Considérations relatives à la décidabilité et à la complexité du problème de la composition de services

Philippe.Balbiani ⋆ Fahima. Cheikh⋆ Guillaume. Feuillade⋆

⋆Université Paul Sabatier, Institut de recherche en informatique de Toulouse
118, route de Narbonne 31062 Toulouse Cedex 9

Résumé :
Le problème de la composition de services consiste à combiner des services afin de répondre à la requête d'un client. Dans cet article, nous considérons un modèle orienté service dans lequel les services sont capables de mettre à jour un système d'informations et d'échanger des messages. Dans ce modèle, le problème de la composition est indécidable. Pour cette raison, nous considérons un modèle simplifié pour lequel le problème de la composition de services est décidable.

Mots-clés : Composition de services, automates conditionnels, complexité.

Abstract:
Services composition problem consists in combining services in order to answer a client request. In this paper, we consider a service oriented model where services are able to update an information system and to exchange messages. In this model the composition problem is undecidable. For this reason, we consider a simplified model for which the composition problem is decidable.

Keywords: Services composition, conditional automata, complexity.

1 Introduction

Les applications orientées services [11] sont à l'origine d'un nouveau paradigme de programmation distribuée qui modifie la façon dont les applications sont spécifiées, implémentées et exécutées. Toutefois, avant que les services ne deviennent une réalité, un certain nombre de défis comme la sécurité des services et la composition des services, doivent être relevés. Avant d'accorder l'accès aux ressources dont ils ont la responsabilité, les services établissent leurs politiques de sécurité et décrivent les conditions sous lesquelles telle ou telle ressource peut être légalement utilisée. Par conséquent, les services interagissent avec leurs clients et avec d'autres services par le biais de protocoles cryptographiques afin d'obtenir leurs certificats et de caractériser leurs droits. La sécurité des services traite de la confidentialité, de l'intégrité et de la disponibilité en rapport avec la problématique de la combinaison et de l'intégration des protocoles et des politiques. Les services permettent de réaliser des parcs d'organisations capables d'exporter leurs services à des clients et de coopérer en composant des services via les réseaux. Par suite, les services sont des éléments logiciels indépendants qui peuvent être composés en vue de faire collaborer entre elles des applications distribuées. La composition des services étudie les situations où les demandes des clients ne peuvent être satisfaites qu'en combinant les services disponibles de manière appropriée [2, 10]. Il y a cinq sections principales dans cet article. Dans la section 2, nous présenterons brièvement un modèle de services et nous élaborerons un modèle formel de la composition. Le problème de la composition évoqué dans la section 2 étant indécidable, nous présenterons, dans la section 3, un modèle simplifié de services basé sur les automates finis. Dans la section 4, nous définirons les concepts qui nous permettront de comparer les services entre eux : équivalence de services (équivalence de trace et bisimulation) et inclusion de services (inclusion de trace et simulation). Nous attaquerons, dans les sections 5 et 6, l'étude de la complexité algorithmique du problème de l'équivalence de services et du problème de l'inclusion de services.

2 Modèle général

Nous présentons un modèle d'applications orientées services (voir la figure 1) qui reprend et généralise le modèle élaboré par Berardi et al [3]. Ensuite, nous définirons le problème de la composition. Notre modèle est constitué des éléments suivants : un système d'informations IF, une communauté de services $\mathcal{C} = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$, un service but $\mathcal{S}_{but}$, un service client $\mathcal{S}_0$ muni d'un ensemble de certificats et un service médiateur S_{med} qui s'interpose entre le client et les services de la communauté. Un *système d'informations* IF peut être vu comme un ensemble d'objets (produits manufacturés, fichiers, etc.) caractérisés en termes d'attributs tels que le prix, la taille, etc. Les *services* d'une communauté $\mathcal{C}$ mettent à jour le système d'informations IF en exécutant des commandes. Ils obtiennent des informations sur IF au moyen d'échanges de messages avec les autres services. Les services sont représentés par des automates conditionnels dans lesquels la transition d'un état à un autre n'est possible que si une certaine condition est vérifiée. Les conditions peuvent concerner la valeur d'une variable locale au service, la valeur d'un attribut pour un objet donné dans IF ou la valeur d'un des certificats du client. Ici, *les certificats* sont des assertions sur les clients des services. Ils sont émis par les services de la communauté. Les services de la communauté, tels que nous les avons définis, seront utilisés par d'autres services appelés *services clients* (notés $\mathcal{S}_0$). Leur objectif est d'obtenir des informations sur le système d'informations. Deux états suffisent pour les définir complètement. A partir de ces états, un service client ne peut qu'émettre ou recevoir des messages. Les conditions des transitions du client sont toujours vérifiées. La requête d'un utilisateur est représentée par un service $\mathcal{S}_{but}$ appelé *service but* et qui ne fait pas partie de la communauté. Les services médiateurs, notés $\mathcal{S}_{med}$, effectuent uniquement des échanges de messages. Leur rôle est de s'interposer entre le service client et les services de la communauté. Les conditions des transitions dans $\mathcal{S}_{med}$ ne concernent que les valeurs de ses variables locales. Lorsqu'un client veut effectuer des calculs à partir d'un système d'informations et qu'aucun des services de la communauté ne peut réaliser seul ces calculs, une solution est de combiner entre eux les services de la communauté. Le problème de la composition des services consiste alors à lier ces services entre eux. Formellement, le problème de la composition de services est le problème de décision suivant : *soient une communauté* $\mathcal{C} = \{\mathcal{S}_1, ..., \mathcal{S}_n\}$, *un ensemble de certificats* $\mathcal{C}ert$, *un service client* $\mathcal{S}_0$ *et un service but* $\mathcal{S}_{but}$, *existe-t-il un service médiateur* $\mathcal{S}_{med}$ *tel que pour tout système d'informations* IF, *le comportement de* $\{\mathcal{S}_0,\mathcal{S}_{but}\}$ *est équivalent à celui de* $\{\mathcal{S}_0, \mathcal{S}_{med}\} \cup \mathcal{C}$. Dans cette définition, l'équivalence entre $\{\mathcal{S}_0,\mathcal{S}_{but}\}$ et $\{\mathcal{S}_0, \mathcal{S}_{med}\} \cup \mathcal{C}$ est basée sur la bisimulation, la simulation, l'équivalence de trace ou l'inclusion de trace. Ainsi défini, le problème de la composition de ser-

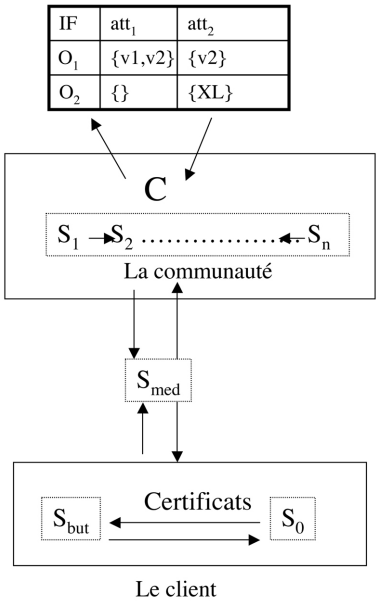

FIG. 1 – Modèle d'application orientées services

vices est indécidable. En effet, il nous a été possible [1] de réduire le problème de l'arrêt des machines de Minsky [7] au problème de la composition défini ci-dessus.

3 Modèle simplifié

Notre modèle simplifié des services est basé sur les automates finis.

3.1 Automates finis

Un automate fini est une structure de la forme $\mathcal{A} = (Q, \Sigma, \rightarrow, q_0)$, dans laquelle Q est un ensemble fini d'états, Σ est un ensemble fini de symboles, $\rightarrow \subseteq Q \times \Sigma \times Q$ est une relation de transition et $q_0 \in Q$ est un état. Pour tout $q \in Q$, pour tout $a \in \Sigma$ et pour tout $q' \in Q$, si $(q, a, q') \in \rightarrow$ alors nous écrivons $q \rightarrow_a q'$. Nous dirons que $\mathcal{A}$ est déterministe lorsque pour tout $q \in Q$ et pour tout $a \in \Sigma$, il existe au plus un $q' \in Q$ tel que $q \rightarrow_a q'$. Un chemin pour $\mathcal{A}$ est une suite finie de la forme (q_0, a_1, q_1), (q_1, a_2, q_2),..., (q_{n-1}, a_n, q_n), telle que pour tout $i \in \{1, \ldots, n\}$,$q_{i-1} \rightarrow_{a_i} q_i$. Le mot $a_1 \ldots a_n$ est sa trace. L'ensemble des traces de tous les chemins pour $\mathcal{A}$ est noté $Tr(\mathcal{A})$.

3.2 Automates conditionnels

Nous présentons maintenant notre modèle simplifié des services. Ce modèle simplifié est celui des automates conditionnels et semble n'avoir jamais fait l'objet d'aucune recherche. Soit At un ensemble de formules atomiques. L'ensemble des littéraux sur At est défini par $Li(At) = At \cup \{\neg p : p \in At\}$. Nous dirons d'une partie I de $Li(At)$ qu'elle est *maximale consistante* lorsque pour toute formule atomique $p \in At$ on a $p \in I$ et $\neg p \notin I$, ou $\neg p \in I$ et $p \notin I$. Un automate conditionnel est une structure de la forme $\mathcal{A} = (Q, At, Ac, \delta, q_0, I_0)$, dans laquelle Q est un ensemble fini d'états, At est un ensemble fini de formules atomiques, Ac est un ensemble fini d'actions, $\delta : Q \times Ac \times Q \longrightarrow 2^{2^{Li(At)} \times 2^{Li(At)}}$ est une fonction de transition, $q_0 \in Q$ est un état et $I_0 \subseteq Li(At)$ est une partie maximale consistante de $Li(At)$. Pour tout $q \in Q$, et pour tout $a \in Ac$ et pour tout $q' \in Q$, $\delta(q, a, q')$ décrit l'ensemble possible des causes et des effets de l'exécution de l'action a entre les états q et q'. L'appartenance d'un couple (I, I') d'ensemble de littéraux sur At à $\delta(q, a, q')$ signifie que I est l'ensemble des préconditions et I' est l'ensemble des postconditions pour l'exécution de l'action a entre les états q et q'. Ces préconditions et postconditions correspondent aux certificats et à leur évolution dont nous avons parlé dans la section 2. A chaque automate conditionnel $\mathcal{A} = (Q, At, Ac, \delta, q_0, I_0)$, nous associons l'automate fini $AF(\mathcal{A}) = (Q', \Sigma', \rightarrow', q'_0)$ défini par : $Q' = \{(q, I) : q \in Q$ et $I \subseteq Li(At)$ est une partie maximale consistante de $Li(At)\}$, $\Sigma' = Ac$, $\rightarrow' \subseteq Q' \times \Sigma' \times Q'$ est la relation de transition définie par $(q, I) \rightarrow'_{a'} (q', I')$ ssi il existe $(J, J') \in \delta(q, a', q')$ tel que $J \subseteq I$ et $I' = (I \setminus \neg J') \cup J'$ où $\neg J' = \{p : \neg p \in J'\} \cup \{\neg p : p \in J'\}$ et $q'_0 = (q_0, I_0)$. Nous observons qu'un temps exponentiel par rapport à la taille de l'automate conditionnel $\mathcal{A}$ est suffisant pour construire l'automate fini $AF(\mathcal{A})$. Nous observons également que la taille de l'automate fini $AF(A)$ est exponentielle par rapport à la taille de l'automate conditionnel A.

3.3 Produits d'automates

L'analyse de la complexité algorithmique du problème de la composition de services nécessitera, dans les sections 5 et 6, l'utilisation du produit d'automates finis. Soit $n \geq 2$. Pour tout $i = 1, \ldots, n$, soit $\mathcal{A}_i = (Q_i, \Sigma_i, \rightarrow_i, q_{0i})$ un automate fini déterministe. Le *produit asynchrone des automates finis* $\mathcal{A}_1, \ldots, \mathcal{A}_n$, noté $\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n$ est l'automate fini $\mathcal{A}' = (Q', \Sigma', \rightarrow', q'_0)$ défini par : $Q' = Q_1 \times \ldots \times Q_n$, $\Sigma' =$

$\Sigma_1 \cup \ldots \cup \Sigma_n$, $q'_0 = (q_{01}, \ldots, q_{0n})$ et $\rightarrow' \subseteq Q' \times \Sigma \times Q'$ est la relation de transition définie par : pour tout $q = (q_1, \ldots, q_n) \in Q'$, pour tout $a \in \Sigma'$ et pour tout $q' = (q'_1, \ldots, q'_n) \in Q'$, $q \rightarrow'_a q'$ ssi il existe $i \in \{1, \ldots, n\}$ tel que $a \in \Sigma_i$, $q_i \rightarrow_a q'_i$ et pour tout $j \in \{1, \ldots, n\}$, si $j \neq i$ alors $q_j = q'_j$. Soit $n \geq 2$. Pour tout $i = 1, \ldots, n$, soit $\mathcal{A}_i = (Q_i, At_i, Ac_i, \delta_i, q_{0i}, I_{0i})$ un automate conditionnel. Le *produit asynchrone des automates conditionnels* $\mathcal{A}_1, \ldots, \mathcal{A}_n$, noté $\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n$ est l'automate conditionnel $\mathcal{A} = (Q, At, Ac, \delta, q_0, I_0)$ défini par : $Q = Q_1 \times \ldots \times Q_n$, $At = At_1 \cup \ldots \cup At_n$, $Ac = Ac_1 \cup \ldots \cup Ac_n$, $q_0 = (q_{01}, \ldots, q_{0n})$, $I_0 = I_{01} \cup \ldots \cup I_{0n}$ et $\delta : Q \times Ac \times Q \rightarrow 2^{2^{Li(At)} \times 2^{Li(At)}}$ est la fonction de transition définie par : pour tout $q = (q_1, \ldots, q_n) \in Q$, pour tout $a \in Ac$, pour tout $q' = (q'_1, \ldots, q'_n) \in Q$, pour tout $J \subseteq Li(At)$ et pour tout $J' \subseteq Li(At)$, $(J, J') \in \delta(q, a, q')$ ssi il existe $i \in \{1, \ldots, n\}$ tel que $a \in Ac_i$, $(J, J') \in \delta_i(q_i, a, q'_i)$ et pour tout $j \in \{1, \ldots, n\}$, si $j \neq i$ alors $q_j = q'_j$. Nous observons que le produit $\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n$ n'est défini que si $I_{01} \cup \ldots \cup I_{0n}$ constitue un ensemble consistant de littéraux. **Exemple** Considérons l'automate conditionnel $\mathcal{A}_1$ représenté par la figure 2 et l'automate conditionnel $\mathcal{A}_2$ représenté par la figure 3. L'automate conditionel $\mathcal{A}_1 \otimes \mathcal{A}_2$ est représenté par la figure 4.

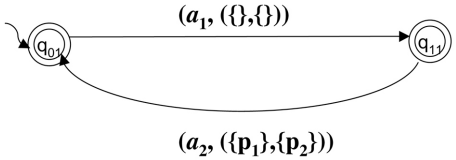

FIG. 2 – Le service $\mathcal{A}_1$

4 Equivalences et préordres

Afin de pouvoir comparer entre eux les automates conditionnels et leurs produits, nous devons définir ce que sont l'inclusion de trace, l'équivalence de trace, la simulation et la bisimulation pour les automates finis. Soit $\mathcal{A} = (Q, \Sigma, \rightarrow, q_0)$ et $\mathcal{A}' = (Q', \Sigma', \rightarrow', q'_0)$ des automates finis. Nous dirons que $\mathcal{A}$ est *inclus pour la trace* dans $\mathcal{A}'$, noté $\mathcal{A} \leq_{tr} \mathcal{A}'$, lorsque $Tr(\mathcal{A}) \subseteq Tr(\mathcal{A}')$. Nous dirons que $\mathcal{A}$ et $\mathcal{A}'$ sont *équivalents pour la trace*, noté $\mathcal{A} \equiv_{tr} \mathcal{A}'$, lorsque $Tr(\mathcal{A}) = Tr(\mathcal{A}')$. Une *simulation* de $\mathcal{A}$ par $\mathcal{A}'$ est une relation binaire $Z \subseteq Q \times Q'$ telle que $q_0 Z q'_0$ et pour tout $q \in Q$ et pour tout $q' \in Q'$, si qZq' alors pour tout $r \in Q$ et pour tout $a \in \Sigma$, si $q \rightarrow_a r$ alors il existe $r' \in Q'$ tel que rZr' et $q' \rightarrow_a r'$. Nous dirons que $\mathcal{A}$ est *simulé* par $\mathcal{A}'$, noté $\mathcal{A} \leq_{si} \mathcal{A}'$, lorsqu'il existe une relation binaire $Z \subseteq Q \times Q'$ telle que Z est une simulation de $\mathcal{A}$ par $\mathcal{A}'$. Nous dirons que $\mathcal{A}$ et $\mathcal{A}'$ sont *bisimilaires*, noté $\mathcal{A} \equiv_{bi} \mathcal{A}'$, lorsqu'il existe une relation binaire $Z \subseteq Q \times Q'$ telle que Z est une simulation de $\mathcal{A}$ par $\mathcal{A}'$ et Z^{-1} est une simulation de $\mathcal{A}'$ par $\mathcal{A}$. La relation bi-

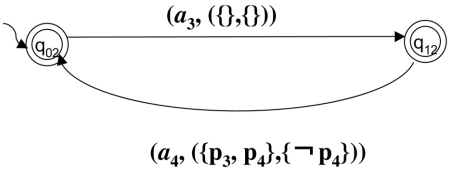

FIG. 3 – Le service $\mathcal{A}_2$

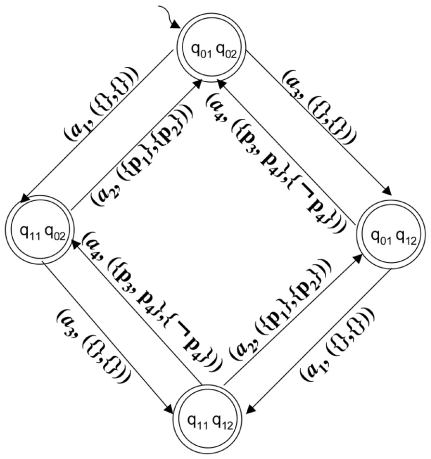

FIG. 4 – Le service $\mathcal{A}_1 \otimes \mathcal{A}_2$

naire $Z \subseteq Q \times Q'$ sera alors appelée bisimulation entre $\mathcal{A}$ et $\mathcal{A}'$. Rappelons qu'il existe des automates finis $\mathcal{A}$ et $\mathcal{A}'$ tels que $\mathcal{A} \leq_{si} \mathcal{A}'$, $\mathcal{A}' \leq_{si} \mathcal{A}$ et non $\mathcal{A} \equiv_{bi} \mathcal{A}'$ [5]. Nous dirons que $\mathcal{A}$ et $\mathcal{A}'$ sont *isomorphes* ssi $\Sigma = \Sigma'$ et il existe une bijection $g : Q \rightarrow Q'$ telle que pour tout état q_1, $q_2 \in Q$ et pour toute action $a \in \Sigma$, $q_1 \rightarrow_a q_2$ ssi $g(q_1) \rightarrow'_a g(q_2)$. Soit $\mathcal{A}$ et $\mathcal{A}'$ des automates conditionnels. Nous dirons que $\mathcal{A}$ est inclus pour la trace dans (resp. simulé par) $\mathcal{A}'$ lorsque $AF(\mathcal{A})$ est inclus pour la trace dans (resp. simulé par) $AF(\mathcal{A}')$. Nous dirons que $\mathcal{A}$ et $\mathcal{A}'$ sont équivalents pour la trace (resp. bisimilaires) lorsque $AF(\mathcal{A})$ et $AF(\mathcal{A}')$ sont équivalents pour la trace (resp. bisimilaires). Remarquons que l'équivalence de trace et la bisimilarité sont des relations d'équivalence tandis que l'inclusion de trace et la similarité sont des relations de préordre.

5 Réductions polynomiales

Nous analysons maintenant la difficulté intrinsèque qu'il y a à comparer entre eux les automates conditionnels ou les produits d'automates conditionnels, le problème de la comparaison de produits d'automates conditionnels étant celui que nous considérons comme étant le plus proche du problème de la composition des services évoqué dans la section 2. Dans cette section, nous caractérisons les bornes inférieures de complexité. Nous allons d'abord montrer que les problèmes suivants : (P^{ac}_{bi}) *: soient deux automates conditionnels $\mathcal{A}$ et $\mathcal{A}'$, déterminer si $\mathcal{A}$ et $\mathcal{A}'$ sont bisimilaires,* (P^{ac}_{et}) *: soient deux automates conditionnels $\mathcal{A}$ et $\mathcal{A}'$, déterminer si $\mathcal{A}$ et $\mathcal{A}'$ sont équivalents pour la trace* et (P^{ac}_{it}) *: soient deux automates conditionnels $\mathcal{A}$ et $\mathcal{A}'$, déterminer si $\mathcal{A}$ est inclus pour la trace dans $\mathcal{A}'$.* sont respectivement EXPTIME-difficile, EXSPACE-difficile et EXSPACE-difficile. Pour cela, nous allons réduire le problème de la bisimulation entre des réseaux de Petri saufs, qui est EXPTIME-difficile [6], au problème (P^{ac}_{bi}) et nous allons réduire le problème de l'équivalence de trace (resp. inclusion de trace) des réseaux de Petri saufs, qui est EXSPACE-difficile [6], au problème (P^{ac}_{et}) (resp. (P^{ac}_{it})). Un *réseau de Petri sauf* [8] est une structure de la forme $\mathcal{N} = (P, T, F, \Sigma, l, m_0)$, dans laquelle $P = \{p_1, \ldots, p_n\}$ est un ensemble fini de places, $T = \{t_1, \ldots, t_m\}$ est un ensemble fini de transitions, $F \subseteq (P \times T) \cup (T \times P)$ est un ensemble fini d'arcs, Σ est un ensemble fini de symboles, $l : T \rightarrow \Sigma$ est une fonction et $m_0 : P \rightarrow \{0, 1\}$ est une fonction. La fonction m_0 est appelée marquage initial de $\mathcal{N}$. De façon générale, un marquage $m : P \rightarrow \{0, 1\}$ pour $\mathcal{N}$ décrit une répartition de jetons dans les places. Pour chaque transition $t \in T$, nous définissons deux sous-ensembles de places : ${}^\bullet t = \{p : p \in P \text{ et } (p, t) \in F\}$ et $t^\bullet = \{p : p \in P \text{ et } (t, p) \in F\}$. L'ensemble ${}^\bullet t$ contient les places qui ont un arc en direction de t tandis que l'ensemble $t^\bullet$ contient les places qui ont un arc en provenance de t. Une transition $t \in T$ est dite *active* (resp. *inactive*) au marquage m si pour toute place $p \in {}^\bullet t$, $m(p) = 1$ (resp. il existe une place $p \in {}^\bullet t$ telle que $m(p) = 0$). Lorsque la transition t est active au marquage m, nous définissons le *tir* de la transition t par l'action qui modifie le marquage m en un marquage m' défini pour toute place $p \in P$ par

– $m'(p) = \begin{cases} m(p) & \text{si } p \notin {}^\bullet t \cup t^\bullet \\ 1 & \text{si } p \in t^\bullet \\ 0 & \text{sinon} \end{cases}$

Nous notons $m[t\rangle$ lorsque la transition t est active au marquage m et $m[t\rangle m'$ lorsque m' est le résultat du tir de t depuis m. A chaque réseau de Petri sauf $\mathcal{N} = (P, T, F, \Sigma, l, m_0)$ nous associons l'automate fini $AF(\mathcal{N}) = (Q', \Sigma', \rightarrow', q'_0)$ défini par $Q' = M$ où M est l'ensemble des marquages pour $\mathcal{N}$, Σ'=Σ, $\rightarrow' \subseteq Q' \times \Sigma' \times Q'$ est la relation de transition définie par $m \rightarrow'_a m'$ ssi il existe une transition $t \in T$ tel que $l(t) = a$ et $m[t\rangle m'$ et $q'_0 = m_0$. Nous définissons, entre réseaux de Petri saufs, les mêmes relations d'équi-

valence et de préordre que celles considérées dans la section 4 :

– $\mathcal{N} \leq_{tr} \mathcal{N}'$ ssi $AF(\mathcal{N}) \leq_{tr} AF(\mathcal{N}')$,
– $\mathcal{N} \equiv_{tr} \mathcal{N}'$ ssi $AF(\mathcal{N}) \equiv_{tr} AF(\mathcal{N}')$,
– $\mathcal{N} \leq_{si} \mathcal{N}'$ ssi $AF(\mathcal{N}) \leq_{si} AF(\mathcal{N}')$ et
– $\mathcal{N} \equiv_{bi} \mathcal{N}'$ ssi $AF(\mathcal{N}) \equiv_{bi} AF(\mathcal{N}')$.

A chaque réseau de Petri sauf $\mathcal{N} = (P, T, F, \Sigma, l, m_0)$, nous associons l'automate conditionnel $R(\mathcal{N}) = (Q, At, Ac, \delta, q_0, I_0)$, défini par : $Q = \{q\}$, $At = P$, $Ac = \Sigma$, $\delta(q, a, q) = \{(h(m), h(m')) : m[t\rangle m'$ et $l(t) = a\}$ $q_0 = q$ et $I_0 = h(m_0)$. Dans la définition ci-dessus, $h : M \rightarrow 2^{Li(At)}$ est la fonction qui associe à chaque marquage $m \in M$ le sous-ensemble $h(m) = \{p : m(p) = 1\} \cup \{\neg p : m(p) = 0\}$ de $Li(At)$. Notons que pour tout marquage $m \in M$, $h(m)$ est une partie maximale consistante de $Li(At)$. Nous allons montrer que : (1) R est calculable par une machine de Turing déterministe en utilisant un espace logarithmique, (2) les automates finis $AF(\mathcal{N})$ et $AF(R(\mathcal{N}))$ sont isomorphes. Concernant (1), étant donné un réseau de Petri sauf $\mathcal{N}$, une machine de Turing $\mathcal{M}$ calcule $R(\mathcal{N})$ de la façon suivante : (a) écrire l'ensemble $\{q\}$ contenant l'unique état de $R(\mathcal{N})$. Ensuite, écrire l'ensemble $At = P$ des formules atomique de $R(\mathcal{N})$, l'ensemble $Ac = \Sigma$ des actions de $R(\mathcal{N})$ et l'état initial $q_0 = q$ de $R(\mathcal{N})$; (b) écrire la partie maximale consistante $I_0 = h(m_0)$ de $Li(At)$. Pour chaque place p, lire la valeur de $m_0(p)$ et ajouter p ou $\neg p$ à I_0 selon que cette valeur est 1 ou 0 ; (c) pour finir, écrire la fonction de transition δ de $R(\mathcal{N})$. Pour ce faire, utiliser deux compteurs i et j. Le compteur i va successivement prendre comme valeurs les transitions de $\mathcal{N}$. Pour chaque valeur de i écrire l'ensemble $\delta(q, a, q)$ où a est l'action correspondant à la transition i via la fonction l. Pour ce faire, donner successivement à j comme valeurs les places de $\mathcal{N}$. Toutes ces étapes peuvent bien sûr être faites de façon déterministe en utilisant un espace logarithmique. Concernant (2), nous procédons de la façon suivante. Les états de $AF(\mathcal{N})$ sont des marquages pour $\mathcal{N}$. Les états de $AF(R(\mathcal{N}))$ sont des couples de la forme (q, I) où q est l'unique état de $R(\mathcal{N})$ et I est une partie maximale consistante de $Li(At)$. Soit g la fonction qui associe à chaque marquage m pour $\mathcal{N}$ le couple $(q, h(m))$. Nous laissons le soin au lecteur de vérifier que g est une bijection telle que pour tout marquage m_1, m_2 pour $\mathcal{N}$ et pour toute action $a \in \Sigma$, $m_1 \rightarrow_a m_2$ (dans $AF(\mathcal{N})$) ssi $g(m_1) \rightarrow_a g(m_2)$ (dans $AF(R(\mathcal{N}))$). Donc :

Théorème 1 *(P^{ac}_{bi}) est EXPTIME-difficile et (P^{ac}_{et}) et (P^{ac}_{it}) sont EXPSPACE-difficiles.*

Démonstration. Il suffit de rappeler que le problème de la bisimulation entre réseaux de Petri saufs est EXPTIME-difficile [6] et que les problèmes de l'équivalence de trace et de l'inclusion de trace entre réseaux de Petri saufs sont EXPSPACE-difficiles [6]. Nous allons ensuite montrer que le problème suivant : (P^{ac}_{si}) *: soient deux automates conditionnels $\mathcal{A}$ et $\mathcal{A}'$, déterminer si $\mathcal{A}$ est simulé par $\mathcal{A}'$*, est EXPTIME-difficile. Pour cela, nous allons réduire le problème, (P^{afd}_{si}) *: soient $\mathcal{A}$, $\mathcal{B}_1, \ldots, \mathcal{B}_n$ des automates finis déterministes, déterminer si $\mathcal{A}$ est simulé par $\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n$*, qui est EXPTIME-difficile [9], au problème (P^{ac}_{si}). Soit $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma_{\mathcal{A}}, \rightarrow_{\mathcal{A}}, q_{0\mathcal{A}})$ un automate fini déterministe. Considérons l'automate conditionnel $R(\mathcal{A}) = (Q, At, Ac, \delta, q_0, I_0)$ tel que : $Q = Q_{\mathcal{A}}$, $At = \emptyset$, $Ac = \Sigma_A$, $\delta(q, a, q') = \{(\emptyset, \emptyset) : q \rightarrow^a_{\mathcal{A}} q'\}$, $q_0 = q_{0\mathcal{A}}$ et $I_0 = \emptyset$. Soit $n \geq 2$. Pour tout $i \in \{1, \ldots, n\}$, soit $\mathcal{B}_i = (Q_i, \Sigma_i, \rightarrow_i, q_{0i})$ un automate fini déterministe et $\mathcal{B}' = (Q_{\mathcal{B}'}, \Sigma_{\mathcal{B}'}, \rightarrow_{\mathcal{B}'}, q_{0\mathcal{B}'})$ leur produit asynchrone. Sans perte de généralité, nous supposons que $\mid Q_1 \mid = \ldots = \mid Q_n \mid$. Soit $t = \lceil log_2(\mid Q_1 \mid) \rceil = \ldots = \lceil log_2(\mid Q_n \mid) \rceil$. Pour tout $i \in \{1, \ldots, n\}$, nous considérons un ensemble $At_i = \{r_{i1}, \ldots, r_{it}\}$ de t atomes. Notons

que les ensembles $At_1, \ldots, At_n$ sont deux à deux disjoints. Soit $f : Q_1 \cup \ldots \cup Q_n \rightarrow 2^{Li(At_1 \cup \ldots \cup At_n)}$ une fonction bijective telle que pour tout $i \in \{1, \ldots, n\}$ et pour tout $q \in Q_i$, $f(q)$ est une partie maximale consistante de $Li(At_i)$. Considérons l'automate conditionnel $R'(\mathcal{B}_1, \ldots, \mathcal{B}_n) = (Q', At', Ac', \delta', q'_0, I'_0)$ tel que : $Q' = \{q'\}$, $At' = At_1 \cup \ldots, At_n$, $Ac' = \Sigma_1 \cup \ldots \cup \Sigma_n$, $\delta'(q', a, q') = \{(f(u), f(v)) : i \in \{1, \ldots, n\}, u, v \in Q_i$ et $u \rightarrow_i^a v\}$, $q'_0 = q'$ et $I'_0 = f(q_{01}) \cup \ldots \cup f(q_{0n})$. Nous allons montrer que : (1) R et R' sont calculables par une machine de Turing déterministe en utilisant un espace logarithmique, (2) les automates finis $\mathcal{A}$ et $AF(R(\mathcal{A}))$ sont isomorphes, (3) les automates finis $\mathcal{B}'$ et $AF(R'(\mathcal{B}_1, \ldots, \mathcal{B}_n))$ sont isomorphes. Concernant (1), l'argument est le même que celui que nous avons développé dans la section **??**. L'automate conditionnel $R(\mathcal{A})$ étant la simple réécriture de l'automate fini déterministe $\mathcal{A}$ nous laissons le soin au lecteur de vérifier (2). Concernant (3), nous procédons de la façon suivante. Les états de $\mathcal{B}'$ sont des n-uplet d'états $(q_1, \ldots, q_n)$ où q_i, $i \in \{1, \ldots, n\}$, est un état de $\mathcal{B}_i$. Les états de $AF(R'(\mathcal{B}_1, \ldots, \mathcal{B}_n))$ sont des couples de la forme (q', I) où q' est l'unique état de $R'(\mathcal{B}_1, \ldots, \mathcal{B}_n)$ et I est une partie maximale consistante de $Li(At')$. Soit g la fonction qui associe à chaque n-uplets d'états $(q_1, \ldots, q_n)$ de $\mathcal{B}'$ le couple (q', I) où $I = f(q_1) \cup \ldots \cup f(q_n)$. Nous laissons le soin au lecteur de vérifier que g est une bijection telle que pour tout n-uplet d'états $(q_{11}, \ldots, q_{1n})$, $(q_{21}, \ldots, q_{2n})$ de $\mathcal{B}'$ et pour toute action $a \in \Sigma_{\mathcal{B}'}$, $(q_{11}, \ldots, q_{1n}) \rightarrow^a_{\mathcal{B}'} (q_{21}, \ldots, q_{2n})$ ssi $g(q_{11}, \ldots, q_{1n}) \rightarrow_a g(q_{21}, \ldots, q_{2n})$ (dans $AF(R'(\mathcal{B}_1, \ldots, \mathcal{B}_n))$). La discussion ci-dessus implique que :

Théorème 2 (P^{ac}_{si}) *est EXPTIME-difficile.*

Démonstration. Il suffit de rappeler que (P^{afd}_{si}) est EXPTIME-difficile [9].

6 Classes de complexité

Dans cette section, nous caractérisons les bornes supérieures de complexité. Pour ce qui concerne (P^{ac}_{bi}), rappelons que le problème de la bisimulation entre automates finis est dans P [5]. Considérons l'algorithme suivant : (1) Construire l'automate fini $AF(\mathcal{A})$; (2) Construire l'automate fini $AF(\mathcal{A}')$; (3) Déterminer si $AF(\mathcal{A})$ et $AF(\mathcal{A}')$ sont bisimilaires. Sachant qu'un temps exponentiel par rapport à la taille de $\mathcal{A}$ et $\mathcal{A}'$ est suffisant pour construire $AF(\mathcal{A})$ et $AF(\mathcal{A}')$ à partir de $\mathcal{A}$ et $\mathcal{A}'$, sachant qu'un temps polynomial par rapport à la taille de $AF(\mathcal{A})$ et $AF(\mathcal{A}')$ est suffisant pour déterminer si $AF(\mathcal{A})$ et $AF(\mathcal{A}')$ sont bisimilaires, il en résulte que :

Théorème 3 (P^{ac}_{bi}) *est dans EXPTIME.*

Un argument semblable à l'argument précédent montrerait que :

Théorème 4 (P^{ac}_{si}) *est dans EXPTIME et* (P^{ac}_{et}) *et* (P^{ac}_{it}) *sont dans EXPSPACE.*

Démonstration. Il suffit de rappeler que le problème de la simulation entre automates finis est dans P [5], que le problème de l'équivalence de trace entre automates finis est dans PSPACE [4] et que le problème de l'inclusion de trace entre automates finis est dans PSPACE [4].

Considérons le problème suivant : $(P^{ac\otimes}_{bi})$ *: soient* $\mathcal{A}, \mathcal{B}_1, \ldots, \mathcal{B}_n$ *des automates conditionnels, déterminer si* $\mathcal{A}$ *et* $\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n$ *sont bisimilaires.* Bien entendu, le théorème 1 implique que $(P^{ac\otimes}_{bi})$ est EXPTIME-difficile. Par ailleurs, sachant qu'un temps doublement exponentiel par rapport à la taille des $\mathcal{B}_1, \ldots, \mathcal{B}_n$ est suffisant pour construire $AF(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n)$, sachant qu'un temps exponentiel par rapport à la taille de $\mathcal{A}$ est suffisant pour construire $AF(\mathcal{A})$, il en résulte que $(P^{ac\otimes}_{bi})$ est dans 2-EXPTIME. Nous pouvons, de

la même façon, montrer que les problèmes suivants : $(P_{si}^{ac\otimes})$ *: soient* $\mathcal{A}$*,* $\mathcal{B}_1, \ldots, \mathcal{B}_n$ *des automates conditionnels, déterminer si* $\mathcal{A}$ *est simulé par* $\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n$*,* $(P_{et}^{ac\otimes})$ *: soient* $\mathcal{A}$*,* $\mathcal{B}_1, \ldots, \mathcal{B}_n$ *des automates conditionnels, déterminer si* $\mathcal{A}$ *et* $\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n$ *sont équivalents pour la trace* et $(P_{it}^{ac\otimes})$ *: soient* $\mathcal{A}$*,* $\mathcal{B}_1, \ldots, \mathcal{B}_n$ *des automates conditionnels, déterminer si* $\mathcal{A}$ *est inclus pour la trace dans* $\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n$*,* sont respectivement dans 2-EXPTIME, 2-EXPSPACE et 2-EXPSPACE. Nous ne connaissons pas l'exacte complexité de $(P_{bi}^{ac\otimes})$, $(P_{si}^{ac\otimes})$, $(P_{et}^{ac\otimes})$ et $(P_{it}^{ac\otimes})$. Les problèmes $(P_{bi}^{ac\otimes})$, $(P_{si}^{ac\otimes})$, $(P_{et}^{ac\otimes})$ et $(P_{it}^{ac\otimes})$ sont ceux que nous considérons comme étant les plus proches du problème de la composition des services évoqué dans la section 2. Dans ces problèmes, en effet, $\mathcal{A}$ représente le service but $\mathcal{S}_{but}$ tandis que $\mathcal{B}_1, \ldots, \mathcal{B}_n$ représentent la communauté $\mathcal{C} = (\mathcal{S}_1, \ldots, \mathcal{S}_n)$.

7 Conclusion

Nous avons présenté dans la section 2 un modèle orienté service et nous avons posé le problème de la composition. Ce problème étant indécidable en général, nous avons considéré dans la section 3 une abstraction des services sous la forme d'automates conditionnels. Nous avons ensuite étudié la complexité du problème de la composition de services dans ce modèle simplifié. Les résultats obtenus montrent que la composition est un problème difficile (EXPTIME-difficile ou EXPSPACE-difficile selon qu'il s'agit de bisimulation ou d'équivalence de trace). Nous avons enfin donné des procédures de décision pour l'ensemble des problèmes étudiés. Une question demeure : celle de l'existence de classes d'automates conditionnels pour lesquelles les problèmes de décision que nous avons étudiés deviennent solubles en temps polynomial ou en espace polynomial.

Références

[1] P. Balbiani et F. Cheikh. *Computational analysis of interactiong Web services : a logical approach*. Rapport interne de l'Irit, 2006.

[2] D. Berardi. *Automatic service composition. Models techniques and tools*. Thèse de l'université *La Sapienza*, 2005.

[3] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull et M. Mecella. Automatic composition of transition-based semantic Web services with messaging. In *Proc. 31st Int. Conf. Very Large Data Bases, VLDB 2005*, K. Böhm, C. Jensen, L. Haas, P. Larson et B. Chin Ooi, 613-624, 2005.

[4] M. Garey et D. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. H. Freeman and Company, 1991.

[5] H. Hüttel et S. Shukla. *On the Complexity of Deciding Behavioural Equivalences and Preorders, A Survey*. Rapport dans la série BRICS, 1996.

[6] L. Jategaonkar et A. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, **154(1)**, 107-143, 1996.

[7] M. Minsky. *Computation Finite and Infinite Machines*. Prentice-Hall, 1967.

[8] T. Murata. Petri Nets : Properties, analysis and applications. In *Proc. of the IEEE*, **77(4)**, 541-580, 1989.

[9] A. Musholl et I. Walukiewicz. A lower bound on Web services composition. A paraître.

[10] M. Pistore, A. Marconi, P. Bertoli et P. Traverso. Automated composition of Web services by planning at the knowledge level. In *Proc. Int. Joint Conf. on Artificiel Intelligence, IJCAI 2005*, L. Kaelbling et A. Saffiotti, 1252-1259, 2005.

[11] M. Singh et M. Huhns. *Service-Oriented Computing. Semantics, Process, Agents.* Wiley, 2005.