Résultats de complexité pour le problème de la composition d'agents

P. Balbiani
Philippe.Balbiani@irit.fr

F. Cheikh
Fahima.Cheikh@irit.fr

G. Feuillade
Guillaume.Feuillade@irit.fr

CNRS- Université de Toulouse
Institut de recherche en informatique de Toulouse

Résumé :
Pour que des agents puissent réaliser des tâches complexes, il est nécessaire de composer entre eux ces agents. La composition d'agents a été caractérisée comme étant un problème de synthèse de plan ou comme étant un problème de synthèse de logiciel : étant donnés un but et un ensemble d'agents, générer une composition des agents qui satisfait le but. Nous proposons des algorithmes, basés sur des résultats récents de synthèse automatique de contrôleurs, pour réaliser la composition automatique d'agents. Nous donnons également des résultats concernant la complexité de ce problème.

Mots-clés : Composition d'agents, synthèse de contrôleurs, complexité et automates.

Abstract:
In order to realize complex tasks by agents, it is sometimes necessary to compose them. The agent composition has been characterized either as a plan synthesis problem or as a software synthesis problem : given a goal and a set of agents, generate a composition of the agents that satisfies the goal. We propose algorithms, based on recent advances in automated synthesis of controllers, for performing automated composition of agents. We also give complexity result concerning this problem.

Keywords: Composition of agents, controller synthesis, complexity and automata.

1 Introduction

Considérons une communauté d'agents, chacun capable de réaliser une tâche simple et un objectif qui est une tâche complexe. Aucun agent n'étant capable de réaliser seul l'objectif, il est nécessaire de faire coopérer les agents pour le mener à bien.

Plusieurs approches permettent de fabriquer automatiquement une solution permettant à la communauté d'agents de réaliser l'objectif. L'approche que nous privilégions dans cet article consiste à considérer les agents comme des services [7, 10]. Nous cherchons alors à composer ces services de telle manière que la composition obtenue réalise l'objectif. La composition se fait alors en fabriquant un agent virtuel, nommé *médiateur*, dont le rôle est de communiquer avec les agents de la communauté pour les amener à se coordonner afin de réaliser l'objectif.

Différentes techniques automatisées ont été proposées pour résoudre le problème de la composition de services [12]. On distingue principalement deux types de méthode de résolution. Le premier consiste à réduire le problème de la composition à un problème de synthèse de plan [14, 15, 18]. Ces méthodes permettent de traiter des problématiques complexes, cependant, la complexité des algorithmes de résolution n'a pas été étudiée en détail. Le deuxième consiste à faire appel aux méthodes de synthèse de logiciel [3, 4, 5]. En pratique cette approche a été utilisée dans le cadre de la composition de services Web ; malheureusement, les solutions reposent sur des restrictions fortes, comme par exemple l'absence de communication entre agents.

Dans cet article, les agents sont abstraits sous la forme d'automates capables de communiquer entre eux. L'échange d'information se fait sous la forme de messages envoyés via des canaux de communication. Nous ne détaillons pas le contenu des messages et chaque canal contient au plus un unique message. Ces restrictions de communication sont nécessaires d'une part pour garantir l'existence d'un algorithme de résolution et d'autre part pour permettre l'étude de la complexité précise du problème.
La tâche à accomplir est donnée sous la forme d'un automate. L'objectif de la composition est de synthétiser un agent médiateur, qui peut uniquement communiquer avec les agents de la communauté afin de garantir que la communauté plus l'agent médiateur ait un comportement similaire ou équivalent à la tâche à accomplir. Nous étudions alors la complexité de ce problème pour les notions de similarité que sont la simulation d'automates ou l'inclusion de traces ainsi que pour les notions d'équivalence que sont la bisimulation d'automates ou l'égalité de traces. Dans la plupart des cas nous proposons un algorithme de résolution.

L'article se présente comme suit. Dans la section 2 un exemple de scénario est présenté. La section 3 définit les agents, leur mode de communication ainsi que les notions de similarité et d'équivalence. La section 4 introduit le problème de la composition : étant donnés un but

et un ensemble d'agents, générer une composition des agents qui satisfait le but. Des résultats de complexités et des algorithmes sont donnés dans la section 5. La section 6 conclut l'article ; nous y présentons certains problèmes qui restent ouverts.

2 Exemple

Pour cet article, nous proposons un exemple sous la forme d'un scénario dans lequel deux agents doivent gérer une situation où il faut sauver un individu de la noyade. Ces deux agents sont hétérogènes, de ce fait leurs canaux de communication ne sont pas forcément les mêmes. Le premier agent intitulé agent *Pilote* peut effectuer trois actions :

- *?Region* : recevoir un message indiquant la position approximative de l'individu en danger,
- *LocaliseVict* : déterminer la position exacte de la victime et
- *!Position* : envoyer un rapport pour confirmer sa position exacte.

Chacune de ces actions est complexe et représente en réalité une séquence d'actions conditionnées. Nous nous intéressons cependant ici à une abstraction de ces actions, que nous considérons comme atomiques.

Le second agent intitulé agent *Sauveteur* peut effectuer les actions suivantes :

- *?Données* : recevoir la position exacte de la victime,
- *RecupVict* : récupérer la victime et
- *!Rapport* : envoyer un rapport pour signaler la réussite ou l'échec de la mission.

Le problème que nous traitons dans cet article est le suivant : étant donné un agent appelé agent *but* qui représente la ou les séquences d'actions souhaitées, créer de façon automatique un agent médiateur, qui communique avec les agents du système afin que le système composé du médiateur et des deux agents soit équivalent à celui de l'agent but (au sens de l'inclusion de traces, l'équivalence de trace, la simulation et la bisimulation). Dans cet exemple, l'agent but devra à la fois déterminer la position de la victime et la récupérer. Notons que l'agent but n'est pas réel. Notons également que pour des raisons de clarté, cet exemple est restrictif par rapport aux problématiques traités dans cet article : il ne présente que 2 agents, pas de communication directe entre agents et les agents sont déterministes.

3 Automates finis et agents

Dans cette section, nous présentons notre modèle des agents basé sur des automates finis communiquant de manière asynchrone. Tout d'abord, nous donnons les définitions utiles concernant les automates finis. Ensuite, nous définissons formellement les agents.

3.1 Automate fini

Soit Σ un ensemble fini d'actions. Un automate fini défini sur Σ est une structure $\mathcal{A} = (S, \Delta, s^{in})$ où S est un ensemble fini d'états, Δ est une fonction $\Delta : S \times \Sigma \to 2^S$ et $s^{in} \in S$ est un état initial. Pour tout $\Sigma' \subseteq \Sigma$, la relation $\to_{\mathcal{A}}^{\Sigma'} \subseteq S \times S$ décrit le changement d'état de l'automate suite à l'exécution d'une action de Σ'. Formellement, $s \to_{\mathcal{A}}^{\Sigma'} t$ ssi il existe $a \in \Sigma'$ tel que $t \in \Delta(s, a)$. On note $\to_{\mathcal{A}}^{\Sigma'\star}$ la fermeture reflexive transitive de $\to_{\mathcal{A}}^{\Sigma'}$. Pour tout $\Sigma' \subseteq \Sigma$, nous dirons que $\mathcal{A}$ *boucle* sur Σ' ssi pour tout $a \in \Sigma'$, $\to_{\mathcal{A}}^{\{a\}} = Id_S$.

Soit Σ' un ensemble fini d'actions considérées comme étant non-observables, un chemin pour $\mathcal{A}$ modulo Σ' est une séquence finie de la forme (s^{in}, a_1, s^1), (s^1, a_2, s^2),..., $(sn-1, a_n, s^n)$ telle que $s^{in} \to_{\mathcal{A}}^{\Sigma'\star} \circ \to_{\mathcal{A}}^{\{a_1\}} \circ \to_{\mathcal{A}}^{\Sigma'\star} s^1$ et pour tout $i \in \{1, \ldots, n-1\}$, $s^i \to_{\mathcal{A}}^{\Sigma'\star} \circ \to_{\mathcal{A}}^{\{a_{i+1}\}} \circ \to_{\mathcal{A}}^{\Sigma'\star} s^{i+1}$. Le mot $a_1 \ldots a_n$ est sa *trace*. L'ensemble des traces de tous les chemins pour $\mathcal{A}$ modulo Σ' est noté $Tr_{\Sigma'}(\mathcal{A})$.

3.2 Inclusion et équivalence de traces

Soient $\mathcal{A}_1 = (S_1, \Delta_1, s_1^{in})$ et $\mathcal{A}_2 = (S_2, \Delta_2, s_2^{in})$ des automates finis définis sur Σ. Pour tout $\Sigma' \subseteq \Sigma$ Nous dirons que $\mathcal{A}_1$ est *inclus pour la trace* dans $\mathcal{A}_2$ modulo Σ', noté $\mathcal{A}_1 \subseteq_{tr} \mathcal{A}_2(\Sigma')$, lorsque $Tr_{\Sigma'}(\mathcal{A}_1) \subseteq Tr_{\Sigma'}(\mathcal{A}_2)$. Nous dirons que $\mathcal{A}_1$ et $\mathcal{A}_2$ sont *équivalents pour la trace* modulo Σ', noté $\mathcal{A}_1 \equiv_{tr} \mathcal{A}_2(\Sigma')$, lorsque $Tr_{\Sigma'}(\mathcal{A}_1) = Tr_{\Sigma'}(\mathcal{A}_2)$. Considérons le cas où les automates finis $\mathcal{A}_1$ et $\mathcal{A}_2$ sont représentés par la Figure 1. Dans ce cas, les deux automates sont équivalents pour la trace modulo $\emptyset$. En effet, $Tr_\emptyset(\mathcal{A}_1) = Tr_\emptyset(\mathcal{A}_2) = \{\epsilon, a, ab\}$.

3.3 Simulation et Bisimulation

Soient $\mathcal{A}_1 = (S_1, \Delta_1, s_1^{in})$ et $\mathcal{A}_2 = (S_2, \Delta_2, s_2^{in})$ deux automates finis définis sur Σ. Pour tout

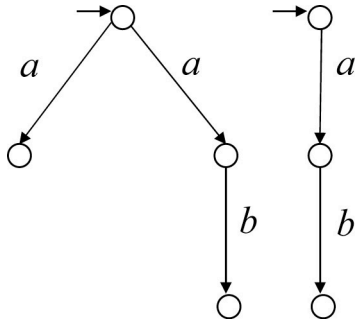

FIG. 1 – De gauche à droite, automates finis $\mathcal{A}_1$ et $\mathcal{A}_2$

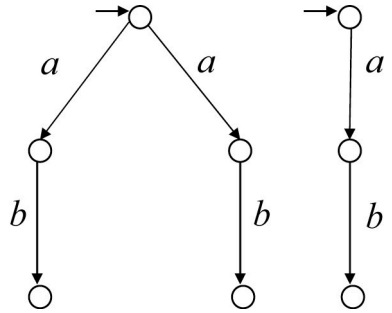

FIG. 2 – De gauche à droite, automates finis $\mathcal{A}_1$ et $\mathcal{A}_2$

$\Sigma' \subseteq \Sigma$, une relation $Z \subseteq S_1 \times S_2$ telle que $(s_1^{in}, s_2^{in}) \in Z$ est appelée *simulation* entre $\mathcal{A}_1$ et $\mathcal{A}_2$ modulo Σ' ssi les conditions suivantes sont satisfaites pour tout $(s_1, s_2) \in Z$ et pour tout $a \in \Sigma \setminus \Sigma'$:

– pour tout $t_1 \in S_1$, si $s_1 \rightarrow_{\mathcal{A}_1}^{\Sigma'\star} \circ \rightarrow_{\mathcal{A}_1}^{\{a\}} \circ \rightarrow_{\mathcal{A}_1}^{\Sigma'\star} t_1$ alors il existe $t_2 \in S_2$ tel que $s_2 \rightarrow_{\mathcal{A}_2}^{\Sigma'\star} \circ \rightarrow_{\mathcal{A}_2}^{\{a\}} \circ \rightarrow_{\mathcal{A}_2}^{\Sigma'\star} t_2$ et $(t_1, t_2) \in Z$,

De plus, pour tout $\Sigma' \subseteq \Sigma$, s'il y a une simulation entre $\mathcal{A}_1$ et $\mathcal{A}_2$ modulo Σ' alors nous écrirons $\mathcal{A}_1 \leq_{si} \mathcal{A}_2(\Sigma')$ et nous dirons que $\mathcal{A}_1$ est *simulé* par $\mathcal{A}_2$.

Une relation $Z \subseteq S_1 \times S_2$ telle que $(s_1^{in}, s_2^{in}) \in Z$ est appelée *bisimulation* entre $\mathcal{A}_1$ et $\mathcal{A}_2$ modulo Σ' ssi les conditions suivantes sont satisfaites pour tout $(s_1, s_2) \in Z$ et pour tout $a \in \Sigma \setminus \Sigma'$:

– pour tout $t_1 \in S_1$, si $s_1 \rightarrow_{\mathcal{A}_1}^{\Sigma'\star} \circ \rightarrow_{\mathcal{A}_1}^{\{a\}} \circ \rightarrow_{\mathcal{A}_1}^{\Sigma'\star} t_1$ alors il existe $t_2 \in S_2$ tel que $s_2 \rightarrow_{\mathcal{A}_2}^{\Sigma'\star} \circ \rightarrow_{\mathcal{A}_2}^{\{a\}} \circ \rightarrow_{\mathcal{A}_2}^{\Sigma'\star} t_2$ et $(t_1, t_2) \in Z$,

– pour tout $t_2 \in S_2$, si $s_2 \rightarrow_{\mathcal{A}_2}^{\Sigma'\star} \circ \rightarrow_{\mathcal{A}_2}^{\{a\}} \circ \rightarrow_{\mathcal{A}_2}^{\Sigma'\star} t_2$ alors il existe $t_1 \in S_1$ tel que $s_1 \rightarrow_{\mathcal{A}_1}^{\Sigma'\star} \circ \rightarrow_{\mathcal{A}_1}^{\{a\}} \circ \rightarrow_{\mathcal{A}_1}^{\Sigma'\star} t_1$ et $(t_1, t_2) \in Z$.

De plus, pour tout $\Sigma' \subseteq \Sigma$, s'il y a une bisimulation entre $\mathcal{A}_1$ et $\mathcal{A}_2$ modulo Σ' alors nous écrirons $\mathcal{A}_1 \longleftrightarrow_{bi} \mathcal{A}_2\ (\Sigma')$ et nous dirons que $\mathcal{A}_1$ et $\mathcal{A}_2$ sont *bisimilaires*.

Rappelons que dans le cas où les automates finis $\mathcal{A}_1$ et $\mathcal{A}_2$ sont déterministes, l'inclusion de trace (resp. équivalence de trace) et la simulation (resp. bisimulation) sont la même relation, ce n'est pas le cas pour les automates non déterministes. La relation de simulation (resp. bisimulation) est plus restrictive que celle de l'inclusion de trace (resp. l'équivalence de trace). Par exemple, dans la Figure 1, l'automate $\mathcal{A}_1$ simule l'automate $\mathcal{A}_2$; en revanche ces deux automates ne sont pas bisimilaires. En effet, après avoir effectué l'action a l'automate $\mathcal{A}_1$ peut se retrouver dans un état où il ne peut plus effectuer l'action b ; ce n'est pas le cas pour l'automate $\mathcal{A}_2$. Dans la Figure 2, les automates $\mathcal{A}_1$ et $\mathcal{A}_2$ sont bisimilaires.

3.4 Produit

Soient $\mathcal{A}_1 = (S_1, \Delta_1, s_1^{in})$ et $\mathcal{A}_2 = (S_2, \Delta_2, s_2^{in})$ deux automates finis sur Σ. Par $\mathcal{A}_1 \otimes \mathcal{A}_2$, nous notons le *produit asynchrone* de $\mathcal{A}_1$ et $\mathcal{A}_2$, c.-à-d. l'automate fini $\mathcal{A} = (S, \Delta, s^{in})$ sur Σ tel que $S = S_1 \times S_2$, Δ est la fonction définie par $(t_1, t_2) \in \Delta((s_1, s_2), a)$ ssi $t_1 \in \Delta_1(s_1, a)$ et $t_2 = s_2$ ou $t_1 = s_1$ et $t_2 \in \Delta_2(s_2, a)$ et $s^{in} = (s_1^{in}, s_2^{in})$. Par $\mathcal{A}_1 \times \mathcal{A}_2$, nous notons le *produit synchrone* de $\mathcal{A}_1$ et $\mathcal{A}_2$, c.-à-d. l'automate fini $\mathcal{A} = (S, \Delta, s^{in})$ sur Σ tel que $S = S_1 \times S_2$, Δ est la fonction définie par $(t_1, t_2) \in \Delta((s_1, s_2), a)$ ssi $t_1 \in \Delta_1(s_1, a)$ et $t_2 \in \Delta_2(s_2, a)$ et $s^{in} = (s_1^{in}, s_2^{in})$.

3.5 Modèle des agents

Dans les travaux concernant la composition de services Web qui ont été réalisé par [3, 4, 5], les services sont représentés par des automates finis à entrée/sortie. Les services communiquent par la transmission asynchrone de messages à travers des canaux de communication. Notre modèle pour les agents s'inspire de ces travaux. Comme leurs auteurs, nous supposons que la communication à travers les canaux est fiable (il n'y a pas de perte de messages durant leurs transmission). Pour simplifier, dans cet article nous considérons une abstraction du contenu des messages. De plus, nous supposons que les canaux ne peuvent pas contenir plus d'un message à la fois. Formellement, soit Π un ensemble fini de canaux et Σ un ensemble fini d'actions, un *agent* défini sur Π et Σ est un automate fini

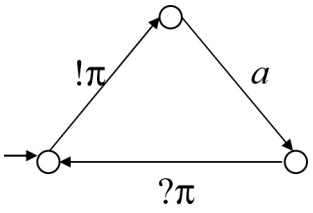

FIG. 3 –

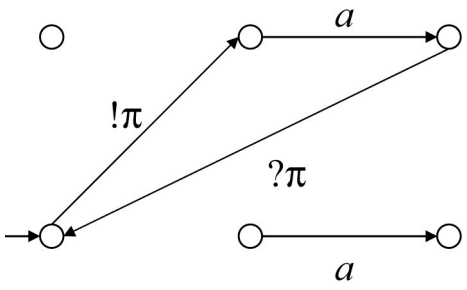

FIG. 4 –

défini sur $(\{!, ?\} \times \Pi) \cup \Sigma$. Pour tout $\pi \in \Pi$, l'action de transmission $!\pi$ consiste à ajouter un message sur le canal π, tandis que l'action de réception $?\pi$ consiste à retirer un message du canal π. L'action $!\pi$ peut être exécutée à condition que le canal ne soit pas plein, c.-à-d. π ne contient pas de message. L'action $?\pi$ peut être exécutée à condition que le canal ne soit pas vide, c.-à-d. π contient un message.

Soit $\mathcal{A} = (S, \Delta, s^{in})$ un agent défini sur $(\{!, ?\} \times \Pi) \cup \Sigma$. L'ensemble des *exécutions* de l'agent $\mathcal{A}$ est un automate fini $Exec(\mathcal{A}) = (S', \Delta', s^{in'})$ défini sur $(\{!, ?\} \times \Pi) \cup \Sigma$ tel que $S' = S \times 2^{\Pi}$, Δ' est la fonction définie par
– $(t, Q) \in \Delta'((s, P), !\pi)$ ssi $t \in \Delta(s, !\pi)$, $\pi \notin P$, $Q = P \cup \{\pi\}$,
– $(t, Q) \in \Delta'((s, P), ?\pi)$ ssi $t \in \Delta(s, ?\pi)$, $\pi \in P$, $Q = P \setminus \{\pi\}$,
– $(t, Q) \in \Delta'((s, P), a)$ ssi $t \in \Delta(s, a)$, $Q = P$,
et $s^{in'} = (s^{in}, \emptyset)$ (initialement tous les canaux sont vides). Observons que $Exec(\mathcal{A})$ est de taille exponentielle par rapport à Π et que nous pouvons le construire en un temps exponentiel. Considérons le cas où $\mathcal{A}$ est l'automate fini représenté dans la Figure 3. Dans ce cas $Exec(\mathcal{A})$ est l'automate fini représenté dans la Figure 4.

4 La composition d'agents

L'objectif de la composition est donné sous la forme d'un but. Le problème s'énonce de la manière suivante : étant donnés un but et un ensemble d'agents, générer une composition d'agents satisfaisant le but. Dans les travaux [14, 15, 18], les buts sont des conditions sur le comportement de la composition et ils peuvent être exprimés dans le langage *EaGLe*. Dans cette approche, les auteurs s'intéressent à la composition de services Web, problème qui consiste à combiner et coordonner les services disponibles en un processus complexe satisfaisant la condition donnée. Selon [3, 4, 5], les buts sont des automates finis à entrée/sortie, ils sont définis comme un agent : le but est l'agent complexe obtenu par la composition, il représente le comportement global souhaité pour la communauté d'agents. Dans notre cas, l'agent but partage la même définition que les agents de la communauté (voir Section 3.5). La composition consiste à combiner et coordonner les services disponibles en un processus complexe qui simule l'automate fini de l'agent but. Cette composition s'effectue en fabriquant un processus *médiateur* qui va organiser la communauté d'agents via des échanges de messages. Dans cet article, nous automatisons la composition définie dans la seconde approche avec pour objectif de vérifier l'une des quatre relations suivantes entre la composition obtenue et l'agent but : inclusion de traces, équivalence de traces, simulation et bisimulation. Cela nous amène au problème de décision suivant :

> $CP(\approx)$: étant donnés un ensemble fini d'actions Σ, deux ensembles finis de canaux Π et $\Pi' \subseteq \Pi$, des automates finis $\mathcal{A} = (S_{\mathcal{A}}, \Delta_{\mathcal{A}}, s^{in}_{\mathcal{A}})$, $\mathcal{B}_1 = (S_{\mathcal{B}_1}, \Delta_{\mathcal{B}_1}, s^{in}_{\mathcal{B}_1})$, …, $\mathcal{B}_n = (S_{\mathcal{B}_n}, \Delta_{\mathcal{B}_n}, s^{in}_{\mathcal{B}_n})$ définis sur $(\{!, ?\} \times \Pi) \cup \Sigma$, déterminer s'il existe un automate fini $\mathcal{C} = (S_{\mathcal{C}}, \Delta_{\mathcal{C}}, s^{in}_{\mathcal{C}})$ défini sur $\{!, ?\} \times \Pi$ tel que $Exec(\mathcal{A}) \approx Exec(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes \mathcal{C})$ $(\{!, ?\} \times \Pi')$.

Dans $CP(\approx)$, l'automate fini $\mathcal{A}$ joue le rôle du but et les automates $\mathcal{B}_1$, …, $\mathcal{B}_n$ jouent le rôle des agents disponibles. L'automate fini $\mathcal{C}$, quant à lui joue le rôle de médiateur entre les agents ; il va donc coordonner les agents de sorte à obtenir, en faisant abstraction des communications internes (les communications sur les canaux de Π'), un processus équivalent au but. En réalité le problème $CP(\approx)$ représente quatre problèmes différents suivant que $\approx$ est égale à $\subseteq_{tr}$, $\equiv_{tr}$, $\leq_{si}$ ou $\longleftrightarrow_{bi}$.
Intuitivement, vouloir que le comportement du but soit inclus dans celui des agents $\mathcal{B}_1$, …, $\mathcal{B}_n$ coordonnés par $\mathcal{C}$ signifie que toute séquence d'actions effectuée par le but peut également être effectuée par les agents coordonnés par le médiateur (en permettant éventuellement aux

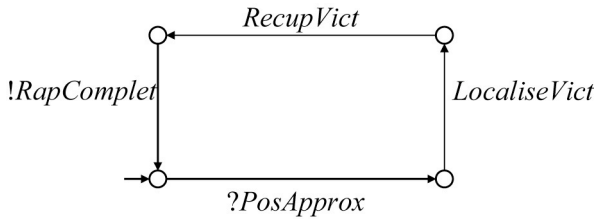

FIG. 5 – Agent *But*

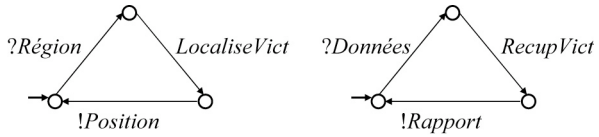

FIG. 6 – De gauche à droite agent *Pilote* et agent *Sauveteur*

agents d'effectuer des actions supplémentaires). Dans le cas où l'équivalence de trace est considérée, les agents coordonnés par le médiateur ne peuvent pas effectuer de séquence d'actions que le but ne peut pas effectuer. Par exemple, cela est utile pour garantir la sécurité du système en interdisant des séquences d'actions potentiellement dangeureuses. Les notions de simulation et de bisimulation, à la différence des notions d'inclusion de traces et d'équivalence de traces, se basent sur les états des systèmes à comparer. Par exemple, deux états sont similaires si on y accède avec la même séquence (passé identique) d'actions et que les séquences à partir de ces deux états sont identiques (futur identique). En particulier, la bisimulation garantit que les systèmes ont, pour un passé identique, les mêmes actions disponibles. Ainsi, ce sont des notions plus restrictives que l'inclusion et l'équivalence de traces.
Considérons notre scénario. Soit $\mathcal{A}$, $\mathcal{B}_1$, $\mathcal{B}_2$, les automates finis représentés respectivement dans les Figures 5 et 6. Dans ces Figures, *LocaliseVict* et *RecupVict* sont des actions et *Position*, *RapComplet*, *PosApprox*, *Données*, *Région* et *Rapport* sont des canaux.

Dans ce cas, l'automate fini $\mathcal{C}$ de la Figure 7 est tel que $Exec(\mathcal{A}) \longleftrightarrow_{bi} Exec(\mathcal{B}_1 \otimes \mathcal{B}_2 \otimes \mathcal{C})$ $(\{!, ?\} \times \{$*Position, Données, Rapport, Région*$\})$.

5 Complexité et algorithmes

Dans cette section, nous étudions la complexité des quatre problèmes de la composition définis dans la section précédente. Nous donnons les algorithmes effectifs pour l'inclusion de traces, la

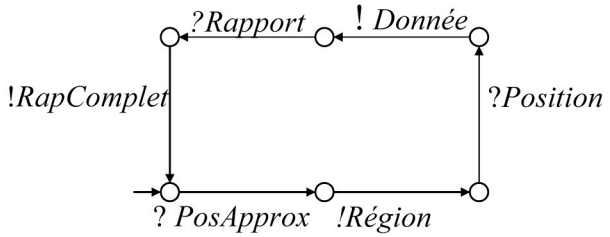

FIG. 7 – Agent *Médiateur*

simulation et la bisimulation.

5.1 Réductions en espace logarithmique

Dans un premier temps, nous présentons des réductions en espace logarithmique pour les problèmes $CP(\subseteq_{tr})$, $CP(\equiv_{tr})$, $CP(\leq_{si})$ et $CP(\longleftrightarrow_{bi})$.

Théorème 1 *Le problème* $CP(\subseteq_{tr})$ *est EXPSPACE-difficile.*

Preuve: Cette preuve se base sur une réduction du problème d'universalité des langages réguliers avec carré. Soit Σ un alphabet fini et Σ^* l'ensemble des mots de longueur finie sur Σ. Nous noterons le mot vide (de longueur 0) par ϵ. Une expression régulière avec carré α définie sur Σ est construite comme suit

$\alpha := \epsilon \mid a \mid \alpha'.\alpha'' \mid \alpha' + \alpha'' \mid \alpha^+ \mid \alpha^2$, où $a \in \Sigma$.

Chaque expression α définit un langage rationnel sur Σ. Nous notons par $L(\alpha)$ le langage défini par α. Ainsi, par exemple, $L(\alpha^2) = L(\alpha).L(\alpha)$.

Considérons le problème de décision suivant :
ER : étant donnés un alphabet fini Σ et une expression régulière avec carré α définie sur Σ, déterminer si $L(\alpha)$=Σ^*.
Sachant que ER est ***EXPSPACE***-difficile [11], il suffit de réduire ER à $CP(\subseteq_{tr})$ afin de montrer que $CP(\subseteq_{tr})$ est ***EXPSPACE***-difficile.
Étant donnée un alphabet fini Σ et une expression régulière avec carré α définie sur Σ, nous devons déterminer si $L(\alpha)$=Σ^*. L'instance $\rho(\Sigma, \alpha)$ de $CP(\subseteq_{tr})$ que nous construisons est donnée par l'ensemble fini d'actions Σ', l'ensemble fini de canaux Π_α, les automates finis $\mathcal{A} = (S_\mathcal{A}, \delta_\mathcal{A}, s^{in}_\mathcal{A})$ et $\mathcal{B}_\alpha = (S_\mathcal{B}, \delta_\mathcal{B}, s^{in}_\mathcal{B})$ sur $(\{!, ?\} \times \Pi_\alpha) \cup \Sigma'$ et $\Pi'_\alpha \subseteq \Pi$ sont définis par
– $\Sigma' = \Sigma$,
– Π_α est construit par induction à partir de α,

– $\Pi'_\alpha = \Pi_\alpha$,
– $\mathcal{A}$ est l'automate fini de la Figure 8,
– $\mathcal{B}_\alpha$ est construit par induction à partir de α.

L'automate $\mathcal{A}$ est construit de telle sorte que le langage reconnu par $\mathcal{A}$, noté $L(\mathcal{A})$ soit égal à Σ^*. Maintenant, nous donnons la construction de $\mathcal{B}_\alpha$ et Π_α en fonction de α. La construction de $\mathcal{B}_\alpha$ est telle que $L(\mathcal{B}_\alpha) = L(\alpha)$.

– Cas où $\alpha = \epsilon$, $\Pi_\epsilon = \{\pi\}$ et $\mathcal{B}_\epsilon$ est l'automate fini de la Figure 9.
– Cas où $\alpha = a$, $\Pi_a = \emptyset$ et $\mathcal{B}_a$ est l'automate fini de la Figure 10.
– Cas où $\alpha = \gamma.\sigma$, $\Pi_{\gamma.\sigma} = \Pi_\gamma \cup \Pi_\sigma \cup \{\pi\}$ où π est un nouveau canal et $\mathcal{B}_{\gamma.\sigma}$ est l'automate fini de la Figure 11.
– Cas où $\alpha = \gamma + \sigma$, $\Pi_{\gamma+\sigma} = \Pi_\gamma \cup \Pi_\sigma \cup \{\pi\}$ où π est un nouveau canal et $\mathcal{B}_{\gamma+\sigma}$ est l'automate fini de la Figure 12.
– Cas où $\alpha = \gamma^+$, $\Pi_{\gamma^+} = \Pi_\gamma \cup \{\pi\}$ où π est un nouveau canal et $\mathcal{B}_{\gamma^+}$ est l'automate fini de la Figure 13.
– Cas où $\alpha = \gamma^2$, $\Pi_{\gamma^2} = \Pi_\gamma \cup \{\pi, \pi'\}$ où π et π' sont de nouveaux canaux et $\mathcal{B}_{\gamma^2}$ est l'automate fini de la Figure 14.

Cela complète la construction. Il est évident que ρ peut être calculé en espace logarithmique. De plus, $L(\alpha)$=Σ^* ssi il existe un automate fini $\mathcal{C} = (S_\mathcal{C}, \Delta_\mathcal{C}, s^{in}_\mathcal{C})$ défini sur $\{!, ?\} \times \Pi_\alpha$ tel que $Exec(\mathcal{A}) \subseteq_{tr} Exec(\mathcal{B}_\alpha \otimes \mathcal{C})$ $(\{!, ?\} \times \Pi'_\alpha)$. Ainsi, $CP(\subseteq_{tr})$ est ***EXPSPACE***-difficile. ⊣

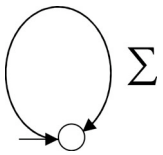

FIG. 8 –

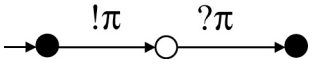

FIG. 9 –

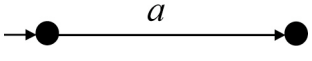

FIG. 10 –

Théorème 2 *Le problème $CP(\equiv_{tr})$ est EXPSPACE-difficile.*

Preuve: Cette preuve se base sur la réduction du problème d'équivalence de trace de réseaux

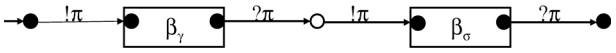

FIG. 11 –

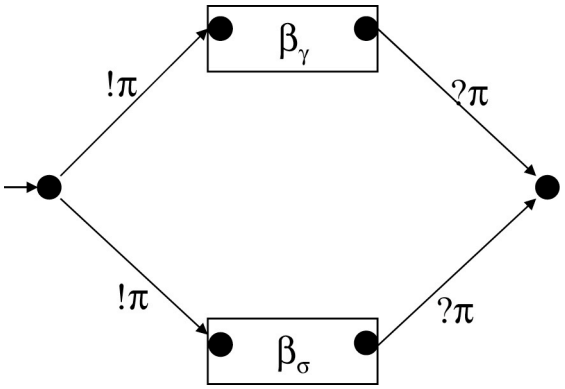

FIG. 12 –

de Petri saufs. Soit Σ un ensemble fini d'actions. Un réseau de Petri sauf défini sur Σ est une structure de la forme $\mathcal{N} = (P, T, F, l)$ où P est un ensemble fini de places, T est un ensemble fini de transitions, $F \subseteq (P \times T) \cup (T \times P)$ est une relation et $l : T \to \Sigma$ est une fonction Pour toute transition $t \in T$, soit ${}^\bullet t$ l'ensemble de toutes les places $p \in P$ telles que $p\ F\ t$ et $t^\bullet$ l'ensemble de toutes les places $p \in P$ telles que $t\ F\ p$. Soit $Exec(\mathcal{N})$=$(S', \Delta', u^{in'})$, l'automate fini défini sur Σ tel que $S' = 2^P$, Δ' est une fonction définie par : $v' \in \Delta'(u', a)$ ssi il existe $t \in T$ tel que $l(t) = a$, ${}^\bullet t \subseteq u'$ et $v' = (u' \setminus {}^\bullet t) \cup t^\bullet$, $u^{in'} = \emptyset$. Considérons le problème de décision suivant :

$RP(\equiv_{tr})$: étant donnés un ensemble fini d'actions Σ et des réseaux de Petri saufs $\mathcal{N} = (P_\mathcal{N}, T_\mathcal{N}, F_\mathcal{N}, l_\mathcal{N})$, $\mathcal{O} = (P_\mathcal{O}, T_\mathcal{O}, F_\mathcal{O}, l_\mathcal{O})$ définis sur Σ, Déterminer si $Exec(\mathcal{N}) \equiv_{tr} Exec(\mathcal{O})$ $(\emptyset)$.

Sachant que $RP(\equiv_{tr})$ est ***EXPSPACE***-difficile [9], il suffit de réduire $RP(\equiv_{tr})$ à $CP(\equiv_{tr})$ afin de montrer que $CP(\equiv_{tr})$ est ***EXPSPACE***-difficile. Etant donnés un ensemble fini d'actions Σ et des réseaux de Petri saufs $\mathcal{N} = (P_\mathcal{N}, T_\mathcal{N}, F_\mathcal{N}, l_\mathcal{N})$, $\mathcal{O} = (P_\mathcal{O}, T_\mathcal{O}, F_\mathcal{O}, l_\mathcal{O})$ définis sur Σ, nous devons déterminer si $Exec(\mathcal{N}) \equiv_{tr} Exec(\mathcal{O})$ $(\emptyset)$. L'instance $\rho(\Sigma, \mathcal{N}, \mathcal{O})$ de $CP(\equiv_{tr})$ que nous construisons est donnée par l'ensemble fini d'actions Σ^e, l'ensemble fini de canaux Π, les automates finis $\mathcal{A} = (S_\mathcal{A}, \Delta_\mathcal{A}, s^{in}_\mathcal{A})$ et $\mathcal{B} = (S_\mathcal{B}, \Delta_\mathcal{B}, s^{in}_\mathcal{B})$ définis sur $(\{!, ?\} \times \Pi) \cup \Sigma^e$ et $\Pi' \subseteq \Pi$ définis par

– $\Sigma^e = \Sigma \cup \{a^e_1, a^e_2, a^e_3, a^e_4\}$ où a^e_1, a^e_2, a^e_3 et a^e_4 sont de nouvelles actions,
– $\Pi = \Pi' = P_\mathcal{N} \cup P_\mathcal{O}$,
– $\mathcal{A}$ est l'automate fini de la Figure 15,

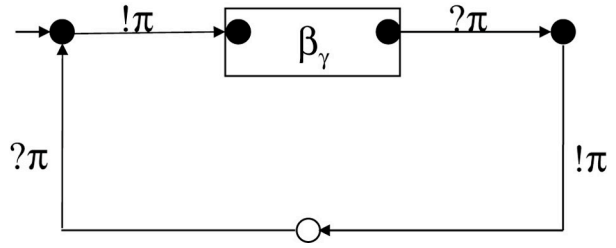

FIG. 13 –

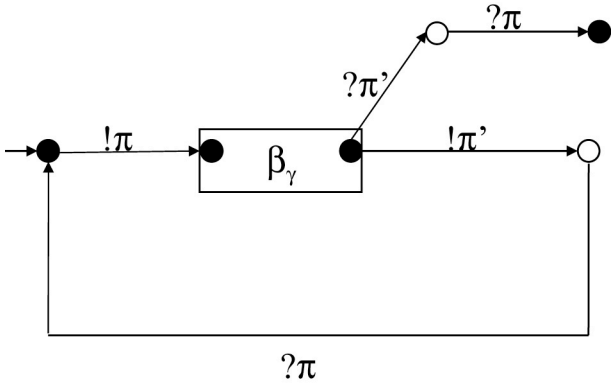

FIG. 14 –

– $\mathcal{B}$ est l'automate fini de la Figure 16.

Pour comprendre comment les parties en forme de fleur de $\mathcal{A}$ et $\mathcal{B}$ sont définies, le lecteur est invité à consulter la Figure 17. Cette figure montre que le tir de la transition a vide les places $P_1, \ldots, P_{|\bullet t|}$ qui sont en entrée de la transition et remplit les places $P'_1, \ldots, P'_{|t\bullet|}$ qui sont en sortie de cette transition. Cette transition est reproduite par les automates $\mathcal{A}$ et $\mathcal{B}$ par la séquence d'actions suivantes : lecture sur les canaux $P_1, \ldots, P_{|\bullet t|}$, correspondant aux places en entrée de la transition, ce qui a pour effet de vider ces canaux, puis la transition a, suivi d'écriture sur les canaux $P'_1, \ldots, P'_{|t\bullet|}$, correspondant aux places en sortie de la transition, ce qui a pour effet de remplir ces canaux. Ainsi le contenu des canaux reflète correctement le contenu des places du réseaux. Les séquences de transitions des automates $\mathcal{A}$ et $\mathcal{B}$ contenant les actions a^e_1, a^e_2, a^e_3 ou a^e_4 servent à garantir qu'aucun médiateur ne peut venir interférer avec la simulation du réseau de Petri. Cela complète la construction. Il est évident que ρ peut être calculé en un espace logarithmique. De plus, $Exec(\mathcal{N}) \equiv_{tr} Exec(\mathcal{O})$ $(\emptyset)$ ssi il existe un automate fini $\mathcal{C} = (S_\mathcal{C}, \Delta_\mathcal{C}, s^{in}_\mathcal{C})$ défini sur $\{!, ?\} \times \Pi$ tel que $Exec(\mathcal{A}) \equiv_{tr} Exec(\mathcal{B} \otimes \mathcal{C})$ $(\{!, ?\} \times \Pi')$. Ainsi, $CP(\equiv_{tr})$ est ***EXPSPACE***-difficile.

⊣

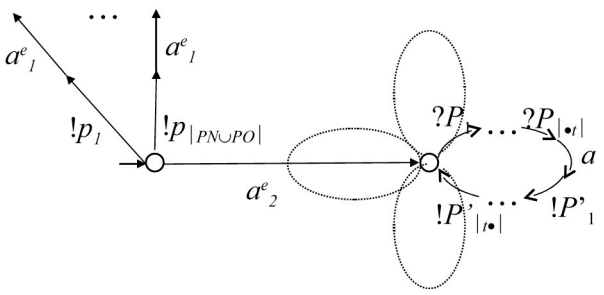

FIG. 15 –

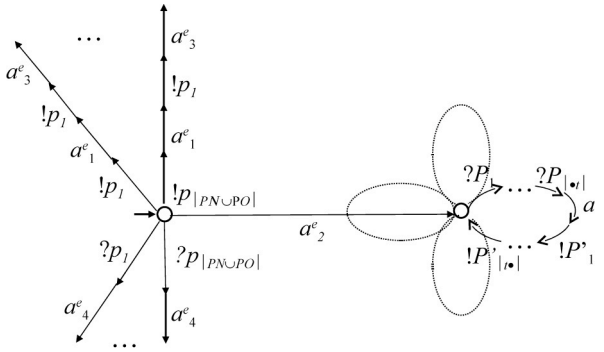

FIG. 16 –

Théorème 3 *Le problème $CP(\leq_{si})$ est EXPTIME-difficile.*

Preuve: Considérons le problème de décision suivant :

Étant donnés un ensemble fini d'actions Σ et des automates finis déterministes $\mathcal{A}, \mathcal{B}_1, \ldots, \mathcal{B}_n$ définis sur Σ, déterminer si $\mathcal{A} \leq_{si} \mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n(\emptyset)$.

Ce problème est ***EXPTIME***-difficile [13]. Il est clair qu'il existe une réduction en espace logarithmique de ce problème au problème $CP(\leq_{si})$. Si on considère les mêmes automates avec $\Pi = \emptyset$ alors $\mathcal{A} \leq_{si} \mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n(\emptyset)$ ssi il existe $\mathcal{C} = (\{s^{in}_\mathcal{C}\}, \Delta_\mathcal{C}, s^{in}_\mathcal{C})$ défini sur $\{!, ?\} \times \emptyset$ tel que $Exec(\mathcal{A}) \leq_{si} Exec(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes \mathcal{C})$ $(\{!, ?\} \times \emptyset)$. ⊣

Théorème 4 *Le problème $CP(\longleftrightarrow_{bi})$ est EXPTIME-difficile.*

Preuve: On utilise la réduction de la preuve du théorème 2. Sachant que le problème de la bisimulation entre deux réseaux de Petri saufs est ***EXPTIME***-difficile [9], on obtient que $CP(\longleftrightarrow_{bi})$ est également ***EXPTIME***-difficile. ⊣

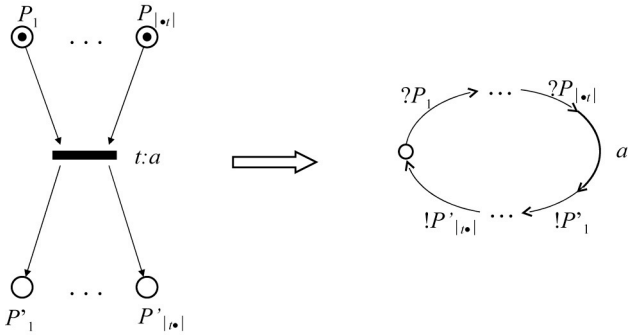

FIG. 17 –

5.2 Classes de complexité

Dans ce qui suit, nous présentons des algorithmes, basés sur les techniques de la synthèse de contrôleurs, pour résoudre les problèmes de la composition $CP(\subseteq_{tr})$, $CP(\equiv_{tr})$, $CP(\leq_{si})$ et $CP(\longleftrightarrow_{bi})$. Nous donnons également leur complexité.

Soit Σ un ensemble fini d'actions et Π un ensemble fini de canaux. Il est commode de considérer un ensemble fini Π° de canaux tel que $(\Sigma \cup \Pi) \cap \Pi^\circ = \emptyset$ et $Card(\Pi) = Card(\Pi^\circ)$ et d'utiliser une bijection $\pi \mapsto \pi^\circ$ de Π vers Π°. Par L°, nous faisons référence à un automate fini $\mathcal{A}' = (S', \Delta', s^{in'})$ défini sur $\{!, ?\} \times \Pi^\circ$ tel que $S' = \{0\}$, Δ' est la fonction définie par $\Delta'(0, !\pi^\circ) = \{0\}$ et $\Delta'(0, ?\pi^\circ) = \{0\}$, et $s^{in'} = 0$. Soit $\mathcal{A} = (S, \Delta, s^{in})$ un automate fini défini sur $(\{!, ?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$. Par $Del^\circ(\mathcal{A})$, nous notons l'automate fini $\mathcal{A}' = (S', \Delta', s^{in'})$ défini sur $(\{!, ?\} \times \Pi) \cup \Sigma$ tel que $S' = S$, Δ' est la fonction définie par

- $\Delta'(s, !\pi) = \Delta(s, !\pi) \cup \Delta(s, !\pi^\circ)$ et $\Delta'(s, ?\pi) = \Delta(s, ?\pi) \cup \Delta(s, ?\pi^\circ)$,
- $\Delta'(s, a) = \Delta(s, a)$,

et $s^{in'} = s^{in}$. Par $Exec^\circ(\mathcal{A})$, nous notons l'automate fini $\mathcal{A}' = (S', \Delta', s^{in'})$ défini sur $(\{!, ?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ de taille exponentielle tel que $S' = S \times 2^\Pi$, Δ' est la fonction définie par

- $(t, Q) \in \Delta'((s, P), !\pi)$ ssi $t \in \Delta(s, !\pi)$, $Q = P \cup \{\pi\}$, $\pi \notin P$ et $(t, Q) \in \Delta'((s, P), ?\pi)$ ssi $t \in \Delta(s, ?\pi)$, $\pi \in P$, $Q = P \setminus \{\pi\}$,
- $(t, Q) \in \Delta'((s, P), !\pi^\circ)$ ssi $t \in \Delta(s, !\pi^\circ)$, $Q = P \cup \{\pi\}$, $\pi \notin P$ et $(t, Q) \in \Delta'((s, P), ?\pi^\circ)$ ssi $t \in \Delta(s, ?\pi^\circ)$, $\pi \in P$, $Q = P \setminus \{\pi\}$,
- $(t, Q) \in \Delta'((s, P), a)$ ssi $t \in \Delta(s, a)$, $Q = P$,

et $s^{in'} = (s^{in}, \emptyset)$. Observons que nous pouvons construire $Exec^\circ(\mathcal{A})$ en un temps exponentiel.

Considérons le problème de décision suivant :

$SC(\approx)$: étant donnés un ensemble fini d'actions Σ, deux ensembles finis de canaux Π et $\Pi' \subseteq \Pi$, des automates finis $\mathcal{A}$, $\mathcal{B}_1, \ldots, \mathcal{B}_n$, déterminer s'il existe un automate fini $\mathcal{C}$ défini sur $(\{!, ?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ qui boucle sur les actions dans $\Sigma \cup (\{!, ?\} \times \Pi)$ et tel que $Exec(\mathcal{A}) \approx Del^\circ(Exec^\circ(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes L^\circ) \times C)(\{!, ?\} \times \Pi')$

Nous avons montré dans l'article [2] que le problème de synthèse de contrôleur $SC(\approx)$ et le problème de la composition $CP(\approx)$ sont équivalents.

Théorème 5 *Le problème $CP(\subseteq_{tr})$ est dans EXPSPACE.*

Preuve: Considérons le problème de décision suivant :

$VP(\subseteq_{tr})$: étant donnés un ensemble fini d'actions Σ, deux ensembles finis de canaux Π et $\Pi' \subseteq \Pi$, des automates finis $\mathcal{A}$, $\mathcal{B}_1, \ldots, \mathcal{B}_n$, déterminer si $Exec(\mathcal{A}) \subseteq_{tr} Del^\circ(Exec^\circ(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes L^\circ))(\{!, ?\} \times \Pi')$

Il est facile de vérifier que le problème $VP(\subseteq_{tr})$ est équivalent au problème $SC(\subseteq_{tr})$. Considérons l'algorithme suivant :

1. Construire l'automate fini $\mathcal{A}'$=$Exec(\mathcal{A})$;
2. Construire l'automate fini $\mathcal{B}$=$Exec^\circ(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes L^\circ)$;
3. Construire l'automate fini $\mathcal{B}' = Del^\circ(\mathcal{B})$
4. Si $\mathcal{A}' \leq_{si} \mathcal{B}'(\{!, ?\} \times \Pi')$ alors retourner $vrai$ sinon retourner $faux$;

Sachant que la taille de $\mathcal{A}'$ est exponentielle par rapport à la taille de Π, que la taille de $\mathcal{B}$ est exponentielle par rapport à la taille de Π et de $\mathcal{B}_1, \ldots, \mathcal{B}_n$, que la taille de $\mathcal{B}'$ est constante par rapport à la taille de $\mathcal{B}$ et qu'un espace polynomial est suffisant pour déterminer si $\mathcal{A}' \leq_{si} \mathcal{B}'(\{!, ?\} \times \Pi')$ [6], il en résulte que $CP(\subseteq_{tr})$ est dans *EXPSPACE.* $\dashv$

Théorème 6 *Le problème $CP(\leq_{si})$ est dans EXPTIME.*

Preuve: L'argument est le même que celui donné pour $CP(\subseteq_{tr})$. Rappelons que le problème suivant est dans P [8] : étant donnés deux ensembles finis d'actions Σ et $\Sigma' \subseteq \Sigma$ et deux automates finis $\mathcal{A}$ et $\mathcal{B}$ définis sur Σ, de plus déterminer si $\mathcal{A} \leq_{si} \mathcal{B}(\Sigma')$. $\dashv$

Théorème 7 *Le problème $CP(\longleftrightarrow_{bi})$ est dans 2-EXPTIME.*

Preuve: Soit Σ un ensemble fini d'actions. Cet ensemble est partitionné en un ensemble d'actions observables Σ_{ob} et un ensemble d'actions non observables Σ_{nob}. L'ensemble Σ est aussi partitionné en un ensemble d'actions contôlables Σ_{ct} et un ensemble d'actions non contrôlables Σ_{nct}. Soit $\mathcal{C}$ un automate fini défini sur Σ. Les contraintes d'observabilité (O) et de contrôlabilité (C) pour l'automate $\mathcal{C}$ sont définies comme suit :

(O) pour tout état s dans $\mathcal{C}$ et pour toute action non observable $a \in \Sigma_{nob}$, s'il existe une transition de s étiquetée par a alors cette transition boucle sur s.

(C) pour tout état s dans $\mathcal{C}$ et pour toute action non contrôlable $a \in \Sigma_{nct}$, il existe une transition de s étiquetée par a.

Considérons le problème de décision suivant :

Étant donnés des ensembles finis d'actions Σ et Σ', des automates finis $\mathcal{A}$ et $\mathcal{G}$ définis sur Σ et des contraintes (O) et (C), déterminer s'il existe un automate fini $\mathcal{C}$ qui satisfait (O) et (C) et tel que $\mathcal{A} \longleftrightarrow_{bi} Del^{\circ}(\mathcal{G} \times C)(\Sigma')$

Ce problème peut être résolu en temps exponentiel [2]. Considérons les entrées du problème $CP(\longleftrightarrow_{bi})$. Soit un ensemble fini d'actions Σ, des ensembles finis de canaux Π et $\Pi' \subseteq \Pi$ et des automates finis $\mathcal{A}, \mathcal{B}_1, \ldots, \mathcal{B}_n$ définis sur $(\{!, ?\} \times \Pi) \cup \Sigma$. Considérons l'algorithme suivant :

1. Construire l'automate fini $\mathcal{A}'$=$Exec(\mathcal{A})$;
2. Construire l'automate fini $\mathcal{B}$=$Exec^{\circ}(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes L^{\circ})$;
3. Utiliser une procédure pour déterminer s'il existe $\mathcal{C}$ défini sur $(\{!, ?\} \times (\Pi \cup \Pi^{\circ})) \cup \Sigma$ qui boucle sur $(\{!, ?\} \times \Pi) \cup \Sigma$ et tel que $\mathcal{A} \longleftrightarrow_{bi} Del^{\circ}(\mathcal{B} \times C)(\{!, ?\} \times \Pi')$;

Cet algorithme est dans *2-EXPTIME*. ⊣

Concernant le problème $CP(\equiv_{tr})$, nous n'avons pas d'algorithme pour le résoudre. Dans la littérature il existe des résultats concernant la difficulté algorithmique d'un problème similaire [17]. Il existe également des algorithme pour résoudre le problème $SC(\equiv_{tr})$ sans la fonction Del° [16]. Cependant, à notre connaissance il n'existe pas d'algorithme pour résoudre le problème $SC(\equiv_{tr})$.

6 Conclusion et travaux futurs

Nous avons présenté un modèle dans lequel les agents sont représentés par des automates finis qui peuvent envoyer et recevoir des messages.

Problème	Difficulté algorithmique	Complexité
$CP(\subseteq_{tr})$	*EXPSPACE*	*EXPSPACE*
$CP(\equiv_{tr})$	*EXPSPACE*	?
$CP(\leq_{si})$	*EXPTIME*	*EXPTIME*
$CP(\longleftrightarrow_{bi})$	*EXPTIME*	*2-EXPTIME*

TAB. 1 – Tableau récapitulatif

Nous avons défini le problème de la composition pour quatre types d'équivalence, l'inclusion de traces, l'équivalence de traces, la simulation et la bisimulation. Les résultats de la complexité des problèmes sont récapitulés dans le tableau 1. Concernant le problème de la composition en considérant l'inclusion de trace et la simulation, nous avons donné la difficulté algorithmique ainsi que la classe de complexité du problème. Plus précisément, nous avons montré que le problème de la composition est *EXPSPACE*-complet pour l'inclusion de traces et *EXPTIME*-complet pour la simulation. Nous avons également prouvé que le problème de la composition est *EXPSPACE*-difficile pour l'équivalence de trace et *EXPTIME*-difficile pour la bisimulation. De plus nous avons proposé un algorithme qui résout le problème de la composition pour la bisimulation en un temps doublement exponentiel par rapport à la taille du problème. Les algorithmes proposés dans cet article nécessitent tous de calculer explicitement la composition de l'ensemble des agents, ce qui est une étape particulièrement coûteuse lorsque le nombre d'agents est élevé. Cependant, les réductions en espace logarithmique proposées dans la section 5.1 montrent qu'il n'existe pas d'algorithme de moindre complexité permettant de résoudre le problème, du moins dans le cas de la simulation et de l'inclusion de traces. Plusieurs approches sont possibles en vue d'obtenir une complexité moindre. La première consiste à proposer une procédure qui sélectionne les agents pertinents par rapport au but à atteindre, soit via une heuristique, soit en donnant a priori des spécifications sur les agents. Une autre approche consiste à restreindre les comportements des agents. D'un autre côté, le modèle proposé dans cet article est déjà restrictif au niveau des communications. Il est possible de l'enrichir en autorisant des canaux de communication qui contiennent des files de messages de longueurs bornées, ce qui reviendrait dans notre formalisme à remplacer un canal par un nombre suffisant de canaux pour simuler la file de messages. Nous savons par contre qu'il n'est pas possible de considérer des files de messages de longueur non bornées ; en effet, la preuve du

théorème 2 serait valide pour des réseaux de Petri généraux pour lequel le problème est indécidable. Pour enrichir le processus de communication, il conviendrait alors de considérer par exemple des messages qui sont décrits par des termes de logique du premier ordre qui indiqueraient le type du message échangé.

7 Remerciements

Pour la préparation de cet article nous avons bénéficié d'un soutien financier dans le cadre des projets COPS (ANR) et ROSACE (RTRA STAE).

Références

[1] A. Arnold, A. Vincent, I. Walukiewicz, 'Games for synthesis of controllers with partial observation'. Theoretical computer science **303** (2003) 7–34.

[2] P. Balbiani, F. Cheikh, G. Feuillade, 'Composition of interactive Web services based on controller synthesis'. In : *2nd International Workshop on Web Service Composition and Adaptation*. IEEE (2008) 521–528.

[3] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, M. Mecella, 'Automated composition of transition-based semantic Web services with messaging'. In : *Proceedings of the 31st International Conference on Very Large Data Bases.* VLDB Endowment (2005) 613–624.

[4] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella, 'Automatic composition of e-services that export their behavior'. In : *Service-Oriented Computing — ICSOC 2003.* Springer (2003) 43–58.

[5] F. Cheikh, G. De Giacomo, M. Mecella, 'Automatic Web services composition in trust-aware communities'. In : *Proceedings of the 3rd ACM Workshop on Secure Web Services.* Association for Computing Machinery (2006) 43–52.

[6] M. Garey, D. Johnson, 'Computers and Intractability : A Guide to the Theory of NP-Completeness'. H. Freeman (1979).

[7] M. Huhns, 'Agents as Web services'. IEEE Internet Computing **6** (2002) 93–95.

[8] H. Hüttel, S. Shukla. 'On the Complexity of Deciding Behavioural Equivalences and Preorders, A Survey'. Rapport dans la série BRICS (1996).

[9] L. Jategaonkar, A. Meyer, 'Deciding true concurrency equivalences on safe, finite nets'. Theoretical computer science **154** (1996) 107–143.

[10] T. Melliti, S. Haddad, A. Suna, A. El Fallah Seghrouchni, '$Web - MASI$: multi-agent systems interoperability using a Web services based approach'. In : *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology 2005.* IEEE (2005) 739–742.

[11] A. Meyer, L. Stockmeyer, 'The equivalence problem for regular expressions with squaring requires exponential space'. In : *Proceedings of the Annual Symposium on Switching and Automata Theory.* IEEE (1972) 125–129.

[12] N. Milanovic, M. Malek, 'Current solutions for Web service composition'. IEEE Internet Computing **8** (2004) 51–59.

[13] A. Muscholl, I. Walukiewicz, 'A Lower bound on Web services composition'. In : *Proceedings of the International Conference on Foundations of Software Science and Computation Structures.* Springer (2007) 274–286.

[14] M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, P. Traverso, 'Planning and monitoring Web service composition'. In : *Artificial Intelligence : Methodology, Systems, and Applications.* Springer (2004) 106–115.

[15] M. Pistore, A. Marconi, P. Bertoli, P. Traverso, 'Automated composition of Web services by planning at the knowledge level'. In : *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence.* International Joint Conferences on Artificial Intelligence (2005) 1252–1259.

[16] P. Ramadge, Wonham, 'The control of descrete event systems'. Proceedings of the IEEE **77** (1989) 81–98.

[17] J. Tsitsiklis, 'On the control of discrete event dynamical systems'. Mathematics of Control, Signals and Systems **2**(1989) 95–107.

[18] P. Traverso, M. Pistore, 'Automated composition of semantic Web services into executable processes'. In : *The Semantic Web — ISWC 2004.* Springer (2004) 380–394.