Synthèse d'un diagnostiqueur distribué et précis

Priscilla Kan John[1] Alban Grastien[1] Yannick Pencolé[2] Pauline Ribot[2]
[1] Australian National University and NICTA*
[2] CNRS ; LAAS ; 7 avenue du Colonel Roche, F-31077 Toulouse, France
[2] Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse
[1](priscilla.kanjohn@anu.edu.au, alban.grastien@nicta.com.au) [2](ypencole@laas.fr, pribot@laas.fr)

Résumé

Le diagnostic de grands systèmes à événements discrets distribués est confronté au problème d'explosion combinatoire. Une mise en œuvre de l'algorithme de diagnostic (diagnostiqueur) utilise un arbre de jonction sur la topologie. L'efficacité de cette méthode dépendant directement du degré d'intrication de la topologie, nous proposons d'adapter l'approche en ignorant certaines connexions. Cependant, ce faisant, on met en péril la précision du diagnostic. Aussi, nous effectuons au préalable une analyse pour déterminer quelles connexions peuvent être ignorées sans compromettre la précision du diagnostiqueur.

Mots Clef

Diagnostic, Système à événements discrets, Arbre de jonction, Précision

Abstract

Diagnosis of large discrete event systems leads to combinatorial explosion. We propose to solve this issue by ignoring some connections on the system and using a junction tree-based approach. Removing connections reduces both the cost and the accuracy of the diagnosis. Therefore, we perform an off-line analysis to determine which connections can be safely ignored.

Keywords

Diagnosis, Discrete event system, Junction tree, Accuracy

1 Introduction

La surveillance de systèmes de grande taille et fortement décentralisés, tels les réseaux de télécommunication, les *web-services* ou les réseaux de distribution d'électricité, est une tâche particulièrement difficile. Une panne peut conduire à une cascade d'événements et d'alarmes qui sont à la fois difficiles à interpréter et potentiellement dangereux pour le système. Les systèmes qui nous intéressent ici sont souvent modélisés par un *système à événements discrets* [2] qui permet de représenter simplement leurs comportements. Nous considérons que le diagnostic est basé sur un tel modèle. En pratique, les observations sur le système sont corrélées avec le modèle pour déterminer si des comportements non souhaitables (pannes) ont eu lieu.

La complexité du raisonnement sur un modèle augmente de manière exponentielle par rapport au nombre de composants. Cela signifie qu'il est impossible d'employer les techniques les plus naïves dès lors que le système comporte plusieurs dizaines de composants. Or, nous considérons ici des systèmes de plusieurs centaines, voire plusieurs milliers de composants. Différentes techniques ont été développées pour combattre cette difficulté : précompilation [9, 4], méthodes symboliques [1, 10, 5], algorithmes décentralisés/distribués [7, 6, 11]. Bien que de grands progrès aient été accomplis, ces techniques se révèlent parfois insuffisantes.

Les composants du système sont physiquement ou logiquement reliés par des *connexions* qui restreignent leur comportements individuels et induisent un comportement collectif non trivial. En conséquence, pour un système à faible connectivité, il est possible de raisonner localement sur de petits sous-systèmes, tandis qu'un système à forte connectivité nécessite d'entretenir une vision globale – donc complexe – du système.

L'idée développée dans cet article est d'ignorer certaines connexions du système. Il est alors possible d'effectuer des raisonnements sur des petits sous-systèmes, et le diagnostic peut être fourni en temps raisonnable. En contrepartie, l'approche conduit à une *perte de précision* du diagnostic. En effet, en ignorant certaines connexions, on omet des informations permettant potentiellement de supprimer certaines hypothèses de diagnostic. Nous effectuons donc au préalable une *analyse de précision* sur le modèle pour déterminer quelles connexions peuvent être ignorées sans nuire à la précision globale du diagnostiqueur.

L'algorithme de diagnostic lui-même est implémenté par une décomposition du réseau en arbre de jonction [6]. La complexité du diagnostic peut alors se déduire de l'arbre de décomposition du système simplifié.

Le reste de l'article est divisé comme suit. Les notations sont présentées dans la section suivante. Le diagnostic est ensuite défini et les approches distribuées sont présentées. Le diagnostic sur une sous-configuration est défini et finalement la détermination de la sous-configuration optimale est discutée.

*NICTA est subventionné par le Gouvernement Australien représenté par le *Department of Broadband, Communications and the Digital Economy* et l'*Australian Research Council* au travers du programme *ICT Centre of Excellence*.

2 Pré-requis

Nous nous plaçons dans le cadre des systèmes à événements discrets (SED) [2] et utilisons la notation des langages pour les représenter. Le formalisme servant à définir les SED et leur problème de diagnostic est présenté de façon générale dans un premier temps.

On note Σ un ensemble de *symboles* (ces symboles représenteront par la suite les événements sur le système). Un *mot* σ sur Σ est une séquence finie $\sigma = s_1.\cdots.s_n$ telle que pour tout i, $s_i \subseteq \Sigma$. Ainsi, si $\Sigma = \{a, b, c, d\}$, alors $\{a\}.\{b, c\}.\{d\}.\{a\}$ est un mot sur Σ où b et c apparaissent simultanément.[1] Le mot vide est noté ε. On note abusivement $s_i \in \sigma$ pour signifier que l'ensemble $s_i \subseteq \Sigma$ est dans la séquence σ. $(2^\Sigma)^\star$ est l'ensemble des mots sur Σ. Un *langage* $\mathcal{L}$ sur Σ est un sous-ensemble de mots $\mathcal{L} \subseteq (2^\Sigma)^\star$.

Définition 1 (Projection) *La* projection *sur Σ' d'un mot σ sur $\Sigma \supseteq \Sigma'$, notée $\mathrm{P}_{\Sigma\to\Sigma'}(\sigma)$, ou plus simplement $\mathrm{P}_{\Sigma'}(\sigma)$, est le mot sur Σ' qui ne conserve que les symboles de Σ' et retire les ensembles de symboles vides. Formellement* $\mathrm{P}_{\Sigma\to\Sigma'}(\sigma) =$

$$\begin{cases} \varepsilon & \text{si } \sigma = \varepsilon, \\ \mathrm{P}_{\Sigma\to\Sigma'}(\sigma') & \text{si } (\sigma = s.\sigma') \wedge (s \cap \Sigma' = \emptyset), \\ (s \cap \Sigma').\mathrm{P}_{\Sigma\to\Sigma'}(\sigma') & \text{si } (\sigma = s.\sigma') \wedge (s \cap \Sigma' \neq \emptyset). \end{cases}$$

La projection sur Σ' du langage $\mathcal{L}$ sur $\Sigma \supseteq \Sigma'$, notée $\mathrm{P}_{\Sigma\to\Sigma'}(\mathcal{L})$ est l'ensemble de ses mots projetés sur Σ' : $\mathrm{P}_{\Sigma\to\Sigma'}(\mathcal{L}) = \{\mathrm{P}_{\Sigma\to\Sigma'}(\sigma) \mid \sigma \in \mathcal{L}\}$. L'opération inverse de projection renvoie l'ensemble des mots sur Σ dont la projection sur Σ' est incluse dans le langage d'origine : $\mathrm{P}^{-1}_{\Sigma\to\Sigma'}(\mathcal{L}) = \{\sigma \in (2^\Sigma)^\star \mid \mathrm{P}_{\Sigma\to\Sigma'}(\sigma) \in \mathcal{L}\}$.

Synchronisation

Les langages peuvent être utilisés localement pour modéliser le comportement d'une entité. Lorsqu'on souhaite raisonner sur un ensemble d'entités, il est alors nécessaire de *synchroniser* ces langages pour générer le langage global. Chaque langage local a un ensemble propre de symboles, disjoint des autres ensembles locaux. Cependant, certains symboles locaux sont liés à différentes manifestations de la même réalité physique. L'opération de synchronisation coordonne ces symboles équivalents en forçant leur simultanéité.

Les symboles équivalents sur différents langages sont représentés sous forme d'ensemble synchrone.

Définition 2 (Ensemble synchrone) *Étant donnés deux ensembles disjoints de symboles Σ_1 et Σ_2, un* ensemble synchrone *$\mathcal{S}$ est un ensemble de paires de symboles provenant des deux ensembles : $\mathcal{S} \subseteq \Sigma_1 \times \Sigma_2$.*

Un élément $\langle a, b\rangle \in \mathcal{S}$ indique que a et b modélisent la même réalité physique, et que la simultanéité de leur présence doit être assurée.

[1] Généralement un mot est défini plus simplement comme une séquence de symboles ; notre notation permet de représenter l'occurrence simultanée d'événements et de satisfaire la propriété d'inclusion de langage sur une sous-topologie (section 5).

Définition 3 (Synchronisation de langages) *Soient deux langages $\mathcal{L}_1$ sur Σ_1 et $\mathcal{L}_2$ sur Σ_2 et $\mathcal{S}$ un ensemble synchrone défini sur Σ_1 et Σ_2. Le* produit synchrone *de $\mathcal{L}_1$ et $\mathcal{L}_2$ sur $\mathcal{S}$, noté $\mathcal{L}_1 \bigotimes_{\mathcal{S}} \mathcal{L}_2$, est défini comme l'ensemble des mots sur $\Sigma = (\Sigma_1 \cup \Sigma_2)$ dont la projection sur chaque ensemble de symboles local est incluse dans le langage local, et qui satisfont la contrainte de simultanéité introduite par $\mathcal{S}$. Formellement :* $\{\sigma \in (2^\Sigma)^\star \mid \big(\forall i \in \{1, 2\},\ \mathrm{P}_{\Sigma\to\Sigma_i}(\sigma) \in \mathcal{L}_i\big) \wedge \big(\forall\langle a, b\rangle \in \mathcal{S},\ \forall s \in \sigma,\ a \in s \Leftrightarrow b \in s\big)\}$.

À condition de définir correctement les ensembles synchrones, il est possible de prouver que ces notations conservent les propriétés de commutativité et d'associativité des notations plus traditionnelles. Quand l'ensemble $\mathcal{S}$ est implicite, il est omis dans les notations : $\mathcal{L}_1 \otimes \mathcal{L}_2$.

Il convient d'introduire une dernière notion ici. Le langage $\mathcal{L}_1$ devant être synchronisé avec le langage $\mathcal{L}_2$ est dit *localement cohérent* avec celui-ci si il contient uniquement des mots synchronisables avec au moins un mot de $\mathcal{L}_2$. L'opération de cohérence calcule le sous-langage cohérent maximum $\mathcal{L}'_1 \subseteq \mathcal{L}_1$ comme suit $\mathcal{L}'_1 = \mathrm{P}_{\Sigma_1}(\mathcal{L}_1 \otimes \mathcal{L}_2)$.

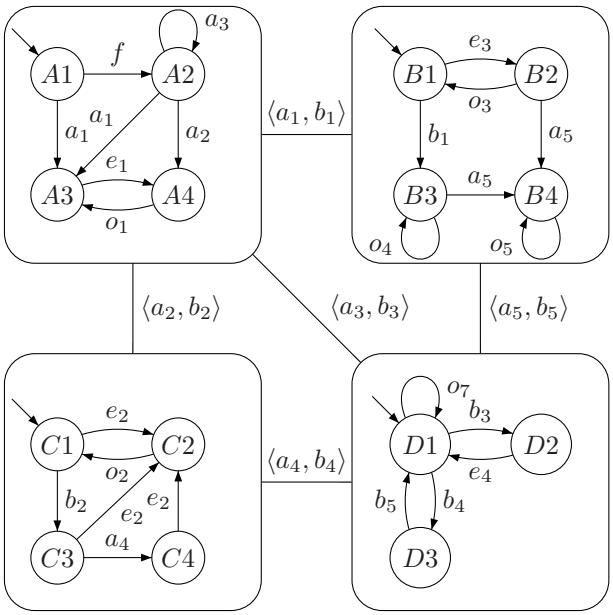

FIG. 1 – Exemple de système

La figure 1 donne un exemple utilisé dans la suite de l'article, avec quatre langages représentés sous forme d'automate et leurs ensembles synchrones. La synchronisation des automates B et D est donnée figure 2 (les transitions comportant deux événements non synchrones ne sont pas représentées). Les deux transitions synchrones relient les états 33 et 23 à l'état 41 (le premier chiffre de chaque état se réfère à B et le second à D). On voit que l'événement b_5 ne peut avoir lieu qu'une fois sur la synchronisation ; ainsi, l'opération de cohérence locale devrait ne conserver que les traces ne comportent qu'une occurrence de b_5.

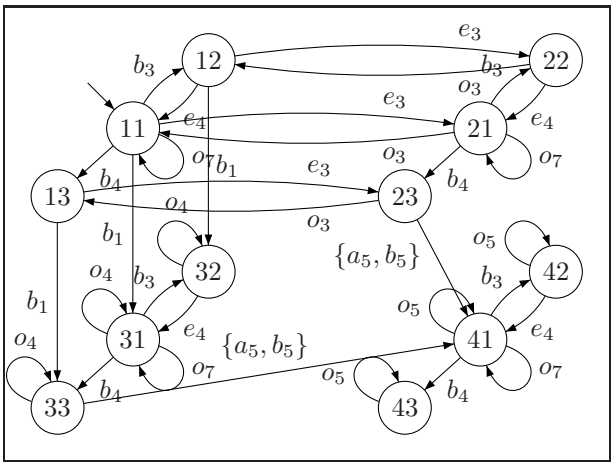

FIG. 2 – Synchronisation de B et D

3 Diagnostic de fautes dans les SED

Le diagnostic d'un système Γ établit si le comportement du système est normal ou défectueux en prenant en compte des observations faites sur le système. Notre travail se situe dans le cadre du diagnostic à base de modèle.

Modèle Nous supposons qu'il existe un modèle complet du système Γ. Le comportement du système est capturé par un langage Mod défini sur un ensemble fini d'événements Σ qui peuvent avoir lieu sur le système ($Mod \subseteq (2^{\Sigma})^{\star}$). En considérant Σ_f comme étant l'ensemble des événements de faute du système, nous avons $\Sigma_f \subseteq \Sigma$.
Le système contient des événements particuliers $\Sigma_o \subseteq \Sigma$ appelés *événements observables*. L'occurrence d'un événement observable génère l'émission d'une *observation*. Les observations sont représentées sous la forme d'un langage noté Obs. Pour simplifier cet article, on considère que les observations ne sont pas soumises au bruit et que Obs contient donc un seul mot.

Diagnostiqueur De façon générique et indépendamment d'un quelconque formalisme sous-jacent, un *diagnostiqueur* est un processus (ou encore une fonction) qui exécute un raisonnement fondé sur des observations dans le but d'évaluer si des fautes se sont produites. En particulier, un *F-diagnostiqueur* Δ^F d'un événement de faute $F \in \Sigma_F$ est un processus particulier de diagnostic qui renvoie trois résultats possibles [8] :

1. F-certain : l'occurrence de F est certaine ;
2. F-sain : la non-occurrence de F est certaine ;
3. F-ambigu : l'occurrence de F est incertaine.

Par définition, la faute F est considérée comme *permanente*. En adoptant cette vision, le processus global de diagnostic sur un système peut ainsi être constitué d'un ensemble de diagnostiqueurs spécialisés qui sont associés à l'ensemble des fautes anticipées dans la modélisation du système surveillé. Dans la suite de cet article, l'objectif est de déterminer un F-diagnostiqueur Δ^F sur une faute F du système Γ sachant que pour les autres la démarche est évidemment identique. Sauf cas particulier et pour des raisons de lisibilité, Δ^F sera simplement noté Δ.

Langage explicatif Le *langage explicatif* est le langage représentant les comportements du système qui *expliquent* les observations. Il se calcule comme suit :

$$\mathit{Expl} = Mod \otimes Obs. \tag{1}$$

Le langage *Expl* se calcule comme le produit synchrone entre Mod et Obs. *Expl* contient les comportements du système qui sont cohérents avec les observations sur le système.
À l'aide de *Expl*, il est facile de définir le F-diagnostiqueur global Δ_Γ qui, à partir de la connaissance globale Mod et de l'ensemble des observations Obs, fournit le meilleur résultat possible, à savoir : $\Delta_\Gamma(Obs) =$

$$\begin{cases} \text{F-certain} & \text{si } \forall \sigma \in \mathit{Expl},\ \exists s \in \sigma : F \in s \\ \text{F-sain} & \text{si } \forall \sigma \in \mathit{Expl},\ \forall s \in \sigma : F \notin s \\ \text{F-ambigu} & \text{sinon.} \end{cases} \tag{2}$$

Quelles que soient les représentations des langages Mod, Obs et *Expl* (par exemple sous forme d'automate comme dans notre exemple), le diagnostic est souvent confronté au problème d'explosion de l'espace de recherche. La taille des structures est exponentielle dans le nombre de composants, ce qui rend difficile leur utilisation dans la pratique et la mise en œuvre directe d'un diagnostiqueur global tel que Δ_Γ. Pour contourner ce problème, des techniques distribuées ont été proposées.

4 Approche Distribuée

Les systèmes techniques modernes sont constitués habituellement de composants qui sont chacun un système en eux-mêmes, avec des comportements simples mais interagissant pour produire en somme un comportement complexe. On appelle le système dans son ensemble un système distribué et on modélise chacun de ses composants séparément. On s'intéresse à un système distribué Γ constitué d'un ensemble de composants : $\Gamma = \{\Gamma_1 \dots \Gamma_n\}$. Chaque composant Γ_i peut être décrit par le langage Mod_i défini sur l'alphabet Σ_i. Une hypothèse implicite est la propriété d'équité sur le système qui implique qu'un composant ne peut devenir infiniment silencieux : sur un comportement infini, le nombre d'observations générées par un composant donné est toujours infini. Les événements de faute sont des événements propres à la physique des composants du système qui ont pour effet de générer des défaillances sur le composant lui-même mais aussi de les propager dans l'ensemble du système. L'occurrence d'une faute de type F est donc considérée comme un événement ne pouvant avoir lieu que sur un composant $\Gamma_i : F \in \Sigma_i \wedge (i \neq j \Rightarrow F \notin \Sigma_j)$. Les composants du système communiquent à l'aide de *connexions*.

Définition 4 *(Connexion) Une* connexion $\mathcal{K}_{ij}$ *existe entre deux composants* Γ_i *et* Γ_j *s'ils ont un lien physique*

ou logique entre eux qui contraint leurs comportements. Un ensemble synchrone $\mathcal{S}_{ij}$ peut servir de modèle abstrait pour une connexion $\mathcal{K}_{ij}$ où $\mathcal{S}_{ij} \subseteq \Sigma_i \times \Sigma_j$ et $\mathcal{S}_{ij} = \mathcal{S}_{ji}$.

La manière dont les composants d'un système distribué sont connectés dicte la *topologie* globale du système. Le modèle global du système est défini implicitement à l'aide de ces ensembles synchrones $\mathcal{S}_{ij}$ par $Mod = Mod_1 \otimes \ldots \otimes Mod_n$ mais il est inutile de le calculer explicitement. Il est possible de modéliser les observations du système de manière décentralisées : $Obs = Obs_1 \otimes \cdots \otimes Obs_n$ [3].
Le langage explicatif $Expl_i$ sur un composant Γ_i est donné par $Mod_i \otimes Obs_i$. Le langage explicatif global *Expl* est calculé par le produit synchrone des langages locaux :

$$\begin{aligned} Expl &= (Mod_1 \otimes \ldots \otimes Mod_n) \otimes (Obs_1 \otimes \ldots \otimes Obs_n) \\ &= (Mod_1 \otimes Obs_1) \otimes \ldots \otimes (Mod_n \otimes Obs_n) \\ &= Expl_1 \otimes \ldots \otimes Expl_n. \end{aligned}$$

Calculer le langage *Expl* en synchronisant tous les composants est souvent impossible si le système est constitué d'un très grand nombre de composants. Les méthodes de diagnostic dites *distribuées* permettent d'éviter ce calcul. Nous utilisons une implémentation à base d'arbre de jonction [6].

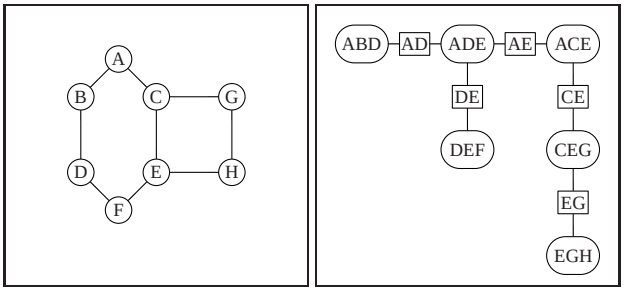

FIG. 3 – Système (gauche) et un arbre de jonction (droite)

Soit un graphe $\mathcal{G} = \langle \Gamma, \mathcal{K} \rangle$ sur les composants du système Γ où $\mathcal{K}$ est l'ensemble de toutes les connexions sur le système. Un arbre de jonction sur $\mathcal{G}$ est une paire $(\mathcal{J}, \mathcal{C})$ où $\mathcal{J}$ est un arbre et $\mathcal{C}$ est une fonction qui lie chaque nœud $\mathcal{N}$ de $\mathcal{J}$ à une *grappe* de composants $\mathcal{C}_i$ (voir exemple figure 3). D'autre part, pour chaque connexion $\mathcal{K}_{ij}$, il doit exister une grappe qui comporte les nœuds : $\{i, j\} \subseteq \mathcal{C}(\mathcal{N})$. Enfin, si deux grappes de l'arbre comportent le même nœud, toutes les grappes entre ces deux grappes doivent également comporter ce nœud (voir le nœud D entre les grappes ABD et DEF).
Pour calculer le diagnostic d'un système distribué, il est suffisant de calculer le langage explicatif local de chaque grappe, et d'effectuer les opérations de cohérence globale depuis les feuilles de l'arbre jusqu'à la racine. Si la racine est choisie pour contenir le composant sur lequel l'événement intervient, le diagnostic global de la faute est obtenu par (2) sur le langage explicatif local de la racine.
Cette méthode permet d'éviter le calcul de *Expl*. En revanche, il est nécessaire de calculer le langage explicatif pour chaque grappe, et la complexité de la représentation de ce langage croît exponentiellement avec le nombre de composants de la grappe. La *largeur arborescente* de la topologie est la taille de la plus grosse grappe de son arbre de jonction moins un. Cette valeur permet de déterminer *a priori* le coût algorithmique du diagnostic par cette méthode.

5 Sous-configuration et précision

L'efficacité de l'algorithme de diagnostic par arbre de jonction dépend directement des connexions entre composants dans le système. Nous proposons donc de relâcher certaines de ces connexions dans le but de générer un arbre sur lequel le raisonnement peut se faire de manière plus efficace.

5.1 Relâchement de connexions

Le relâchement de connexions est formalisé avec les notions de sous-topologie et de sous-configuration.

Définition 5 (Sous-topologie) *Une* sous-topologie *sur un système distribué Γ est un sous-ensemble de connexions $\mathcal{Y} \subseteq \mathcal{K}$.*

Une sous-topologie $\mathbb{T}$ définit un langage $\mathcal{L}(\mathbb{T})$ qui correspond à la synchronisation des langages locaux sur les connexions de l'ensemble $\mathcal{Y}$.
Nous illustrons ce fait sur l'exemple de la figure 1. Ce système comporte quatre composants A à D et cinq connexions $\langle a_i, b_i \rangle$. Un mot possible de ce système est $\{f\}.\{e_3\}.\{a_2, b_2\}.\{a_4, b_4\}.\{a_5, b_5\}.\{o_5\}$. Considérons à présent la sous-topologie où la connexion $\langle a_2, b_2 \rangle$ est ignorée. Le langage de cette sous-topologie comprend des mots supplémentaires, incluant : $\{e_3\}.\{b_2\}.\{a_4, b_4\}.\{a_5, b_5\}.\{o_5\}$. De manière générale, les mots de la sous-topologie $\mathbb{T} \subseteq \mathbb{T}'$ doivent satisfaire moins de contraintes que ceux de $\mathbb{T}'$. En conséquence, $\mathcal{L}(\mathbb{T}') \subseteq \mathcal{L}(\mathbb{T})$, que $\mathcal{L}$ représente le modèle ou le langage explicatif.
On peut calculer le diagnostic du système en utilisant le modèle d'une sous-topologie à la place du modèle du système. La largeur arborescente de la sous-topologie étant plus faible que celle du système, le coût associé au diagnostic par arbre de jonction est plus faible. Il reste cependant à déterminer si le diagnostic est correct.
Puisque le modèle du système est inclus dans le modèle de la sous-topologie, si le diagnostic de la sous-topologie renvoie $\mathrm{F\text{-}certain}$ ou $\mathrm{F\text{-}sain}$, alors le diagnostiqueur global renvoie le même résultat. En revanche, le diagnostic peut devenir ambigu.
Ainsi, reprenons l'exemple de la figure 1. L'observation de o_5 permet de diagnostiquer sans ambiguïté l'occurrence de la faute f (o_5 implique successivement l'occurrence de a_5, b_5, b_4, a_4, b_2, a_2 et finalement f). En revanche, nous avons montré plus haut qu'ignorer la connexion $\langle a_2, b_2 \rangle$ fait apparaître un nouveau mot finissant par o_5 et ne comportant pas la faute f. Le diagnostic basé sur une sous-topologie

ignorant la connexion $\langle a_2, b_2\rangle$ est donc *imprécis* (voir section suivante).
En pratique, une sous-topologie peut isoler des composants du système. Dans ce cas, il est inutile de surveiller les observations de ces composants, puisque le modèle indique qu'ils fonctionnent indépendamment des autres composants. Aussi, nous définissons la notion de sous-configuration.

Définition 6 (Sous-configuration) *Une* sous-configuration $\mathbb{C}$ *est un tuple* $(\{\Gamma_{p_1}, \ldots, \Gamma_{p_m}\}, \mathcal{Y}_\mathbb{C}, \overline{\mathcal{Y}_\mathbb{C}})$ *où* $\{\Gamma_{p_1}, \ldots, \Gamma_{p_m}\}$ *est un ensemble de composants,* $\mathcal{Y}_\mathbb{C}$ *est un ensemble de connexions entre les composants de* $\mathbb{C}$*, et* $\overline{\mathcal{Y}_\mathbb{C}}$ *est l'ensemble des connexions sur les composants de* $\mathbb{C}$ *omises dans* $\mathcal{Y}_\mathbb{C}$.

Une sous-configuration particulière est l'*ensemble connecté* d'une sous-topologie ; elle est définie par l'ensemble des composants directement ou indirectement connectés au composant sur lequel la faute peut avoir lieu.

5.2 Diagnostiqueur précis sur $\mathbb{C}$

Afin de préserver la qualité du diagnostic, il est donc nécessaire de déterminer une sous-configuration sur laquelle le diagnostiqueur $\Delta_\mathbb{C}$ est en mesure d'obtenir un diagnostic aussi *précis* que le diagnostiqueur global Δ_Γ. La précision du diagnostiqueur $\Delta_\mathbb{C}$ est formellement définie de la façon suivante.

Définition 7 (Précision) *Le diagnostiqueur* $\Delta_\mathbb{C}$ *est* précis *si pour toute séquence observable* σ *issue du système telle que* $\Delta_\Gamma(\sigma) = \text{F-certain}$*, pour toute continuation observable* σ' *du système, il existe* $n \in \mathbb{N}$ *telle que* $|\sigma'| \geq n$*,* $\Delta_\mathbb{C}(\mathrm{P}_\mathbb{C}(\sigma.\sigma')) = \text{F-certain}$.

Le résultat suivant suit trivialement de cette définition.

Propriété 1 *Le diagnostiqueur global* Δ_Γ *est précis.*

L'intérêt d'un diagnostiqueur précis réside dans sa capacité à obtenir le même résultat que le diagnostiqueur global du système si la faute F a effectivement eu lieu avec un délai éventuel mais fini (propriété d'observabilité équitable).
En effet, si la faute F a eu lieu, alors, pour une séquence d'observations donnée quelconque σ le diagnostiqueur global a deux possibilités de réponses, soit $\Delta_\Gamma(\sigma) = \text{F-ambigu}$, soit $\Delta_\Gamma(\sigma) = \text{F-certain}$. Du fait que l'observabilité du système est équitable, on sait qu'après un nombre fini d'observations du système, une observation va être produite par l'un des composants de $\mathbb{C}$. Soit $\sigma''.o$ cette séquence finie telle que o soit produit par $\mathbb{C}$. Si $\Delta_\Gamma(\sigma) = \text{F-ambigu}$, deux cas sont désormais possibles :

1. soit l'ambiguïté est toujours présente $\Delta_\Gamma(\sigma\sigma''.o) = \text{F-ambigu}$,
2. soit elle ne l'est plus $\Delta_\Gamma(\sigma\sigma''.o) = \text{F-certain}$.

Si l'ambiguïté est toujours présente, on a par construction $\Delta_\mathbb{C}(\mathrm{P}_\mathbb{C}(\sigma\sigma''.o)) = \text{F-ambigu} = \Delta_\Gamma(\sigma\sigma''.o)$. Si l'ambiguïté n'est plus présente, soit on a déjà $\Delta_\mathbb{C}(\mathrm{P}_\mathbb{C}(\sigma\sigma''.o)) = \text{F-certain}$, soit en attendant un nombre fini n d'observations σ', on aura $\Delta_\mathbb{C}(\mathrm{P}_\mathbb{C}(\sigma\sigma''.o\sigma')) = \text{F-certain}$ et donc au final un résultat identique mais obtenu uniquement en observant $\mathbb{C}$.

5.3 Caractérisation d'un diagnostiqueur précis

Pour garantir que $\Delta_\mathbb{C}$ soit précis, il suffit donc d'analyser *a priori* si la sous-configuration $\mathbb{C}$ rassemble des connaissances suffisantes. Dans la suite, la sous-configuration analysée est notée $\mathbb{C} = \{\{\Gamma_{p_1}, \ldots, \Gamma_{p_m}\}, \mathcal{Y}_\mathbb{C}, \overline{\mathcal{Y}_\mathbb{C}}\}$. Pour que l'analyse ait un sens, il faut que la faute F se produise sur l'un des composants $\Gamma_F = \Gamma_{p_i}$ de $\mathbb{C}$. On considère également que $\mathbb{C}$ est un ensemble connecté comme présenté en 5.1. On note $\mathbb{C}_{\max} = \{\{\Gamma_{p_1}, \ldots, \Gamma_{p_m}\}, \mathcal{Y}_\mathbb{C} \cup \overline{\mathcal{Y}_\mathbb{C}}, \emptyset\}$ la sous-configuration associée à $\mathbb{C}$ dans laquelle aucune connexion n'est relâchée. $\mathbb{C}_{\max}$ prend donc en compte toutes les connexions du système impliquant les composants $\{\Gamma_{p_1}, \ldots, \Gamma_{p_m}\}$. Le langage d'événements généré par la sous-configuration $\mathbb{C}$ (resp. $\mathbb{C}_{\max}$) est noté $\mathcal{L}_\mathbb{C}$ (resp. $\mathcal{L}_{\mathbb{C}_{\max}}$). Par définition, $\mathcal{L}_{\mathbb{C}_{\max}} \subseteq \mathcal{L}_\mathbb{C}$. Pour des raisons de simplicité dans cette section, Σ est restreint à l'ensemble des événements de $\mathbb{C}$ (et donc de $\mathbb{C}_{\max}$). Parmi les événements de Σ on distingue en particulier : l'ensemble Σ_o des événements observables, l'ensemble Σ_r^{ext} des événements interactifs de $\mathbb{C}$ associés à des connexions relâchées externes (c.-à-d. une connexion du système dont seul l'un des composants appartient à $\mathbb{C}$).
Vérifier si une sous-configuration $\mathbb{C}$ est suffisante pour en extraire un diagnostiqueur précis consiste à effectuer une analyse de *traces*.

Définition 8 (Trace) *Soit* $F \in \Sigma$ *une faute et* $\mathbb{C}$ *une sous-configuration, l'ensemble des* traces *de* F *dans* $\mathbb{C}$ *est le langage :*

$$\mathcal{T}(\mathbb{C}, F) = \{\tau = s_1.\cdots.s_m \in \mathcal{L}_\mathbb{C}, \exists s_i \in \tau, F \in s_i\}.$$

De même, le complément de $\mathcal{T}(\mathbb{C}, F)$ dans $\mathcal{L}_\mathbb{C}$ (noté $\mathcal{T}(\mathbb{C}, \neg F)$) constitue l'ensemble des traces où la faute F n'est pas présente.

Définition 9 (Trace observable) *Soit* F *une faute et* $\mathbb{C}$ *une sous-configuration, une* trace observable *de* F *dans* $\mathbb{C}$ *est une séquence d'événements observables du langage :*

$$Obs(\mathbb{C}, F) = \mathrm{P}_{\Sigma \to \Sigma_o}(\mathcal{T}(\mathbb{C}, F)).$$

De même $Obs(\mathbb{C}, \neg F)$ représente l'ensemble des traces observables de $\mathbb{C}$ où F n'est pas présente.

Quelles sont les sources d'imprécision ?
Soit une séquence observable $\sigma.o$ du système se terminant par un événement observable o de $\mathbb{C}$ et telle que le diagnostiqueur global réponde $\Delta_\Gamma(\sigma.o) = \text{F-certain}$, soit $\sigma'.o$ la projection observable de $\sigma.o$ sur $\mathbb{C}$. Le premier constat est que la réponse de $\Delta_\mathbb{C}$ à l'observation de $\sigma'.o$ ne peut être que F-certain ou F-ambigu car $\sigma'.o$

est nécessairement une trace observable de F dans $\mathbb{C}$. Le deuxième constat est que si $\Delta_{\mathbb{C}}(\sigma'.o) = \text{F-certain}$, il n'y a pas de problème de précision. Il reste donc le cas problématique où $\Delta_{\mathbb{C}}(\sigma'.o) = \text{F-ambigu}$ alors que $\Delta_{\Gamma}(\sigma.o) = \text{F-certain}$. Dans ce cas, $\sigma'.o$ est une trace observable ambiguë ($\sigma'.o \in Obs(\mathbb{C}, F) \cap Obs(\mathbb{C}, \neg F)$). La difficulté consiste maintenant à déterminer un critère sur la configuration $\mathbb{C}$ qui garantit que si le diagnostic de $\Delta_{\mathbb{C}}$ est ambigu alors celui de Δ_{Γ} l'est aussi et ceci sans calculer Δ_{Γ}. Dans un premier temps, il faut remarquer que l'ambiguïté issue de $\Delta_{\mathbb{C}}(\sigma'.o)$ a en général deux origines différentes.

1. L'ensemble des composants de $\mathbb{C}$ ne sont pas assez observables localement et seules des observations issues de composants externes à $\mathbb{C}$ supprime l'ambiguïté (ce problème est intrinsèque à $\mathbb{C}_{\max}$).
2. Il y a trop de connexions relâchées dans $\mathbb{C}$. Le diagnostic est ambigu car $\Delta_{\mathbb{C}}$ présuppose l'existence de comportements qui ne sont pas possibles dans $\mathbb{C}_{\max}$.

Critère de détection de la précision de $\mathbb{C}_{\max}$
La séquence $\sigma.o$ représente une séquence observable de Γ donc il existe au moins une trace τ de Γ telle que $Obs(\tau) = \sigma.o$. Soit $\tau_{int} = \mathrm{P}_{\Sigma_r^{ext}}(\tau)$ la trace interactive issue de τ et associée à la configuration $\mathbb{C}_{\max}$, le critère de détection repose alors sur le résultat suivant.

Propriété 2 *S'il existe dans $\mathbb{C}_{\max}$ deux traces τ_F et $\tau_{\neg F}$ telles que :*
– $\mathrm{P}_{\Sigma_r^{ext}}(\tau_F) = \mathrm{P}_{\Sigma_r^{ext}}(\tau_{\neg F}) = \tau_{int}$
– $\mathrm{P}_{\Sigma_o}(\tau_F) = \mathrm{P}_{\Sigma_o}(\tau_{\neg F}) = \sigma'.o$
– $F \in \tau_F \wedge F \notin \tau_{\neg F}$
alors $\Delta_{\Gamma}(\sigma.o) = \text{F-ambigu}$.

Preuve : le résultat est immédiat. Dès lors que τ_F participe à une trace globale pouvant expliquer la séquence observable $\sigma.o$, la trace $\tau_{\neg F}$ participe nécessairement à une autre trace globale expliquant $\sigma.o$ (car elles ont les mêmes projections observables et interactives). Finalement, il existe bien deux traces globales expliquant $\sigma.o$ l'une contenant F et l'autre pas. □
Cette propriété relate le cas favorable où il n'y a pas de problème de précision (à savoir que $\Delta_{\mathbb{C}_{\max}}(\sigma'.o) = \Delta_{\Gamma}(\sigma.o) = \text{F-ambigu}$).

Propriété 3 *Si $\Delta_{\Gamma}(\sigma.o) = \text{F-certain}$ et $\Delta_{\mathbb{C}_{\max}}(\sigma'.o) = \text{F-ambigu}$ alors il existe dans $\mathbb{C}_{\max}$ au moins deux traces τ_F ($F \in \tau_F$) et $\tau_{\neg F}$ ($F \notin \tau_{\neg F}$) telles que :*
– $\mathrm{P}_{\Sigma_o}(\tau_F) = \mathrm{P}_{\Sigma_o}(\tau_{\neg F}) = \sigma'.o$,
– $\forall \tau \in \mathcal{T}(\mathbb{C}_{\max}) | \, \mathrm{P}_{\Sigma_o}(\tau) = \sigma'.o \wedge \mathrm{P}_{\Sigma_r^{ext}}(\tau) = \mathrm{P}_{\Sigma_r^{ext}}(\tau_F) \implies F \in \tau$.

Preuve : La première condition vient du fait que $\Delta_{\mathbb{C}_{\max}}(\sigma'.o) = \text{F-ambigu}$ si et seulement s'il existe au moins deux traces τ_F ($F \in \tau_F$) et $\tau_{\neg F}$ ($F \notin \tau_{\neg F}$) telles que $\mathrm{P}_{\Sigma_o}(\tau_F) = \mathrm{P}_{\Sigma_o}(\tau_{\neg F}) = \sigma'.o$. La deuxième condition est directement issue de la propriété 2 par contrapposé et qui autorise le fait que $\Delta_{\Gamma}(\sigma.o)$ puisse être F-certain. □

Propriété 4 *Pour que $\Delta_{\mathbb{C}_{\max}}$ soit précis, il suffit que l'ensemble des couples $(\tau_F, \tau_{\neg F})$ défini par la propriété 3 soit fini.*

Preuve : Considérons une séquence observable $\sigma.o$ avec o issue de $\mathbb{C}_{\max}$ telle que $\Delta_{\Gamma}(\sigma.o) = \text{F-certain}$ et supposons que $\Delta_{\mathbb{C}_{\max}}(\mathrm{P}_{\Sigma \to \Sigma_o}(\sigma o)) = \text{F-ambigu}$. Si $\Delta_{\mathbb{C}_{\max}}$ n'est pas précis alors il existe au moins une suite infinie de continuations observables $\sigma_1 o_1$, $\sigma_1 o_1 \sigma_2 o_2 \ldots$ avec o_i issue de $\mathbb{C}_{\max}$ telle que $\Delta_{\mathbb{C}_{\max}}(\mathrm{P}_{\Sigma \to \Sigma_o}(\sigma o \sigma_1 o_1)) = \text{F-ambigu}$, $\Delta_{\mathbb{C}_{\max}}(\mathrm{P}_{\Sigma \to \Sigma_o}(\sigma o \sigma_1 o_1 \sigma_2 o_2)) = \text{F-ambigu} \ldots$ d'où la présence d'un ensemble infini de couples $(\tau_F, \tau_{\neg F})$ d'après la propriété 3. □

Critère de détection de la précision de $\mathbb{C}$
La différence entre une configuration $\mathbb{C}$ quelconque et la configuration $\mathbb{C}_{\max}$ associée est le relâchement de connexions internes qui conduit le diagnostiqueur $\Delta_{\mathbb{C}}$ à considérer un ensemble de traces contenant l'ensemble des traces de $\mathbb{C}_{\max}$. La conséquence en terme de précision est la suivante. Dès lors que $\Delta_{\Gamma}(\sigma.o) = \text{F-certain}$, il existe nécessairement une trace $\tau_F \in \mathcal{T}(\mathbb{C}_{\max}, \neg F)$ contenant F et qui participe à l'explication de $\sigma.o$ comme il l'a été décrit précédemment. Maintenant, au niveau de la configuration $\mathbb{C}$, le diagnostiqueur $\Delta_{\mathbb{C}}$ répond non seulement en fonction de la présence ou non de traces $\tau_{\neg F}$ de $\mathcal{T}(\mathbb{C}_{\max}, \neg F)$ produisant les mêmes observations que τ_F mais aussi en fonction des traces $\tau_{\neg F}$ de $\mathcal{T}(\mathbb{C}, \neg F) \setminus \mathcal{T}(\mathbb{C}_{\max}, \neg F)$ produisant les mêmes observations mais issues du relâchement des connexions internes de $\mathbb{C}$. La conséquence est qu'un critère de précision pour $\Delta_{\mathbb{C}}$ est identique à celui pour une configuration $\mathbb{C}_{\max}$ (c.-à-d. la propriété 4) mais repose sur la propriété 5 suivante qui étend la propriété 3.

Propriété 5 *Si $\Delta_{\Gamma}(\sigma.o) = \text{F-certain}$ et $\Delta_{\mathbb{C}}(\sigma'.o) = \text{F-ambigu}$ alors il existe dans $\mathbb{C}_{\max}$ au moins une trace τ_F ($F \in \tau_F$) et dans $\mathbb{C}$ une trace $\tau_{\neg F}$ ($F \notin \tau_{\neg F}$) telles que :*
– $\mathrm{P}_{\Sigma_o}(\tau_F) = \mathrm{P}_{\Sigma_o}(\tau_{\neg F}) = \sigma'.o$,
– $\forall \tau \in \mathcal{T}(\mathbb{C}_{\max}) | \, \mathrm{P}_{\Sigma_o}(\tau) = \sigma'.o \wedge \mathrm{P}_{\Sigma_r^{ext}}(\tau) = \mathrm{P}_{\Sigma_r^{ext}}(\tau_F) \implies F \in \tau$.

5.4 Algorithme de vérification

Le premier point à remarquer est que le diagnostiqueur $\Delta_{\mathbb{C}}$ n'est précis que si le diagnostiqueur $\Delta_{\mathbb{C}_{\max}}$ l'est lui-même (cela découle directement de la définition). L'algorithme proposé est décrit en terme de langage et analyse successivement la précision de $\Delta_{\mathbb{C}_{\max}}$ puis de $\Delta_{\mathbb{C}}$. Par des mécanismes successifs d'intersection et de projection de langages, l'algorithme élimine les traces qui ne conduisent pas à un problème de précision pour ne conserver à la fin que les traces posant des problèmes. Si ce nombre de traces est fini, on en conclut que $\Delta_{\mathbb{C}}$ est précis. Comme nous l'avons vu dans l'analyse précédente, seuls les événements interactifs Σ_r^{ext} et observables Σ_o entrent en jeu dans la vérification de la précision, aussi les autres types d'événements sont abstraits par projection des traces

$\mathcal{T}(F)$ et $\mathcal{T}(\neg F)$ (lignes 2–3). Avec les lignes 4–5, l'objectif est de calculer les sources d'ambiguïté qui ne constituent pas un problème de précision (voir propriété 2) par l'intersection $\mathcal{T}(F) \cap \mathcal{T}(\neg F)$ et qui sont donc éliminées de $\mathcal{T}'(F)$. Enfin, il faut vérifier qu'ils n'existent pas dans les traces restantes de $\mathcal{T}'(F)$ un ensemble infini de traces dont la projection observable est aussi celle de traces venant de $\mathcal{T}(\neg F)$ (ligne 6). Pour cela, on calcule l'ensemble des projections observables communes à $\mathcal{T}(F)$ et à $\mathcal{T}(\neg F)$ et par projection inverse de retrouver les traces de $\mathcal{T}'(F)$ à conserver. Finalement, si $\mathcal{T}'(F)$ est fini (lignes 7–11) alors la propriété 4 est vérifiée et $\Delta_{\mathbb{C}_{\max}}$ est précis. Ensuite, il suffit de réitérer le processus (lignes 12–20) pour traiter les traces non fautives de $\mathcal{L}_{\mathbb{C}}(\neg F) \setminus \mathcal{L}_{\mathbb{C}_{\max}}(\neg F)$ et les comparer avec les traces fautives de $\mathcal{L}_{\mathbb{C}_{\max}}(F)$ afin d'établir si l'extension de la propriété 4 est également vérifiée.

Algorithme 1 Vérification de la précision de $\Delta_{\mathbb{C}}$

1: **Entrées :** Sous-configuration $\mathbb{C}$, Faute F
2: $\mathcal{T}(F) \leftarrow \mathrm{P}_{\Sigma \to \Sigma_o \cup \Sigma_r^{ext}}(\mathcal{L}_{\mathbb{C}_{\max}}(F))$
3: $\mathcal{T}(\neg F) \leftarrow \mathrm{P}_{\Sigma \to \Sigma_o \cup \Sigma_r^{ext}}(\mathcal{L}_{\mathbb{C}_{\max}}(\neg F))$
4: $Ambigu(F) \leftarrow \mathcal{T}(F) \cap \mathcal{T}(\neg F)$
5: $\mathcal{T}'(F) \leftarrow \mathcal{T}(F) \setminus (Ambigu(F))$
6: $\mathcal{T}'(F) \leftarrow \mathcal{T}'(F) \cap \mathrm{P}^{-1}_{\Sigma_o \cup \Sigma_r^{ext}}(\mathrm{P}_{\Sigma_o \cup \Sigma_r^{ext} \to \Sigma_o}(\mathcal{T}(F)) \cap \mathrm{P}_{\Sigma_o \cup \Sigma_r^{ext} \to \Sigma_o}(\mathcal{T}(\neg F)))$
7: **Si** $\mathcal{T}'(F)$ est fini **alors**
8: {La propriété 3 ne se produit pas indéfiniment.}
9: **Si** $\mathbb{C} = \mathbb{C}_{\max}$ **alors**
10: Le diagnostiqueur $\Delta_{\mathbb{C}}$ est précis
11: **sinon**
12: $\mathcal{T}'(\neg F) \leftarrow \mathrm{P}_{\Sigma \to \Sigma_o \cup \Sigma_r^{ext}}(\mathcal{L}_{\mathbb{C}}(\neg F) \setminus \mathcal{L}_{\mathbb{C}_{\max}}(\neg F))$
13: $\mathcal{T}'(F) \leftarrow \mathcal{T}(F) \setminus (Ambigu(F))$
14: $\mathcal{T}'(F) \leftarrow \mathcal{T}'(F) \cap \mathrm{P}^{-1}_{\Sigma_o \cup \Sigma_r^{ext}}(\mathrm{P}_{\Sigma_o \cup \Sigma_r^{ext} \to \Sigma_o}(\mathcal{T}(F)) \cap \mathrm{P}_{\Sigma_o \cup \Sigma_r^{ext} \to \Sigma_o}(\mathcal{T}'(\neg F)))$
15: **Si** $\mathcal{T}'(F)$ est fini **alors**
16: {La propriété 5 ne se produit pas indéfiniment.}
17: Le diagnostiqueur $\Delta_{\mathbb{C}}$ est précis
18: **sinon**
19: Problème éventuel d'imprécision de $\Delta_{\mathbb{C}}$ lié à des connexions relâchées internes
20: **Fin si**
21: **Fin si**
22: **sinon**
23: La précision de $\Delta_{\mathbb{C}}$ ne peut être démontrée à ce niveau
24: **Fin si**

5.5 Exemple illustratif

Nous reprenons l'exemple de la figure 1 et illustrons la précision. Si $\mathbb{C}$ ne comprend que le composant A, alors $\mathbb{C} = \mathbb{C}_{\max}$. Dans ce cas, $\Delta_{\mathbb{C}}$ retourne toujours un résultat ambigu. Il existe un nombre infini de traces dans laquelle la faute f a eu lieu et dont le comportement interactif est différent de celui des traces pour lesquelles f n'a pas eu lieu (ces traces débutent par $f.a3.\cdots$ ou encore $f.a2.\cdots$), la propriété 3 est vérifiée un nombre infini de fois. On remarque également que les autres traces de f débutant par $f.a1.\cdots$ ont la même projection interactive et observable que les traces dans lesquelles f n'a pas eu lieu. Ces traces sont intrinsèquement ambiguës (voir ligne 4 de l'algorithme 1). Maintenant, considérons la sous-configuration $\mathbb{C} = \{\{A, B, C, D\}, \{\langle a_2, b_2\rangle, \langle a_4, b_4\rangle, \langle a_5, b_5\rangle\}, \{\langle a_1, b_1\rangle, \langle a_3, b_3\rangle\}\}$. $\Delta_{\mathbb{C}_{\max}}$ est nécessairement précis ici car $\mathbb{C}_{\max}$ est le système complet dans cet exemple simple $\Sigma_r^{ext} = \emptyset$ et dans ce cas le $\mathcal{T}'(f)$ (ligne 7) résultant de l'algorithme est vide par construction. Il reste à savoir si le relâchement des connexions $\{\langle a_1, b_1\rangle, \langle a_3, b_3\rangle\}$ induit un problème de précision. Il suffit alors de constater que le relâchement n'induit pas une augmentation du nombre de traces observables de $\neg f$ et donc que l'ensemble restant $\mathcal{T}'(f)$ est lui aussi vide (ligne 15).

6 Choisir une sous-configuration

Nous discutons ici la façon de déterminer la sous-configuration du système qui minimise le coût du diagnostic (et en particulier la largeur arborescente) tout en conservant la précision. La sous-topologie $\mathbb{T}$ est *meilleure* que $\mathbb{T}'$ si $\mathbb{T}$ a une précision plus forte que $\mathbb{T}'$, ou qu'elle a la même précision pour un coût plus faible. L'ensemble des sous-topologies, noté Υ, sur le système est défini comme l'ensemble $2^{\mathcal{K}}$ des parties de $\mathcal{K}$ qu'il convient d'explorer pour trouver la sous-topologie *optimale*. L'ensemble partiellement ordonné $\langle \Upsilon, \subseteq, \supseteq \rangle$, où $\subseteq$ représente l'inclusion traditionelle, forme un treillis. La précision et le coût des sous-topologies ont des propriétés de monotonicité sur ce treillis. En effet, soit $\mathbb{T} \subseteq \mathbb{T}'$,

– puisque le langage décrit par $\mathbb{T}'$ est inclus dans le langage décrit par $\mathbb{T}$, le diagnostic retourné à partir de $\mathbb{T}'$ est nécessairement plus précis que le diagnostic retourné à partir de $\mathbb{T}$;
– d'autre part, tout arbre de jonction de $\mathbb{T}'$ est également un arbre de jonction de $\mathbb{T}$. Donc, la largeur d'arbre de $\mathbb{T}$ est au pire égale à la largeur d'arbre de $\mathbb{T}'$.

Il est donc possible d'explorer le treillis de manière efficace, soit en partant de la topologie du système et en retirant des connexions jusqu'à obtenir une sous-topologie non précise, soit au contraire en partant de la sous-topologie vide et en ajoutant des connexions jusqu'à obtenir une sous-topologie précise. Nous nous plaçons dans un contexte où le nombre de composants est potentiellement très grand – plusieurs milliers – tandis que nous souhaitons construire des sous-topologies de largeur arborescente très faible – quelques unités tout au plus. Aussi, nous préconisons une approche partant de $\mathbb{T}_{\perp} = \emptyset$ et ajoutant incrémentalement de nouvelles connexions. Une fois une sous-topologie précise construite, il est possible de raffiner celle-ci en ignorant des connexions ajoutées inutilement. L'exploration est résumée Algorithme 2.

Algorithme 2 Exploration de Υ

1: **Entrée :** Γ, F
2: $\mathbb{T} := \emptyset$
3: **Tant que** $\mathbb{T}$ n'est pas précise, **faire**
4: Ajouter une connexion à $\mathbb{T}$.
5: **Fin tant que**
6: **Tant que** $\exists c \in \mathbb{T}$ tel que $\mathbb{T} \setminus \{c\}$ est précise, **faire**
7: Supprimer c dans $\mathbb{T}$.
8: **Fin tant que**
9: **Renvoyer** $\mathbb{T}$

L'exploration de Υ peut être améliorée en utilisant les heuristiques suivantes :

– L'analyse de précision d'une sous-topologie $\mathbb{T}$ permet de générer une explication de non précision. En effet dès lors que l'algorithme de vérification détermine que $\mathcal{T}'(F)$ est infini, il est en mesure d'affirmer quelles connexions sont impliquées dans la perte de précision. Il est donc possible de modifier de l'Algorithme 2 en choisissant intelligemment les connexions ajoutées ligne 4.
– Si l'arbre de jonction construit à partir de $\mathbb{T}$ est également un arbre de jonction pour $\mathbb{T}' \supseteq \mathbb{T}$, alors le coût associé au second n'est pas supérieur au coût du premier, et il est possible d'effectuer le calcul de précision directement sur $\mathbb{T}'$.

L'algorithme proposé dans l'article ne renvoie pas la solution optimale, tout d'abord parce que les critères de précision décrits dans la section précédente ne sont pas nécessaires, et ensuite parce que l'exploration est locale.

7 Conclusion

Nous proposons dans cet article une approche originale pour réduire la complexité du diagnostic à base de modèle sur des systèmes à événements discrets. Elle consiste à ignorer des connexions du système tout en s'assurant que le diagnostic reste précis. L'intrication des connexions entre les composants du système influence directement la complexité du raisonnement diagnostique. Donc, considérer le plus petit nombre de connexions possible permet de rendre la tâche moins compliquée et de fournir un résultat en temps raisonnable. Conserver la précision du diagnostic après avoir ignoré des connexions n'est pas trivial. Pour le faire, nous effectuons une analyse sur le système pour vérifier quelles connexions peuvent être omises sans compromettre le diagnostic.

Dans des travaux futurs, nous pensons pouvoir améliorer différents aspects de la technique proposée dans cet article. Pour commencer, au lieu d'approximer le coût du diagnostic uniquement par la largeur arborescente du système, nous souhaiterions prendre en compte d'autre facteurs comme le nombre de nœuds dans l'arbre, l'arborescence de l'arbre, l'observabilité (i.e. la proportion d'événements observables) dans les composants de chaque nœud.

Nous souhaitons aussi modifier le critère de précision à présent Booléen pour qu'il puisse prendre une valeur réelle reflétant le degré de précision d'une sous-configuration à un moment donné. Cela nous permettra de trouver un compromis entre coût de diagnostic et degré de précision.

Un autre problème intéressant est la considération de plusieurs fautes en même temps au lieu de ne regarder qu'une faute à la fois. Chaque faute étant associée à un arbre de jonction, on peut intégrer tous les arbres de jonctions pour obtenir un diagnostic avec plusieurs fautes. On pourra alors réutiliser les calculs sur l'un des arbres pour le calcul sur un autre arbre, en particulier si ceux-ci partagent des grappes de composants.

Références

[1] A. Benveniste, E. Fabre, S. Haar, and Cl.. Jard. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Transactions on Automatic Control*, 48(5) :714–727, 2003.

[2] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[3] M.-O. Cordier and A. Grastien. Exploiting independence in a decentralised and incremental approach of diagnosis. In M. Veloso, editor, *Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 292–297. AAAI press, 2007.

[4] Ch.. Dousson and T. V. Duong. Découverte de chroniques avec contraintes temporelles à partir de journaux d'alarmes pour la supervision de réseaux de télécommunications. In *Première Conf'erence francophone sur l'Apprentissage automatique (CAP-99)*, pages 17–24, 1999.

[5] A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In R. Holte, editor, *Nineteenth National Conference on Artificial Intelligence (AAAI-07)*. AAAI Press, 2007.

[6] P. Kan John and A. Grastien. Local consistency and junction tree for diagnosis of discrete-event systems. In *European Conference on Artificial Intelligence (ECAI-08)*, 2008.

[7] Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence (AIJ)*, 164 :121–170, 2005.

[8] Y. Pencolé, D. Kamenetsky, and A. Schumann. Towards low-cost diagnosis of component-based systems. In *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Process (SAFEPROCESS)*, Beijing, P.R. China, aug 2006.

[9] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9) :1555–1575, 1995.

[10] A. Schumann, Y. Pencolé, and S. Thiébaux. Symbolic models for diagnosing discrete-event systems. In *Sixteenth European Conference on Artificial Intelligence (ECAI'04)*, 2004.

[11] Y. Wang, T.-S. Yoo, and S. Lafortune. New results on decentralized diagnosis of discrete event systems. In *42nd Annual Allerton Conference on Communication, Control, and Computing*, 2004.