wedge st0\}$. *Nous remarquerons que les deux premiers diagnostics sont identiques, mais proviennent d'impliquants différents.*

Nous formalisons cet exemple à travers le lemme suivant qui nous aidera à montrer qu'une FND contient aussi tous les diagnostics minimaux.

Lemme 6.1 *Soit* Π *la description d'un système observé sous forme FND et* L *un ensemble cohérent de littéraux de mode, alors*

$$\forall I \in \Pi, restrict(I, L) \in Diag(L, \Pi)$$

Preuve. Pour tout $I \in \Pi$, I est un impliquant de Π, il est donc cohérent avec Σ. Considérons l'ensemble $L \setminus restrict(I, L)$. Cet ensemble ne contient aucun littéral de $restrict(I, L)$. On en déduit que $restrict(I, L) \cup \overline{L \setminus restrict(I, L)}$ est cohérent avec I, il est donc aussi cohérent avec Π. Par définition, $restrict(I, L)$ est un diagnostic. □

On déduit du lemme 6.1 que le calcul d'un impliquant de Π contient un diagnostic. En pratique nos suppositions sur la représentation FND, qui semblent de prime abord forte, peuvent être modérées par le fait que les impliquants peuvent être calculés de façon incrémentale par un Solver SAT ([CCCB96, Cas96]). Cependant, pour de petits systèmes ou de larges systèmes mais distribués, nous supposons que la transformation de la description d'un sous-système d'une FNC vers une FND locale peut certainement être effectuée. Nous montrons maintenant que parmi les impliquants d'une FND, sont présents tous les diagnostics minimaux. Si on revient à la description du service du pair P_B on constate que deux de ses impliquants contiennent le diagnostic minimal minimal {}. Plus formellement, le théorème suivant permet d'affirmer que les diagnostics minimaux sont contenus dans n'importe quelle FND encodant le système observé.

Théorème 6.1 (Une FND contient tous les diagnostics minimaux d'un système) *Soit* Π *une description FND d'un système observé, et* L *un ensemble cohérent de littéraux de mode :*

$$min_{\subseteq}(Diag(L, \Pi)) = min_{\subseteq}(restrict(\Pi, L))$$

Preuve. $\subseteq$) Montrons que tout diagnostic minimal est la restriction minimale d'un impliquant sur un langage cohérent de littéraux. Soit Δ un diagnostic minimal, par définition $\Delta \cup \overline{\{L \setminus \Delta\}}$ est cohérent avec Π. Ainsi, il existe un impliquant I de Π cohérent avec $\Delta \cup \overline{\{L \setminus \Delta\}}$. Nous déduisons qu'aucun littéral de l'ensemble de littéraux $L \setminus \Delta$ s'opposant à $\overline{\{L \setminus \Delta\}}$ est contenu dans I : $restrict(I, L \setminus \Delta) = \emptyset$. Or $restrict(I, L) = restrict(I, \Delta) \cup restrict(I, L \setminus \Delta)$. Nous avons donc $restrict(I, L) = restrict(I, \Delta)$. De plus, Δ est un diagnostic minimal ce qui signifie que tout sous ensemble $\Delta' \subset \Delta$ uni à au moins un littéral l'de $\overline{L \setminus \Delta'}$ n'est pas cohérent avec Π . Ainsi aucun impliquant de Π ne peut contenir seul Δ' sans contenir un autre littéral de $L \setminus \Delta'$. Nous déduisons donc que $restrict(I, L) = \Delta$ et que $restrict(I, L)$ est un restriction sur L minimale cohérente avec Π.

$\supseteq$) Montrons par l'absurde que tout impliquant restreint minimal sur le langage cohérent de littéraux L est un diagnostic minimal. Soit $restrict(I, L) \in min_{\subseteq}(restrict(\Pi, L))$. Par le

lemme 6.1, nous savons que $restrict(I, L)$ est un diagnostic. Supposons qu'il ne soit pas minimal. Il existe alors un diagnostic Δ, t.q. $\Delta \subseteq restrict(\underline{I, L})$. Par conséquent il existe l dans $restrict(I, L)$ t.q. l n'est pas dans Δ. Dans ce cas $\Delta \cup \overline{\{ L \setminus \Delta\}} \cup I$ est contradictoire. Mais puisque $\Delta \cup \overline{\{ L \setminus \Delta\}}$ est cohérent avec Π, alors il existe $I' \neq I$ t.q. $\Delta \cup \overline{\{ L \setminus \Delta\}} \cup I'$ est cohérent avec $restrict(I', L) \subseteq \Delta$. Nous déduisons que $restrict(I', L) \subseteq \Delta \subseteq restrict(I, L)$. Cette dernière remarque est contradictoire avec l'hypothèse $restrict(I, L) \in min_{\subseteq}(restrict(\Pi, L))$. □

Dans [ADS08] ce théorème est énoncé pour une FND quelconque, il s'applique au cas distribué où le système est modélisé par un réseau de FNDs. Pour cela, il suffit de remarquer que tout diagnostic global minimal se représente par un diagnostic distribué Δ^D construit à partir d'impliquants locaux restreints sur leur propre langage cible. Toutefois essayer de prolonger un impliquant local par un diagnostic global peut être interdit, notamment si les littéraux de modes doivent rester confidentiels. Afin de sélectionner les diagnostics locaux les plus intéressants, nous chercherons à vérifier qu'il appartiennent à un diagnostic global minimal.

6.3.2 Vérifier qu'un diagnostic local appartient à un diagnostic distribué minimal

À l'aide du réseau structuré de FNDs globalement cohérent, nous montrons qu'il est possible de vérifier que pour tout cluster que tout diagnostic local issu de la restriction d'un impliquant local est inclus dans un diagnostic distribué minimal. Tout d'abord, nous rappelons qu'un diagnostic distribué $\Delta^D = \{\Delta_{ct} : ct \in CT\}$ est un ensemble de diagnostic locaux, où chaque diagnostic local a été décidé par chaque cluster et où l'union des diagnostics locaux est cohérente avec le système global. Un diagnostic distribué est dit minimal ssi le diagnostic global qu'il induit (c-à-d l'union de diagnostic locaux qui le composent) est un diagnostic minimal de la description d'un système global.

Intuitivement, afin de vérifier qu'un diagnostic local globalement cohérent d'un cluster Δ_{ct} appartient à un diagnostic distribué minimal, nous vérifions dans un premier temps que Δ_{ct} est un diagnostic minimal de ct. Dans ce chapitre nous faisons l'hypothèse que les variables de modes (par exemple okX, okY,..) servant à exprimer les diagnostics sont des variables locales aux clusters. Dans ce cas, nous avons que tout diagnostic local minimal globalement cohérent d'un cluster appartient à un diagnostic distribué minimal. En effet, soit Δ_{ct} un diagnostic local minimal globalement cohérent. Δ_{ct} est un diagnostic local minimal, il n'existe donc aucun diagnostic formé sur $vars(\Delta_{ct})$ qui le sous-somme. Δ_{ct} est aussi un diagnostic local globalement cohérent, il existe donc un ensemble de diagnostics distribués qui prolongent Δ_{ct}. Les diagnostics minimaux globaux prolongeant Δ_{ct} sont des diagnostics minimaux distribués de tout le système. Ils ne peuvent être sous-sommés sur $vars(\Delta_{ct})$ par aucun diagnostic distribué ne prolongeant pas (Δ_{ct} et par définition ne peuvent être sous-sommé par aucun diagnostic distribué prolongeant Δ_{ct}.

De plus, en étendant la notion de diagnostic minimal d'un cluster au diagnostic minimal d'un ensemble de clusters, nous déduisons aussi qu'un diagnostic minimal d'un ensemble de clusters fait partie d'un diagnostic distribué minimal.

Par simplicité nous réunissons les différentes notions de diagnostic local d'un cluster, diagnostic local d'un ensemble de cluster et diagnostic distribué à travers la notion de sous-diagnostic distribué.

Définition 6.4 (Sous-diagnostic distribué, sous-diagnostic distribué minimal) *Soient un arbre joint de FNDs globalement cohérent $T((CT, \chi, \Pi), E)$ $CT' \subseteq CT$. Un sous-diagnostic distribué $\Delta^{CT'}$ est un ensemble de diagnostics locaux de CT' issu de restrictions d'impliquants de*

FNDs, cohérent avec l'union des descriptions des clusters CT'.

$$\Delta^{CT'} = \{restrict(\Pi(ct'), L_{ct'}) : ct' \in CT'\} \text{ t.q. } \Delta^{CT'} \bigwedge_{ct' \in CT'} \Pi(ct') \not\models \bot$$

Un sous-diagnostic distribué $\Delta^{CT'}$ est minimal, s'il n'existe pas de sous-diagnostic distribué $\Delta_2^{CT'}$ formé sur CT' t.q. $\Delta_2^{CT'} \subset \Delta^{CT'}$

Compte tenu de cette définition, un diagnostic local est un sous-diagnostic distribué défini sur un cluster. De même, parce que chaque impliquant de chacune des FNDs est globalement cohérent, tout sous-diagnostic distribué se prolongera toujours par un diagnostic distribué sur tous les clusters. Enfin, dans la section précédente nous avons vu que n'importe quelle FND représentait parmi ses impliquants tous ses diagnostics minimaux. On en déduit que la FND qui serait induite par la distribution des FNDs locales à chaque cluster contiendrait les diagnostics minimaux. Intuitivement, on comprend que parmi tous les sous-diagnostics distribués construits sur CT sont contenus les diagnostics distribués minimaux. Afin d'identifier les diagnostic locaux d'un système qui sont contenus dans un diagnostic global minimal, nous avons la propriété suivante sur les sous-diagnostics distribués :

Propriété 6.1 (sous-diagnostic distribué minimal) *Étant donné un arbre joint de FNDs globalement cohérent $T((CT, \chi, \Pi), E)$, un sous-diagnostic distribué $\Delta_i^{CT'}$ fait partie d'un diagnostic distribué minimal des clusters CT t.q $CT' \subseteq CT$ ssi*

1. *$\Delta_i^{CT'}$ est un sous-diagnostic distribué minimal*
2. *sinon il existe un ensemble $Diag^{\subset}$ de sous-diagnostics distribués le sous-sommant t.q. $Diag^{\subset} = \{\Delta_k^{CT'} \text{ t.q. } \Delta_k^{CT'} \subset \Delta_i^{CT'}\}$ et $\Delta_j^{CT \setminus CT'}$ où :*
   (a) *$\Delta_i^{CT'} \cup \Delta_j^{CT \setminus CT'}$ est un diagnostic distribué minimal et*
   (b) *pour tout $\Delta_k^{CT'} \in Diag^{\subset}$ $\Delta_k^{CT'} \cup \Delta_j^{CT \setminus CT'}$ n'est pas un sous-diagnostic distribué.*

Dans notre algorithme 6.3.2 132 "EstDansMinGlobal" nous mettons à profit cette propriété pour vérifier si un diagnostic local $\Delta*$ se prolonge par un diagnostic distribué minimal. Pour cela, nous vérifions incrémentalement, en passant d'un cluster à l'autre si au moins un sous-diagnostic distribué prolongeant $\Delta*$ est minimal.

Exemple. 6.5 *Reprenons le système de FNDs globalement cohérentes du cluster ct_E de la Figure 6.4. Supposons que ct_E veuille savoir si son diagnostic local $\Delta^2_{ct_E} = \{\neg okE\}$ fait partie d'un diagnostic distribué minimal. Tout d'abord il vérifie si $\Delta^2_{ct_E}$ est minimal (cf point 1 de la propriété précédente). Ce n'est pas le cas, $\Delta^1_{ct_E} = \{\}$ le sous-somme. (On passe au point 2) on cherche donc à prolonger $\Delta^2_{ct_E}$ par un diagnostic Δ' d'un des pairs voisins afin que Δ' et $\Delta^2_{ct_E}$ forme un sous-diagnostic distribué minimal. Nous rappelons que tout sous-diagnostic distribué minimal fait partie d'un diagnostic minimal global. En raison de la politique de confidentialité, ct_E ne pourra pas envoyer son diagnostic $\Delta^2_{ct_E}$ à ses voisins. En revanche, il pourra envoyer des formules définies sur le vocabulaire partagé avec ces derniers. Il cherchera à prolonger le diagnostic $\Delta^2_{ct_E}$ par les impliquants qui le contiennent par $\Pi^{=}$. Afin de vérifier la minimalité chez le pair voisin ct_E va aussi chercher aussi à prolonger les diagnostics qui sous-somment $\Delta^2_{ct_E}$ de $\Pi^{\subset}$. De même, en raison de la confidentialité, la totalité d'un impliquant ne pourra être transmise aux pairs voisins. Seule son expression sur le vocabulaire partagé sera communiquée. Cela suffit pour prolonger de façon cohérente les diagnostics induits par les ensembles $\Pi^{=}$ et $\Pi^{\subset}$. Dès qu'un impliquant local d'un pair voisin, cohérent avec $\Pi^{=}$, contient un diagnostic local non sous-sommé par l'un des diagnostics locaux induits par $\Pi^{\subset}$, c'est gagné. On est sûr que $\Delta^2_{ct_E}$ fait partie d'un sous-diagnostic distribué minimal.*

Plus formellement on suppose que le protocole s'exécute à travers un arbre joint enraciné au cluster initiateur $ct*$. Tout d'abord, (ligne 1-3) le cluster initiateur partage ses impliquants locaux $\Pi(ct*)$ en deux ensembles disjoints : $\Pi^{\subset}$ qui regroupe les impliquants ayant un diagnostic local sous-sommant $\Delta*$ et $\Pi^{=}$ qui regroupe les impliquants prolongeant $\Delta*$. Si $\Delta*$ est minimal c-à-d $\Pi^{\subset}$ est vide (ligne 5) $ct*$ retourne $Vrai$. Si ce n'est pas le cas et il n'y pas d'autres fils à explorer $ct*$ retourne faux (ligne 6). De la ligne (8 à 11) on cherche à prolonger $\Delta*$ par un sous-diagnostic distribué minimal. $ct*$ communique à chaque fils ct_f par un message $ReqDiagMin$ à la fois les impliquants restreints sur le vocabulaire partagé ($\Pi^{=}_{ct_f}$) prolongeant $\Delta*$ (ligne 9), mais aussi les restrictions d'impliquants porteurs d'un plus petit diagnostic $\Pi^{\subset}_{ct_f}$ (ligne 10).

Algorithme 8: EstDansMinGlobal
Entrée: $\Delta*$ un diagnostic local d'un cluster $ct*$
Sortie: Booléen : { Vrai, Faux} indique si $\Delta*$ fait partie d'un diagnostic global minimal
(1) *// initialisation de la vérification par $ct*$ qui devient la racine du arbre joint*
(2) $\Pi^{=} \leftarrow \{I \in \Pi | restrict(I, L_{ct*}) = \Delta*\}$
(3) $\Pi^{\subset} \leftarrow \{I \in \Pi | restrict(I, L_{ct*}) \subset \Delta*\}$
(4) *// cas d'arrêts la vérification*
(5) retourne Vrai si $\Pi^{\subset}$ est vide
(6) retourne Faux si $ct*$ n'a pas de fils
(7) *// vérification dans les sous-arbres de $ct*$*
(8) **foreach** $ct_f \in Fils(ct*)$
(9) $\Pi^{=}_{ct_f} \leftarrow restrict(\Pi^{=}, \chi_{ct*} \cap \chi_{ct_f})$
(10) $\Pi^{\subset}_{ct_f} \leftarrow restrict(\Pi^{\subset}, \chi_{ct*} \cap \chi_{ct_f})$
(11) envoie $ReqDIAGMIN(\Pi^{=}_{ct_f}, \Pi^{\subset}_{ct_f})$ à ct_f
(12)
(13) *// vérification dans une branche partant de $ct*$*
(14) **if** ct **reçoit** $ReqDIAGMIN(\Pi^{=}_{ct'}, \Pi^{\subset}_{ct'})$ de son parent
(15) $\Pi^{=} \leftarrow \{I \in \Pi | \exists J \in \Pi^{=}_{ct'}, I \wedge J \not\models \bot\}$
(16) $\Pi^{\subset} \leftarrow \{I \in \Pi | \exists J \in \Pi^{\subset}_{ct'}, I \wedge J \not\models \bot\}$
(17) *// cas d'arrêts de la vérification dans la branche et réponse au parent*
(18) renvoie $RepsDIAGMIN(Vrai)$ si
(19) $\exists \Delta_i \in restrict(\Pi^{=}, L_{ct})$ t.q. $\forall \Delta_k \in restrict(\Pi^{\subset}, L_{ct}) : \Delta_k \not\subseteq \Delta_i$
(20) renvoie $RepsDIAGMIN(Faux)$ si ct est une feuille
(21) *// vérification dans les sous-arbres de ct*
(22) **foreach** $ct_f \in Fils(ct)$
(23) $\Pi^{=}_{ct_f} \leftarrow restrict(\Pi^{=}, \chi_{ct} \cap \chi_{ct_f})$
(24) $\Pi^{\subset}_{ct_f} \leftarrow restrict(\Pi^{\subset}, \chi_{ct} \cap \chi_{ct_f})$
(25) envoie $ReqDIAGMIN(\Pi^{=}_{ct_f}, \Pi^{\subset}_{ct_f})$ à ct_f
(26)
(27) *// remonté des réponses des feuilles vers la racine*
(28) **if** ct **reçoit** $RepsDIAGMIN(b)$ d'un fils ct'
(29) $brancheContientMin \leftarrow brancheContientMin \vee b$
(30) **if** ct a reçu $RepsDIAGMIN$ de tous ses fils ct'
(31) **if** $ct \neq ct*$
(32) envoie $RepsDIAGMIN(brancheContientMin)$ à parent
(33) **else**
(34) retourne $brancheContientMin$

Lorsqu'un fils *ct* reçoit une requête $ReqDiagMin$ (ligne 14), il sélectionne dans $\Pi^{=}$ ses impliquants locaux (indirectement ses diagnostics locaux) cohérents avec un prolongement de $\Delta*$ (ligne 15). Dans $\Pi^{\subset}$ il sélectionne ses impliquants locaux cohérents avec les sous-diagnostics distribués sous-sommant les sous-diagnostics distribués prolongés par $\Delta*$ (ligne 16). Un sous-diagnostic distribué minimal prolongeant $\Delta*$ est trouvé si un diagnostic Δ_i de $\Pi^{=}$ n'est sous-sommé par aucun diagnostic de $\Pi^{\subset}$ (ligne 17-19). S'il n'y a pas de possibilité de prolonger $\Delta*$ par un sous-diagnostic distribué minimal (on a atteint une feuille, ligne 20). Sinon, s'il y a encore une possibilité, *ct* fait appel à ses fils (ligne 21 à 25). De la ligne 27 à 34 on organise la remonté des réponses de façon à détecter la terminaison. Chaque pair atteint par une requête $ReqDIAGMIN$ attend les réponses de ses fils avant de répondre. On peut aller plus vite, à partir du moment où l'un des fils se trouve dans un prolongation contenant un sous-diagnostic distribué minimal de $\Delta*$, le cluster peut répondre.

Nous n'avons pas eu le temps d'implémenter et de tester cette approche. Toutefois nous notons qu'au pire des cas cet algorithme nécessite un nombre de message de l'ordre du diamètre du réseau structuré. Chaque message contient au pire une formule de la taille de la plus grande des FNDs du réseau. On en déduit que la complexité de vérification est linéaire en la largeur de $T((CT, \chi, \Pi), E)$. $T((CT, \chi, \Pi), E)$ garanti un diagnostic distribué compilé.

6.3.3 Résumé du chapitre

En conclusion, dans ce chapitre, nous repartons du résultat de Token Elimination qui décompose le schéma d'un système vers un schéma structuré du cluster qui respecte les accointances des pairs et la confidentialité des variables locales. Nous enrichissons les clusters des descriptions FNCs de chaque pair et formons ainsi un système structuré (arbre joint de CNFs). Afin que ce système garantisse un diagnostic correct et complet, nous compilons l'arbre joint de FNCs vers un arbre joint de FNDs globalement cohérentes. Nous montrons que le système structuré de FNDs résultat modélise aussi l'ensemble des diagnostics minimaux du système. En raison de la politique de confidentialité où les variables de mode peuvent rester locales, nous avons exhibé un algorithme capable de vérifier si les diagnostics locaux encodés dans chacun des impliquants d'une FND locale fait partie d'un diagnostic global minimal. Cet algorithme s'exécute en un temps linéaire par rapport à la taille de chaque description. Le système structuré de FNDs globalement cohérent est donc un système compilé pour le diagnostic distribué.

Chapitre 7

Conclusion

Les applications cherchant à préserver la confidentialité de leurs données sont omniprésentes sur le web. Il y a donc un intérêt croissant à pouvoir analyser le comportement de ces systèmes tout en respectant la confidentialité des données des applications sous-jacentes.

Le défi majeur du diagnostic distribué est de pouvoir expliquer le comportement d'un système distribué par un ensemble de pairs ayant un accès et une visibilité réduite de ce système, tout comme l'aurait fait un unique diagnostiqueur ayant la connaissance globale de tout le système.

Ainsi, en plus des questions usuelles de mises à jour, d'espace mémoire et de rapidité des réponses auxquelles répondent les approches interprétées ou compilées de diagnostic, le contexte distribué apporte de nouvelles contraintes telles que la répartition géographique ainsi que le contrôle d'accès des descriptions manipulées. La plupart des approches envisagées dans le cadre du diagnostic distribué supposent l'existence d'un superviseur global et ne prévoient pas de politique d'accès sur le partage du vocabulaire des descriptions du système. En présence d'applications où l'accès à l'information est restreint, ces approches peuvent donc être inadaptées. À notre connaissance, seuls le système d'inférence distribué Somewhere [ACG+06] ou plus récemment Deca [AGR09], auxquels notre étude fait suite, cherchent à préserver la confidentialité des connaissances locales. Dans le contexte des CSPs ou COPs distribués, la problématique de confidentialité est plus avancée et a été étudiée par Silaghi et al [SF02], Yokoo et al [YSH02], Greenstadt et al [GPBT06] et plus récemment par Leaute et al [LF11]. Dans ces travaux la confidentialité s'applique à des systèmes où toute l'assignation de variables est cryptée mais peut circuler librement à travers le réseau. Seuls certains pairs pourront la décrypter. Par rapport à ces approches, la politique d'accès à laquelle nous nous adressons dans un premier temps n'autorise ni aux variables ni à leurs assignations de circuler librement à travers le réseau. Un pair qui n'a pas l'accès à une variable n'en n'aura jamais la connaissance et ne sera jamais amené à la considérer durant son raisonnement.

Dans ce contexte, nous présentons (dans le chapitre 3) un nouveau cadre pour le diagnostic distribué : *les systèmes de pairs diagnostiqueurs* qui permettent d'expliquer les comportements anormaux d'un système distribué géographiquement réparti et où le partage est contrôlé par une politique d'accès. Dans un système de pairs diagnostiqueurs, initialement, chaque pair décrit le comportement d'un sous-système par une formule propositionnelle. En raison de la politique d'accès, chaque pair ne pourra qu'enrichir sa description locale de nouvelles connaissances autorisées par la politique d'accès, faisant évoluer le système de pairs initial vers un système appelé *accessible*. C'est à partir d'une description accessible locale que chaque pair pourra élaborer des diagnostics. Grâce à ses liens d'accointances (son voisinage), un pair pourra aussi vérifier que les

explications locales calculées à partir de descriptions auxquelles il a accès sont aussi cohérentes avec les descriptions de pairs voisins. Ces dernières explications sont des *diagnostics localement cohérents*, elles constituent ce que chaque pair est en mesure de calculer. En raison de la politique d'accès, il sera souvent impossible à un pair du système d'expliquer le comportement global de ce dernier juste à partir d'une description locale auquel il a accès. En revanche le comportement global d'un système pourra être expliqué par un ensemble de pairs. Dans ce cadre l'objectif du diagnostic distribué d'un système de pairs diagnostiqueurs consiste à transformer le système modélisant le comportement distribué vers un système qui garantit les deux propriétés suivantes :

1. (correction) Tout diagnostic localement cohérent fait partie d'une explication cohérente du comportement du système global
2. (complétude) Toute explication cohérente du comportement du système global se retrouve à travers un ensemble de diagnostics localement cohérents répartis à travers le système
3. (compilation) Peut-on vérifier en un temps polynomial en la taille des descriptions des pairs qu'un diagnostic local peut se prolonger par un diagnostic global minimal ?

On comprend mieux que l'objectif n'est pas forcément de regrouper les diagnostics globaux d'un système (car ce ne serait pas toujours possible) mais plutôt de pouvoir transformer un système de pairs vers un système accessible où il sera possible de vérifier rapidement (ou de composer) tout diagnostic global minimal à partir de diagnostic locaux.

Compte tenu de la problématique, avant même d'envisager de résoudre,notre première étape a été de répondre aux questions suivantes :

Quels sont les systèmes qui permettent de garantir un diagnostic distribué correct et complet ? Peut-on les caractériser ?

Cette caractérisation est importante car elle nous permet d'identifier les systèmes accessibles qui répondent à notre objectif. À la fin du chapitre 3 nous avons montré que si le système accessible à partir duquel les pairs élaborent leurs diagnostics locaux est sémantiquement équivalent au système observé initial (c-à-d la sémantique du système global reste inchangé alors que les descriptions locales ont pu évoluer) alors il garantit un diagnostic distribué complet. Ensuite nous montrons que la propriété de la running intersection est une propriété fondamentale dans la caractérisation des systèmes accessibles garantissant un diagnostic distribué correct. Dans le chapitre 4 nous nous intéressons aux caractéristiques graphiques des systèmes permettant de garantir un diagnostic distribué correct. Tout d'abord nous faisons correspondre les systèmes garantissant un diagnostic distribué correct aux systèmes pour lesquels tous les modèles locaux sont globalement cohérents. Cette correspondance permet aussi de reporter la caractérisation des systèmes (toujours en se basant uniquement sur une représentation graphique de ceux-ci) garantissant un diagnostic distribué correct à la caractérisation des systèmes garantissant l'équivalence entre cohérence locale et cohérence globale des descriptions locales. À la base de la caractérisation nous retrouvons donc, de manière peu surprenante, la propriété de running intersection. Celle-ci nous permet de définir les schémas joints (c-à-d schémas d'un système de pairs où les pairs contenant une même variable forment un graphe connexe). Nous montrons que

les systèmes structurés (c-à-d dont le schéma contient un arbre couvrant joint) suffisent à garantir un diagnostic distribué correct.

Cette propriété étend un résultat bien connu de CSP et de Base de Données à des graphes dont les noeuds sont étiquetés par des ensembles de variables. Nous montrons par ailleurs que dans

le cas des systèmes d'interactions (c-à-d les systèmes où les pairs contenant une même variable forment une clique), les systèmes structurés garantissant la cohérence distribuée sont nécessaires. Cette propriété constitue l'un des résultats majeurs de cette caractérisation qui nous permet de conclure que :

Dans cas des systèmes d'interactions, seuls les schémas structurés sont en mesure de garantir un diagnostic distribué correct.

Dans le chapitre 5, nous cherchons à transformer la description d'un système de pair initial vers un système structuré en vue de le diagnostiquer. Pour cela, nous définissons l'opération de décomposition arborescente distribuée qui étend la définition de décomposition arborescente introduite par Robertson et Seymour [RS86], à notre contexte distribué. Dans ce nouveau contexte, nous avons cherché à satisfaire un politique d'accès particulière : la confidentialité (le vocabulaire local à un pair ne pourra pas être divulgué). Nous avons proposé un nouvel algorithme de décomposition arborescente distribué TE basé sur des élections locales avec passage d'un jeton. Nous avons aussi vu que la décomposition arborescente est aussi connue pour accélérer la résolution d'un problème. La qualité de la décomposition arborescente est primordiale. Une bonne technique de décomposition arborescente peut offrir des gains exponentiels. Il est donc important, même si cette technique est coûteuse, de pouvoir l'appliquer en pratique. Nous avons expérimenté notre algorithme TE sur les familles de réseaux "petit monde" connues pour caractériser certaines structures importantes que l'on peut retrouver dans certaines applications du monde réel. Afin de pouvoir le comparer à d'autres méthodes de décomposition de référence de l'état de l'art du contexte centralisé et du contexte distribué, nous l'avons adapté afin qu'il s'applique à un graphe d'interactions.

La qualité des arbres joints produits par TE sur des instances petit monde est nettement meilleure que l'état de l'art des algorithmes distribués, et reste comparable à l'état de l'art des décompositions arborescentes basées sur l'élimination de variables dans un contexte centralisé.

Dans le dernier chapitre nous repartons de la décomposition arborescente distribuée construite par TE à travers le réseau et l'enrichissons des différentes descriptions des pairs afin du former un système structuré de DNFs. Dans cette approche, nous restreignons volontairement les politiques d'accès possibles aux variables, en nous limitant uniquement au caractère privé/public des variables (les variables partagées entre au moins deux pairs seront forçément publiques). Dès lors, le système stucturé garantit un diagnostic distribué complet. Il est équivalent au système initial (seule la forme des théories locales est passée de CNFs à des FNDs). Ce système structuré de FNDs ne garantit pas encore un diagnostic distribué correct. Il faut pour cela que toutes ses FNDs soient à la fois localement et globalement cohérentes. Pour cela, nous utilisons un parcours en 2 phases qui transforme chaque FND locale en FND globalement cohérente. En plus de garantir un diagnostic distribué correct nous montrons qu'une structure de FNDs globalement cohérente est aussi un système compilé pour le diagnostic distribué. Pour cela nous mettons en évidence un algorithme où chaque pair sera en mesure de vérifier que chacun des ses diagnostics locaux fait partie d'un diagnostic global minimal. Nous nous basons sur le théorème 1 de [ADS08] qui montre que toute FND d'un système représente aussi ses diagnostics minimaux. Enfin, en plus d'offrir un système compilé pour le diagnostic distribué, nous remarquons qu'en théorie, tout arbre joint structuré de DNFs globalement cohérentes est susceptible d'offrir une représentation plus succincte qu'un OBDD.

Nous avons donc montré dans ce manuscrit qu'il était possible d'appréhender la problématique de diagnostic distribué par une approche novatrice, qui ne considère qu'à minima la vision

globale du système. Dans cette approche, les variables privées restent privées, et aucun pair n'a la connaissance de la défaillance des autres pairs. Chaque pair peut néanmoins réagir à la présence de défaillances locales expliquant la défaillance globale du système. On peut donc imaginer que cette simple connaissance soit suffisante pour que le pair puisse réagir en conséquence et lancer une procédure de réparation, sans obligatoirement en avertir les autres pairs. Bien entendu, beaucoup reste à faire et cette thèse ouvre de nombreuses perspectives dans cette voie. Ainsi, il serait important non pas de considérer la présence d'une défaillance locale dans l'un des diagnostics globaux, mais plutôt la présence d'une défaillance locale dans l'un des diagnostics les plus probables, globalement. Dans cette voie prometteuse, beaucoup reste à faire. Il faudrait aussi prendre en compte la haute dynamicité d'un véritable réseau de pairs, ce qui permettrait de prendre en compte des observations changeantes ou oscillantes, le départ ou l'arrivée de pairs (comment maintenir l'état compilé du réseau ?), ou encore la réparation d'un pair suite à un diagnostic défavorable.

Bibliographie

[ACG+06] Philippe Adjiman, Philippe Chatalic, François Goasdoué, Marie-Christine Rousset, and Laurent Simon. Distributed reasoning in a peer-to-peer setting : Application to the semantic web. *Journal of Artificial Intelligence Research (JAIR)*, 25 :269–314, 2006.

[ADS08] Vincent Armant, Philippe Dague, and Laurent Simon. Distributed consistency-based diagnosis. In *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings*, pages 113–127, 2008.

[AGR09] Nada Abdallah, François Goasdoué, and Marie-Christine Rousset. Dl-liter in the light of propositional logic for decentralized data management. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 2010–2015, 2009.

[AS09] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern sat solvers. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 399–404, 2009.

[ASD12] Vincent Armant, Laurent Simon, and Philippe Dague. Distributed tree decomposition with privacy. In *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, pages 102–117, 2012.

[BA99] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439) :509–512, Oct 1999.

[BATJ91] Tom Bylander, Dean Allemang, Michael C. Tanner, and John R. Josephson. The computational complexity of abduction. *Artificial Intelligence*, 49 :25–60, May 1991.

[BC93] Christian Bessière and Marie-Odile Cordier. Arc-consistency and arc-consistency again. In *Proceedings of the 11th National Conference on Artificial Intelligence. Washington, DC, USA, July 11-15, 1993*, pages 108–113, 1993.

[Bel58] Richard Bellman. On a routing problem. *Quaterly of Applied Mathematics*, 1 :87–90, 1958.

[BFK93] C. Beckstein, R. Fuhge, and G. Kraetzschmar. Supporting assumption-based reasoning in a distributed environment. *Workshop on Distributed Artificial Intelligence*, 1993.

[BFN06] J. Biteus, E. Frisk, and M. Nyberg. Distributed diagnosis by using a condensed local representation of the global diagnoses with minimal cardinality. In *DX-06, International Workshop on the Principles of Diagnosis*, 2006.

[BG95] Fahiem Bacchus and Adam J. Grove. On the forward checking algorithm. In *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, pages 292–308, 1995.

[BHFJ03] Albert Benveniste, Stefan Haar, Eric Fabre, and Claude Jard. Distributed monitoring of concurrent and asynchronous systems. In *CONCUR*, pages 1–26, 2003.

[Bie08] Armin Biere. Picosat essentials. *Journal on Satisfiability, Boolean Modeling and Computation*, 4(2-4) :75–97, 2008.

[BJ99] Fabrice Bouquet and P. Jégou. ROBDD : Une étude sur les ordres et stratégies de construction. In *5èmes Journées Nationales sur la Résolution Pratique de Problèmes NP-Complets, JNPC'99*, pages 207–215, Lyon, France, June 1999.

[BMBM05] Christian Bessière, Arnold Maestre, Ismel Brito, and Pedro Meseguer. Asynchronous backtracking without adding links : a new member in the abt family. *Artificial Intelligence*, 161(1-2) :7–24, 2005.

[BNF08] Jonas Biteus, Mattias Nyberg, and Erik Frisk. An algorithm for computing the diagnoses with minimal cardinality in a distributed system. *Engineering Applications of Artificial Intelligence*, 21(2) :269–276, 2008.

[Bod98] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theorical Computer Science*, 209(1-2) :1–45, 1998.

[Cas96] Thierry Castell. Computation of prime implicates and prime implicants by a variant of the Davis and Putnam procedure. In *Eigth International Conference on Tools with Artificial Intelligence (ICTAI '96), November 16-19, 1996, Toulouse, France*, pages 428–429, 1996.

[CCCB96] Thierry Castell, Claudette Cayrol, Michel Cayrol, and Daniel Le Berre. Using the Davis and Putnam procedure for an efficient computation of preferred models. In *ECAI 96,12th European Conference on Artificial Intelligence, Budapest, Hungary, August 11-16, 1996, Proceedings*, pages 350–354, 1996.

[CG07] Marie-Odile Cordier and Alban Grastien. Exploiting independence in a decentralised and incremental approach of diagnosis. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 292–297, 2007.

[CL99] Christos G. Cassandras and Stéphane Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, September 1999.

[CNR06] Philippe Chatalic, Gia Hien Nguyen, and Marie-Christine Rousset. Reasoning with inconsistencies in propositional peer-to-peer inference systems. In *ECAI 2006, 17th European Conference on Artificial Intelligence, August 29 - September 1, 2006, Riva del Garda, Italy, Including Prestigious Applications of Intelligent Systems (PAIS 2006), Proceedings*, pages 352–356, 2006.

[CPD07] Luca Console, Claudia Picardi, and Daniele Theseider Dupré. A framework for decentralized qualitative model-based diagnosis. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 286–291, 2007.

[CPR02] Luca Console, Claudia Picardi, and Marina Ribaudo. Process algebras for systems diagnosis. *Artificial Intelligence*, 142(1) :19–51, 2002.

[CS01] Philippe Chatalic and Laurent Simon. Multiresolution for sat checking. *International Journal on Artificial Intelligence Tools*, 10(4) :451–481, 2001.

[Dar98] Adnan Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research (JAIR)*, 8 :165–222, 1998.

[Dar04] Adnan Darwiche. New advances in compiling cnf into decomposable negation normal form. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, pages 328–332, 2004.

[DD96] Rina Dechter and Avi Dechter. Structure-driven algorithms for truth maintenance. *Artificial Intelligence*, 82(1-2) :1–2, 1996.

[Dec92] Rina Dechter. From local to global consistency. *Artificial Intelligence*, 55 :87–107, May 1992.

[Dec99] Rina Dechter. Bucket elimination : A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2) :41–85, 1999.

[Dec03] Rina Dechter. *Constraint processing*. Elsevier Morgan Kaufmann, 2003.

[Dec06] Rina Dechter. Tractable structures for constraint satisfaction problems. In *Handbook of Constraint Programming, part I, chapter 7*, pages 209–244. Elsevier, 2006.

[DF02] Rina Dechter and Daniel Frost. Backjump-based backtracking for constraint satisfaction problems. *Artificial Intelligence*, 136(2) :147–188, 2002.

[DGG+08] Artan Dermaku, Tobias Ganzow, Georg Gottlob, Benjamin J. McMahan, Nysret Musliu, and Marko Samer. Heuristic methods for hypertree decomposition. In *MICAI 2008 : Advances in Artificial Intelligence, 7th Mexican International Conference on Artificial Intelligence, Atizapán de Zaragoza, Mexico, October 27-31, 2008, Proceedings*, pages 1–11, 2008.

[Dij59] Edsger Wybe Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[dK86] Johan de Kleer. An assumption-based tms. *Artificial Intelligence*, 28(2) :127–162, 1986.

[dK89] Johan de Kleer. A comparison of atms and csp techniques. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence. Detroit, MI, USA, August 1989*, pages 290–296, 1989.

[dKMR92] Johan de Kleer, Alan K. Mackworth, and Raymond Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2-3) :197–222, 1992.

[dKW87] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1) :97–130, 1987.

[dKW89] Johan de Kleer and Brian C. Williams. Diagnosis with behavioral modes. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence. Detroit, MI, USA, August 1989*, pages 1324–1330, 1989.

[DM02] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research (JAIR)*, 17 :229–264, 2002.

[DP89] Rina Dechter and Judea Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38(3) :353–366, 1989.

[EB05] Niklas Eén and Armin Biere. Effective preprocessing in sat through variable and clause elimination. In *Theory and Applications of Satisfiability Testing, 8th International Conference, SAT 2005, St. Andrews, UK, June 19-23, 2005, Proceedings*, pages 61–75, 2005.

[EBBB08] Redouane Ezzahir, Christian Bessiere, Imade Benelallam, and Mustapha Belaissaoui. Asynchronous breadth first search dcop algorithm. *Applied Mathematical Sciences*, 2(37-40) :1837–1854, 2008.

[FB07] Eric Fabre and Albert Benveniste. Partial order techniques for distributed discrete event systems : Why you cannot avoid using them. *Discrete Event Dynamic Systems*, 17(3) :355–403, 2007.

[FD95] Yousri El Fattah and Rina Dechter. Proceedings of the fourteenth international joint conference on artificial intelligence, ijcai 95, montréal québec, canada, august 20-25 1995, 2 volumes. In *IJCAI*, pages 1742–1749, 1995.

[FdGJ09] Aurélie Favier, Simon de Givry, and Philippe Jégou. Exploiting problem structure for solution counting. In *Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings*, pages 335–343, 2009.

[Fel06] Alexander Feldman. A multi-valued sat-based algorithm for faster model-based diagnosis. In *DX-06,International Workshop on the Principles of Diagnosis*, 2006.

[Fel10] Alexander Feldman. Solving model-based diagnosis problems with max-sat solvers and vice versa. In *DX- 10, International Workshop on the Principles of Diagnosis*, 2010.

[FFJS04] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Stumptner. Consistency-based diagnosis of configuration knowledge bases. *Artificial Intelligence*, 152(2) :213 – 234, 2004.

[FLP08] Boi Faltings, Thomas Léauté, and Adrian Petcu. Privacy guarantees through distributed constraint satisfaction. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Sydney, NSW, Australia, December 9-12, 2008*, pages 350–358, 2008.

[FM09] Hélène Fargier and Pierre Marquis. Knowledge compilation properties of trees-of-bdds, revisited. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 772–777, 2009.

[FPvG08] Alexander Feldman, Gregory M. Provan, and Arjan J. C. van Gemund. Computing minimal diagnoses by greedy stochastic search. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 911–918, 2008.

[FPvG10] Alexander Feldman, Gregory M. Provan, and Arjan J. C. van Gemund. Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research (JAIR)*, 38 :371–413, 2010.

[Fre82] Eugene C. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29 :24–32, January 1982.

[FvG06] Alexander Feldman and Arjan J. C. van Gemund. A two-step hierarchical algorithm for model-based diagnosis. In *Proceedings, The Twenty-First National Conference*

on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, AAAI, July 16-20, 2006, Boston, Massachusetts, USA, pages 827–833, 2006.

[GA08] Alban Grastien and Anbulagan. Incremental diagnosis of DES by satisfiability. In *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings*, pages 787–788, 2008.

[Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1984.

[GARK07] Alban Grastien, Anbulagan, Jussi Rintanen, and Elena Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 305–310, 2007.

[GG02] Carlos Guestrin and Geoffrey J. Gordon. Distributed planning in hierarchical factored mdps. In *UAI '02, Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada, August 1-4, 2002*, pages 197–206, 2002.

[GLS99] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '99, pages 21–32, New York, NY, USA, 1999. ACM.

[GLS00] Georg Gottlob, Nicola Leone, and Francesco Scarcello. A comparison of structural csp decomposition methods. *Artificial Intelligence*, 124(2) :243–282, 2000.

[Gol80] Martin Charles. Golumbic. *Algorithmic graph theory and perfect graphs / Martin Charles Golumbic*. Academic Press, New York :, 1980.

[GPBT06] Rachel Greenstadt, Jonathan P. Pearce, Emma Bowring, and Milind Tambe. Experimental analysis of privacy loss in dcop algorithms. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 1424–1426, 2006.

[GW10] Stéphane Grumbach and Zhilin Wu. Distributed tree decomposition of graphs and applications to verification. In *IPDPS Workshops*, pages 1–8, 2010.

[Ham02] Youssef Hamadi. Optimal distributed arc-consistency. *Constraints*, 7(3-4) :367–385, 2002.

[HD04] Jinbo Huang and Adnan Darwiche. Using dpll for efficient obdd construction. In *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings*, 2004.

[HD05a] Jinbo Huang and Adnan Darwiche. Dpll with a trace : From sat to knowledge compilation. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 156–162, 2005.

[HD05b] Jinbo Huang and Adnan Darwiche. On compiling system models for faster and more scalable diagnosis. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 300–306, 2005.

[HD07] Jinbo Huang and Adnan Darwiche. The language of search. *Journal of Artificial Intelligence Research (JAIR)*, 29 :191–219, 2007.

[HIST05] Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarinov. Schema mediation for large-scale semantic data sharing. *VLDB Journal*, 14(1) :68–83, 2005.

[Ino92] Katsumi Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 56 :301–353, August 1992.

[JG08] Priscilla Kan John and Alban Grastien. Local consistency and junction tree for diagnosis of discrete-event systems. In *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings*, pages 209–213, 2008.

[JJ94] Finn Verner Jensen and Frank Jensen. Optimal junction trees. In *UAI '94 : Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence, Seattle, Washington, USA, July 29-31*, pages 360–366, 1994.

[JNT05] Philippe Jégou, Samba Ndiaye, and Cyril Terrioux. Computing and exploiting tree-decompositions for solving constraint networks. In *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, pages 777–781, 2005.

[JNT07] Philippe Jégou, Samba Ndiaye, and Cyril Terrioux. Dynamic heuristics for backtrack search on tree-decomposition of csps. In *IJCAI*, pages 112–117, 2007.

[KDLD05] Kalev Kask, Rina Dechter, Javier Larrosa, and Avi Dechter. Unifying tree decompositions for reasoning in graphical models. *Artificial Intelligence*, 166(1-2) :165–193, 2005.

[KK03] Meir Kalech and Gal A. Kaminka. On the design of social diagnosis algorithms for multi-agent teams. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 370–375, 2003.

[KK05] Meir Kalech and Gal A. Kaminka. Towards model-based diagnosis of coordination failures. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 102–107, 2005.

[KKME06] Meir Kalech, Gal A. Kaminka, Amnon Meisels, and Yehuda Elmaliach. Diagnosis of multi-robot coordination failures using distributed csp algorithms. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, 2006.

[KKZ02] J. Kurien, X. Koutsoukos, and F. Zhao. Distributed diagnosis of networked, embedded systems. In *DX-02,International Workshop on the Principles of Diagnosis*, 2002.

[KS92] Henry A. Kautz and Bart Selman. Planning as satisfiability. In *ECAI*, pages 359–363, 1992.

[LF11] Thomas Léauté and Boi Faltings. Coordinating logistics operations with privacy guarantees. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2482–2487, 2011.

[lyd] lydia. Language for system diagnosis and it is a modeling language and a reasoning tool-kit biased. `http://cette.url.est.ennuyeuse.dom`.

[LYDM09] Yingmin Li, Lina Ye, Philippe Dague, and Tarek Melliti. A decentralized model-based diagnosis for bpel services. In *ICTAI 2009, 21st IEEE International Conference on Tools with Artificial Intelligence, Newark, New Jersey, USA, 2-4 November 2009*, pages 609–616, 2009.

[Mac77] Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1) :99–118, 1977.

[MH96] S. A. M. Makki and George Havas. Distributed algorithms for depth-first search. *Information Processing Letters*, 60(1) :7–12, 1996.

[MJ89] Cindy L. Mason and Rowland R. Johnson. *DATMS : a framework for distributed assumption based reasoning*, pages 293–317. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[Moz91] Igor Mozetic. Hierarchical model-based diagnosis. *International Journal of Man-Machine Studies*, 35(3) :329–362, 1991.

[MTT04] Roberto Micalizio, Pietro Torasso, and Gianluca Torta. On-line monitoring and diagnosis of multi-agent systems : A model based approach. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, pages 848–852, 2004.

[MZ07] Amnon Meisels and Roie Zivan. Asynchronous forward-checking for discsps. *Constraints*, 12(1) :131–150, 2007.

[ND98] T. Nguyen and Yves Deville. A distributed arc-consistency algorithm. *Science of Computer Programming*, 30(1-2) :227–250, 1998.

[New03] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2) :167–256, Jun 2003.

[NG04] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2) :026113, Feb 2004.

[PC05] Yannick Pencolé and Marie-Odile Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164(1-2) :121–170, 2005.

[PF05] Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 266–271, 2005.

[Pro01] Gregory Provan. Hierarchical model-based diagnosis. In *DX-01, International Workshop on the Principles of Diagnosis*, 2001.

[Pro02] Gregory M. Provan. A model-based diagnosis framework for distributed embedded systems. In *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002*, pages 341–352, 2002.

[PW07] Gregory M. Provan and Jun Wang. Automated benchmark model generators for model-based diagnostic inference. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 513–518, 2007.

[RB62] Lester Randolph Ford Richard Bellman. Flows in networks. *Princeton University Press*, 1962.

[Rei87] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1) :57–95, 1987.

[RPBD12] Philippe Rinaudo, Yann Ponty, Dominique Barth, and Alain Denise. Tree decomposition and parameterized algorithms for rna structure-sequence alignment including tertiary interactions and pseudoknots. *CoRR*, abs/1206.3789, 2012.

[RS86] Neil Robertson and P. D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3) :309 – 322, 1986.

[RtTBW02] Nico Roos, Annette ten Teije, André Bos, and Cees Witteveen. An analysis of multi-agent diagnosis. In *The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy, Proceedings*, pages 986–987, 2002.

[RtTW03] N. Roos, A. ten Teije, and C. Witteveen. A protocol for multi agent diagnosis with spatially distributed knowledge. *AAMAS 2003*, 2003.

[SBH07] Sathiamoorthy Subbarayan, Lucas Bordeaux, and Youssef Hamadi. Knowledge compilation properties of tree-of-bdds. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 502–507, 2007.

[SdV01] Laurent Simon and Alvaro del Val. Efficient consequence finding. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 359–370, 2001.

[SF02] Marius-Calin Silaghi and Boi Faltings. Self reordering for security in generalized english auctions (gea). In *The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy, Proceedings*, pages 459–460, 2002.

[SMI89] M. B. Sharma, N. K. Mandyam, and S. S. Iyangar. An optimal distributed depth-firstssearch algorithm. In *ACM Conference on Computer Science*, pages 287–294, 1989.

[SNL06] Benno Stein, Oliver Niggemann, and Theodor Lettmann. Speeding up model-based diagnosis by a heuristic approach to solving sat. In *Proceedings of the 24th IASTED international conference on Artificial intelligence and applications*, pages 273–278, Anaheim, CA, USA, 2006. ACTA Press.

[SP99] A. K. Shiny and Arun K. Pujari. An efficient algorithm to generate prime implicants. *Journal of Automatic Reasoning*, 22 :149–170, February 1999.

[SSL+96] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D.C. Teneketzis. Failure diagnosis using discrete-event models. *Control Systems Technology, IEEE Transactions on*, 4(2) :105 –124, March 1996.

[ST05] Luciano Serafini and Andrei Tamilin. Drago : Distributed reasoning architecture for the semantic web. In *The Semantic Web : Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings*, pages 361–376, 2005.

[SVAV05] Alexander Smith, Andreas G. Veneris, Moayad Fahim Ali, and Anastasios Viglas. Fault diagnosis and logic debugging using boolean satisfiability. *IEEE Transaction on CAD of Integrated Circuits and Systems*, 24(10) :1606–1621, 2005.

[SW03] Markus Stumptner and Franz Wotawa. Coupling csp decomposition methods and diagnosis algorithms for tree-structured systems. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 388–393, 2003.

[SW05] R. Su and W. Murray Wonham. Global and local consistencies in distributed fault diagnosis for discrete-event systems. *IEEE transactions on automatic control*, 50(12) :1923–1935, 2005.

[SW06] R. Su and W. Murray Wonham. Hierarchical fault diagnosis for discrete-event systems under global consistency. *Discrete Event Dynamic Systems*, 16(1) :39–70, 2006.

[SWP88] Barbara Smith, Ralph Wilkerson, and Gerald E. Peterson. Automated circuit diagnosis using first order logic tools. In *Proceedings of the 1st international conference on Industrial and engineering applications of artificial intelligence and expert systems - Volume 1*, IEA/AIE '88, pages 456–465, New York, NY, USA, 1988. ACM.

[TT02] Gianluca Torta and Pietro Torasso. The role of obdds in controlling the complexity of model based diagnosis. In *DX-02,International Workshop on the Principles of Diagnosis*, 2002.

[TT07] Gianluca Torta and Pietro Torasso. An on-line approach to the computation and presentation of preferred diagnoses for dynamic systems. *AI Commununications*, 20(2) :93–116, 2007.

[Wal99] Toby Walsh. Search in a small world. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 1172–1177, 1999.

[Wot11] Franz Wotawa. On the use of abduction as an alternative to decision trees in environmental decision support systems. *IJAEIS*, 2(1) :63–82, 2011.

[WP09] Jun Wang and Gregory M. Provan. Characterizing the structural complexity of real-world complex networks. In *Complex (1)*, pages 1178–1189, 2009.

[WR03] Brian C. Williams and Robert J. Ragno. Conflict-directed A* and its role in model-based embedded systems. *Journal of Discrete Applied Mathematics*, 2003.

[YDIK98] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem : Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5) :673–685, 1998.

[YSH02] Makoto Yokoo, Koutarou Suzuki, and Katsutoshi Hirayama. Secure distributed constraint satisfaction : Reaching agreement without revealing private information. In *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings*, pages 387–401, 2002.

[Zim95] Philip Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.