THESE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Université Toulouse III - Paul Sabatier*
Discipline ou spécialité : *Informatique*

Présentée et soutenue par *Fahima CHEIKH*
Le *19/06/2009*

Titre : *Composition de services: Algorithmes et complexités*

JURY :

Mr Philippe BALBIANI	Directeur de recherche CNRS - Université Paul Sabatier	Directeur de thèse
Mr Jean-Paul BODEVEIX	Professeur – Université Paul Sabatier	Président
Mr Guiseppe DE GIACOMO	Professeur – Université La Sapienza (Rome)	Membre
Mr Guillaume FEUILLADE	Maître de conférences – Université Paul Sabatier	Membre
Mr Olivier PERRIN	Maître de conférences – Université de Nancy	Membre
Mme Sophie PINCHINAT	Maître de conférences HDR – Université de Rennes 1	Rapporteur
Mr Philippe SCHNOEBELEN	Directeur de recherche CNRS – Ecole Normale Supérieure de Cachan	Rapporteur

---

Ecole doctorale : *Mathématique Informatique et Télécommunication de Toulouse*
Unité de recherche : *Institut de Recherche en Informatique de Toulouse*
Directeur de Thèse : *Philippe BALBIANI*
Rapporteur : *Mme. Sohpie PINCHINAT et Mr.Philippe SCHNOEBELEN*

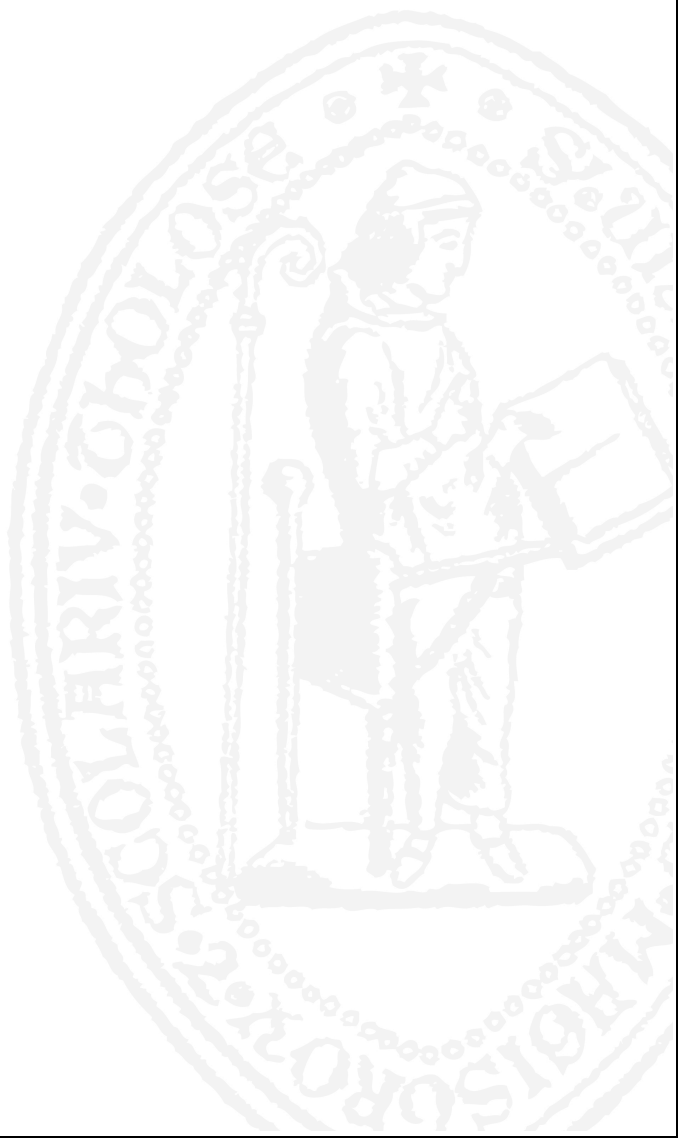

Table des matières

Liste des tableaux

Table des Algorithmes

Résumé

Le problème de la combinaison des services, autrement appelé problème de la composition, constitue le foyer d'une intense activité de recherche. Composer les services entre eux, c'est entrelacer leurs séquences d'actions, de manière à obtenir des séquences qui satisfassent les exigences des clients. Le problème de la composition de services est difficile à résoudre en général.

Dans cette thèse nous considérons des services qui peuvent à la fois exécuter des actions de communications ainsi que des actions internes. De plus, des conditions peuvent être exigées et des effets peuvent être appliqués sur les transitions. Formellement, les services sont représentés par des automates communicants conditionnels.

Nous définissons, pour ce modèle, le problème de la composition et nous étudions sa décidabilité pour différentes relations d'équivalence et de préordre à savoir : l'inclusion de traces, l'équivalence de traces, la simulation et la bisimulation.

Suite aux résultats de décidabilité obtenus, nous proposons trois variantes pour le modèle initial. Pour chacune d'elles, nous définissions le problème de la composition et nous étudions sa complexité pour les relations citées ci-dessus.

Mots-clès

Composition de services, automates, relations d'équivalence et de préordre, décidabilité et complexité.

Abstract

The problem of combining services, also called the services composition problem, constitutes the centre of intense research activity. To compose services consists of merging their action sequences in order to obtain new sequences that satisfy the client requirements. This problem is in general difficult to solve.

In this thesis, we consider services that are able to perform two kinds of actions : communication actions and internal actions. Moreover, services are able to verify conditions before an action execution and to change the value of variables after an action execution. Formally, services are represented by conditional communicating automata.

We define for this model the services composition problem and we study its decidability for the following preorder and equivalence relations : traces inclusion, trace equivalence, simulation and bisimulation.

Further to the decidability results obtained we propose three variants of the initial model. For each variant we define the composition problem and we study its complexity regarding the relations cited above.

Key-words

Services composition, automata, equivalence relations, preorder relations, decidability and complexity.

À la mémoire de mon père

Remerciements

J'exprime mon profond remerciement à Philippe BALBIANI qui m'a permis de réaliser un de mes rêves les plus chères. Grâce à lui, j'ai pu effectuer une thèse dans un cadre très agréable. Je le remercie pour sa patience, ses conseils et surtout pour m'avoir transmis son goût pour la recherche.

Je tiens également à remercier Guillaume FEUILLADE, pour sa disponibilité ses encouragements et pour m'avoir fait découvrir la théorie de la synthèse de contrôleur. Cela m'a permis de voir ma thèse sur un autre angle de vue, chose que j'apprécies.

Je n'oublie pas de remercier Giuseppe DE GIACOMO et toute son équipe, pour leur bon accueil au sein de leur laboratoire à l'Université La Sapienza. Ce séjour a été très bénéfique et enrichissant aussi bien d'un point de vue scientifique qu'humain.

Un très grand merci aux rapporteurs Philippe SCHNOEBELEN et Sophie PINCHINAT pour avoir accepté de lire ma thèse et pour leurs rapports positifs et encourageants.

Je tiens aussi à remercier Jean Paul BODEVEIX et Olivier PERRIN pour l'intérêt qu'ils ont porté à mon travail en acceptant d'être membre de mon Jury.

Je remercie également tous les membres du projet COPS avec qui j'ai partagé de bons moments. Leurs questions et remarques mon permis d'une part de mieux comprendre la problématique considérée dans ma thèse et d'autre part de trouver certaines solutions. Sans oublier tous les membres de l'équipe LILaC, pour la bonne ambiance et les moments agréables que j'ai passé en leur compagnie.

Je remercie vivement mes parents, mon mari et ma famille de m'avoir encouragé à venir en France pour réaliser une thèse, ainsi que pour leur soutient au quotidien. Je remercie également tous mes amis, que je considère comme ma seconde famille.

Enfin, que tous ceux qui ont contribué, de près ou de loin, à l'élaboration de ce travail trouvent ici l'expression de ma gratitude.

Chapitre 1

Introduction

1.1 Programmation orientée services

La programmation orientée service [SH05] (SOC, Service Oriented Computing) est un paradigme de programmation qui utilise les services comme des constituants élémentaires à partir desquels sont réalisées des applications réparties. En particulier, afin de créer un nouveau service qui peut réaliser une tâche complexe que les services existants ne peuvent pas réaliser, une des solutions est de composer ces services.

Par conséquent, l'architecture des services Web est une architecture orientée composant [MMM06] (SOA, Service Oriented Architecture). Les concepts de bases de cette architecture sont les trois éléments suivants :

- Service fournisseur,
- Service annuaire et
- Service client.

Les services fournisseurs sont des services qui proposent des fonctionnalités que d'autres services peuvent utiliser. Lorsque les services fournisseurs sont créés, le but est qu'ils soient accessibles par le maximum d'utilisateurs. Pour cela, il faut que ces services puissent être facilement retrouvés. En effet, les services sont répertoriés dans un service particulier appelé annuaire. Dans cet annuaire se trouve l'adresse et la description des services fournisseurs répertoriés. Les utilisateurs des services fournisseurs sont appelés services clients.

Il existe différents problèmes concernant les services du modèle SOA. Parmi ces travaux nous citons : la description de services [CCMW01], la découverte de services [Dra01], la validation et le test de services [PBL08] et la composition de services [Ber05, MPT08].

Dans le cadre de cette thèse, nous sommes intéressée par l'étude de la complexité du problème de la composition.

1.2 Les services et le problème de la composition de services

Le problème de la combinaison des services, autrement appelé problème de la composition [BCGM06, PMBT05], constitue le foyer d'une intense activité de recherche. Ce problème peut être décrit comme suit : étant donnés un ensemble de services disponibles et un but, est-il possible de combiner les services disponibles afin de réaliser le but ? Pour avoir une description plus précise, il faut d'abord définir les services.

Il existe différentes définitions pour les services [ACKM03]. Certaines sont plus générales que d'autres. Par exemple, dans certaines définitions, un service représente toute application accessible sur internet. Dans des définitions plus spécifiques, comme celle proposée par [JC], les services sont définis comme des applications basées sur le langage XML (Extensible Markup Language). Plus précisément, un service doit utiliser le langage WSDL (Web Services Description Language) pour sa description statique, il doit utiliser le protocole de communication SOAP (Simple Object Access Protocol) pour communiquer et il doit utiliser l'annuaire UDDI (Universal Description, Discovery and Integration) pour être répertorié et pour rechercher d'autres services.

D'un point de vue formel, il existe aussi plusieurs représentations des services. Souvent, les services sont vus comme des automates finis. Dans ces automates, il est possible d'effectuer uniquement des actions de communication [PTBM05], d'exécuter uniquement des actions internes [BFDP08, BCGM06] ou d'exécuter les deux types d'actions [BCG+05]. Parfois, ces automates sont enrichis par l'ajout de conditions et d'effets sur les transitions [BCG+05].
Les réseaux de Petri sont également utilisés pour représenter formellement les services [HB03, MPPP02]. Par ailleurs, les réseaux de Petri permettent aussi la description des conditions et des effets pour les transitions. Dans d'autres approches, les services sont représentés par des axiomes de la logique linéaire [RKM04].

Par conséquent, la description formelle du problème de la composition est différente d'un formalisme à l'autre. Par exemple, lorsque les services sont des automates, le problème de la composition devient celui de l'équivalence entre un automate est un produit d'automate [BFDP08, MW08]. Lorsque

les services sont représentés par des axiomes, le problème de la composition consiste à prouver un séquent à partir d'un ensemble d'axiomes.

Cependant, même si le formalisme choisi pour représenter les services est le même, les approches pour le résoudre ne sont pas forcément identiques. Par exemple, dans [PTBM05] et [BCG+05], les services sont formalisés par des automates, mais dans [PTBM05] les auteurs utilisent une méthode basée sur la planification pour résoudre le problème de la composition et dans [BCG+05], les auteurs utilisent une méthode basée sur la satisfaction d'une formule dans la logique dynamique propositionnelle (PDL). Ainsi, des techniques de résolution de différents domaines peuvent être appliquées afin de résoudre le problème de la composition.

1.2.1 Notre approche

En ce qui nous concerne, nous représentons les services par des *automates communicants conditionnels CCA* [BCF08a, BCF08b]. Ces automates sont capables d'exécuter des actions internes ainsi que des actions de communication. De plus, ils contiennent des préconditions et des effets sur les transitions. Notre modèle s'inspire du modèle COLOMBO [BCG+05]. Plus particulièrement, il utilise certaines de ses notions :

- service client,
- service but,
- services disponibles et
- service médiateur.

L'objectif de la composition est d'utiliser les services disponibles afin de créer un service qui peut effecteur les tâches que le service client veut réaliser. La structure du service souhaité est représentée par le service but. Ce dernier communique uniquement avec le client. Quant au service médiateur, c'est celui qui sera synthétisé afin de réaliser le but. Plus précisément, le service médiateur peut effectuer uniquement des actions de communication. Par conséquent, afin de simuler les actions internes que le but effectue, il devra invoquer les services disponibles.

En termes de niveaux d'abstraction, notre modèle se situe entre le modèle COLOMBO [BCG+05] et le modèle Romain [Ber05, BCGM06]. Effectivement, il est plus abstrait que le modèle COLOMBO par le fait de considérer des messages sans contenu et des actions sans paramètres. De plus, à la différence du modèle COLOMBO dans lequel les services partagent une base de données, dans notre modèle les services partagent un ensemble fini de variables propositionnelles. Toutefois, notre modèle est moins abstrait que le modèle Romain, car dans ce dernier les actions internes et les actions de

communication ne sont pas distinguables. De plus, dans le modèle Romain, il n'existe ni de préconditions ni d'effets sur les transitions. Par conséquent, notre modèle est plus réaliste que le modèle Romain.

Le problème de la composition que nous considérons est également inspiré du modèle COLOMBO et il est défini de la façon suivante :

Étant donnés un service but, un service client et des services disponibles, déterminer l'existence d'un service médiateur tel que : le système composé du service but et du service client soit équivalent au système composé du service médiateur, du service client et des services disponibles.

L'une des différences entre ce problème et celui décrit dans COLOMBO est le type d'équivalence considéré. En effet, dans notre modèle nous considérons : l'inclusion de traces, l'équivalence de traces, la simulation et la bisimulation. Tandis que, dans COLOMBO, l'isomorphisme est la relation d'équivalence considérée. L'isomorphisme entre les automates est une relation plus restrictive que les relations que nous avons citées, ci-dessus. L'isomorphisme peut être vue comme une égalité entre deux systèmes, modulo le renommage des états d'un des systèmes par les états de l'autre. Dans notre approche, l'objectif est de pouvoir exécuter les actions dans l'ordre exigé par l'utilisateur, mais pas forcément avec le même nombre d'états que le service but.

Une autre différence réside dans le fait que dans COLOMBO les services considérés sont déterministes. En effet, toutes les actions effectuées par les services sont supposées observables. Ce qui n'est pas le cas dans la réalité. De plus, dans le modèle COLOMBO, des restrictions ont été considérées afin de résoudre le problème. Nous n'avons pas considéré ces restrictions, spécifiques aux services Web, car nous voulons apporter des résultats qui peuvent être appliqués dans d'autres domaines. Comme le domaine des systèmes multiagents en intelligence artificielle.

Dans le Chapitre 2, nous donnons une comparaison détaillée de notre modèle avec le modèle COLOMBO. Nous verrons également plus en détail les différents formalismes utilisés pour représenter les services. Ainsi que les différentes approches utilisées pour résoudre le problème de la composition. Nous verrons à partir des résultats décrits que le problème de la composition n'est pas un problème facile à résoudre.

1.3 Objectif et contribution de la thèse

L'objectif de cette thèse n'est pas de créer un nouveau modèle pour représenter les services. Toutefois, il faut nous abstraire du modèle COLOMBO afin de réduire la complexité du problème de la composition. Cette

abstraction, nous permettra également de résoudre le problème de la composition pour des relations d'équivalences et de préordre non étudiées dans d'autre travaux.

En effet, le problème de la composition pour la simulation [BFDP08] a été étudié mais dans un modèle plus abstrait que celui considéré dans cette thèse. Les résultats concernant la simulation sont donnés dans le cas où les services n'effectuent pas d'action de communication.

Quant au problème de la composition pour la relation de bisimulation il a été étudié [PBLH06] avec un modèle inspiré du modèle COLOMBO, cependant les auteurs considèrent certaines restrictions : les services sont déterministes et après l'envoi d'un message le service se retrouve dans un état à partir duquel il ne peut plus effectuer d'action, ou bien l'unique action qu'il peut effectuer est la réception d'un message. Ces restrictions permettent de réduire le nombre de transitions possibles à partir d'un état donné.

Quant au problème de la composition avec les relations d'inclusion de traces et l'équivalence de traces, il n'a pas été considéré. Cependant, soulignons que dans le cas où les automates représentant les services disponibles sont déterministes ainsi que leur produit alors : le problème pour la simulation (resp. bisimulation) est équivalent au problème pour l'inclusion de traces (resp. équivalence de traces). Dans le Chapitre 3, nous donnons la définition formelle de ces relations. Dans le Chapitre 4, à la suite de la définition formel du problème de la composition, nous expliquons la différence entre ces relations d'un point de vue pratique.

Notre objectif est donc de pouvoir analyser le problème de la composition pour différentes relations : inclusion de traces, équivalence de traces, simulation et bisimulation.

De plus, ayant comme but de résoudre le problème de la composition pour un modèle de service proche de la réalité, nous considérons : des actions de communication et des conditions et des effets sur les transitions, sans pour autant considérer les restrictions spécifiques aux services. En effet, nous voulons avoir un modèle et des résultats qui peuvent s'appliquer dans d'autres domaines. Ce modèle est proche des réseaux de Petri, nous verrons dans les Chapitre 4, 5 et 6 qu'il est possible d'associer à un automate communicant conditionnel un réseau de Petri dont l'exécution est isomorphe à celle de l'automate communicant conditionnel[1].

Effectivement, nous apporterons les contributions suivantes :

[1] L'exécution d'un réseau de Petri est représentée par son graphe de marquage. Dans le Chapire 3, nous donnons une définition formelle de l'exécution d'un ACC et d'un réseau de Petri.

1. Dans le cas où les services communiquent à travers des ports non-bornés (voir Chapitre 4), nous avons montré que le problème de la composition est indécidable pour l'équivalence de traces et la bisimulation et nous avons montré qu'il est décidable pour l'inclusion de traces et la bisimulation.
2. Nous avons considéré trois variantes de notre modèle. La première variante (voir Chapitre 5) est une abstraction de notre modèle initial. Plus précisément, dans ce modèle, les actions de communication sont considérées comme des actions internes [BCF07, CGM06, BFDP08]. Dans la seconde variante (voir Chapitre 6), nous considérons à nouveau la communication entre les services [BCF09, BCF08b, BCF08a]. Cependant, cette communication est effectuée à travers des ports bornés. Dans la troisième variante (voir Chapitre 7), nous considérons une communication synchrone entre les services. C'est-à-dire, qu'un service ne peut envoyer de message que s'il existe un autre service pour le recevoir. Réciproquement, un service ne peut recevoir de message que s'il existe un service prêt à l'envoyer.

   Pour ces trois variantes nous avons résolu le problème de la composition en considérant : l'inclusion de traces, l'équivalence de traces, la simulation et la bisimulation. Plus particulièrement, nous donnons des résultats de complexité en termes de bornes inférieures et supérieures. Dans le Chapitre 8 nous donnons le détail des résultats obtenus.
3. Dans la seconde variante, afin de résoudre le problème de la composition, nous l'avons réduit à un problème de satisfaction d'une formule du μ-calcul [BCF08a]. L'avantage de cette réduction est qu'elle nous permet d'exprimer en plus de l'équivalence entre deux systèmes, le fait qu'un système vérifie certaines propriétés. Une de ces propriétés peut être la vivacité du système ou sa tolérance aux fautes. Ainsi, la méthode que nous avons proposée permet de résoudre un problème plus général que le problème de la composition initial.
4. Dans la troisième variante, nous avons introduit une nouvelle description pour le problème de la composition. Cette description permet de prendre en considération des aspects de sécurité [BCHK09]. Comme nous l'avons déjà précisé, les services peuvent exiger des conditions pour effectuer des transitions. Ces conditions peuvent concerner des certificats que les utilisateurs doivent posséder [CGM06]. Dans la nouvelle description du problème de la composition, au lieu de se focaliser sur l'existence d'un médiateur, on se focalise sur l'existence d'une formule caractéristique de la composition. Cette formule, lorsqu'elle est

satisfaite, garantit que le système composé des services disponibles est équivalent au service but. En pratique, cette formule permet d'exprimer des restrictions sur les certificats que le client doit posséder. Nous avons proposé des procédures pour résoudre ce problème pour : l'inclusion de traces, la simulation, la bisimulation et l'inclusion de trace.

Cependant, il reste des problèmes pour lesquels nous n'avons pas trouvé de solution. Par exemple, nous n'avons pas de complexité exacte pour le problème de la composition de la seconde variante, lorsque les relations d'équivalence de traces et de bisimulation sont considérées. De plus, il y a certaines notions importantes qui ne sont pas considérées dans notre modèle. Par exemple, la notion de contraintes de durée sur les transitions dans les automates. Enfin, dans notre approche, les services sont supposés parfaits et ne peuvent subir de panne. Ce qui n'est pas le cas dans la réalité. Dans le Chapitre 8, nous discutons plus en détails ces différents points.

1.4 Organisation de la thèse

Dans le Chapitre 2, nous donnons des notions de base concernant la description des services et nous présentons différentes approches existantes pour la résolution du problème de la composition. Dans le Chapitre 3, nous donnons les définitions de base concernant les automates et les réseaux de Petri. Ensuite, dans le Chapitre 4, nous décrivons notre modèle des services et nous présentons nos résultats concernant la décidabilité/indécidabilité du problème de la composition. Dans les Chapitres 5, 6 et 7, nous présentons les première, deuxième et troisième variantes de notre modèle ainsi que les résultats de complexité que nous avons obtenus Enfin, dans le Chapitre 8, nous établissons un bilan sur les résultats obtenus et nous discutons des travaux futurs que nous envisageons.

Chapitre 2

Travaux voisins

2.1 Description des services

Les services sont créés pour être éventuellement utilisés par des humains ou par d'autres services. Pour que cela soit possible, il faut que les services soient décrits avec précision. Par exemple, il faut que les services publient les opérations qu'ils peuvent réaliser, le type de données qu'ils manipulent ainsi que les échanges de messages qu'ils effectuent. Il existe différents langages pour la description d'un service Web. Parmi ces langages, nous citons : Web Services Description Language (WSDL) [CCMW01], Web Service Choreography Interface (WSCI) [BS02], Web Services Choreography Description Language (WSCDL) [KBR04] et Business Process Execution Language for Web Services (BPEL4WS) [IBM07].

Il existe d'autres langages pour la description de services Web. Parmi eux il y a ceux basés sur l'utilisation d'ontologie, comme OWL-S [DM04][1] (Ontology Web Language for Services). OWL-S est muni d'un langage pour décrire la sémantique des services appelés SWS (Semantic Web Services).

Description WSDL

Un document WSDL définit un service en définissant les opérations qu'il fournit, les messages qu'il échange, les données qu'il utilise, l'adresse du réseau ainsi que le protocole utilisé pour le transfert de données. La description WSDL est une description statique de l'interface d'un service. Cela signifie que dans cette description l'ordre des opérations n'est pas déterminé.

[1] OWL-S était à l'origine appelé DARPA Agent Markup Language for Services (DAML-S) [ABH+02].

Les opérations sont considérées uniquement d'un point de vue atomique. Pour une meilleure compréhension nous présentons l'exemple suivant [BS02].

Exemple 2.1.1. *Le service Travel Agent décrit par le document WSDL de la Figure 2.1, est un service d'agence de voyage. Ce service peut effectuer plusieurs opérations à savoir : OrderTrip, BookTickets et SendStatement. Pour effectuer l'opération OrderTrip, le service doit recevoir le message tripOrderRequest pour ensuite envoyer le message tripOrderAcknowledgement. De la même façon, pour effectuer l'opération bookTickets, le service doit recevoir le message bookingRequest pour ensuite envoyer le message bookingConfirmation. Concernant l'opération SendStatement le service n'attend pas de recevoir un message pour l'effectuer, cependant il envoie le message SendStatement. Tous les messages reçus et envoyés par le service sont définis dans la partie message. Chaque message contient des informations d'un certain type. Par exemple, le message tripOrderRequest qui sera envoyé par le client du service, doit contenir les informations sur le voyageur (traveler) ainsi que des informations sur le voyage (trip). Les types traveler et trip sont définis dans la partie types. Par exemple, les informations concernant le voyageur seront son nom et son identifiant, de type chaîne de caractères.*

Dans la Figure 2.2, est représentée la séquence d'échange d'informations effectuée entre le service *Travel Agent* et un client/utilisateur de ce service. Néanmoins, cette séquence n'est pas décrite dans le fichier WSDL. Par conséquent, un client du service *Travel Agent* ne peut pas savoir dans quel ordre il doit échanger ces messages avec le service. De plus, à partir du fichier WSDL les dépendances entre les opérations ne sont pas spécifiées. En effet, il n'est pas clair comment le service relie la requête de confirmation avec la requête du voyage préalablement soumise. Enfin, dans le document WSDL il n'existe pas de proposition d'alternative pour gérer les erreurs. L'utilisateur ne sait pas ce qui peut se produire dans le cas d'une erreur durant l'exécution d'un service. Le service *Travel Agent* doit également avoir la possibilité de choisir entre le fait d'accepter un paiement ou pas, en se basant sur les informations contenues dans la carte de crédit de l'utilisateur du service. Afin de remédier à ces problèmes, la description dynamique de l'interface des services appelé WSCI a été mis au point. Nous présentons cette description dans la section suivante.

Description WSCI

La description WSCI permet de compléter la description statique d'un document WSDL. Comme nous l'avons signalé dans la section précédente,

```
<? xml version = "1.0"?>
<definitions name = "Travel Agent Static Interface"
targetNamespace = "http ://example.com/consumer/TravelAgent"
⋮ >

<!-- ****COMPLEX TYPES **** -->
</types>
    <complexType name = "Traveler">
      <sequence>
        <element name = "name" type = "xsd :string" />
        <element name = "travelerID" type = "xsd :string" />
     </sequence>
    </complexType>
    <complexType name = "trip">
      <sequence>
        <element name = "itineraryID" type = "xsd :string" />
        <element name = "startDate" type = "date" />
⋮
      </sequence>
    </complexType>
⋮
</types>

<!-- **** MESSAGES **** -->
<message name = "tripOrderRequest">
    <part name = "traveler" type = "tns :traveler"/>
    <part name = "trip" type = "tns :trip"/>
</message>
⋮
<-- ****TRAVEL AGENT PORT TYPES ****-->
<portType name = "TAtoTraveler">
    <operation name = "OrderTrip">
      <input message = "tns :tripOrderRequest"/>
      <output message = "tns :tripOrderAcknowledgement"/>
    </operation>
    <operation name = "bookTickets">
      <input message = "tns :bookingRequest"/>
      <output message = "tns :bookingConfirmation"/>
    </operation>
    <operation name = "SendStatement">
      <output message = "tns :statement"/>
  </operation>
</portType>
</definitions>
```


Fig. 2.1 – Document WSDL du service Travel Agent.

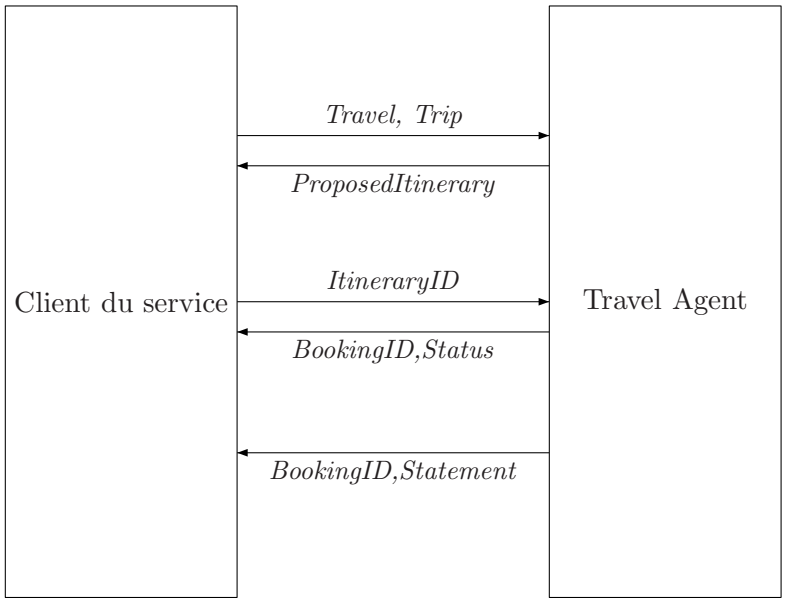

FIG. 2.2 – Séquence d'échange de messages entre le service Travel Agent et un client.

dans le cadre d'une collaboration entre services, il est parfois nécessaire que les services participants décrivent leur comportement. En effet, l'ordre dans lequel ils effectuent leurs opérations, les dépendances entre leurs opérations ainsi que leur gestion des erreurs doivent être spécifiés. Lorsqu'un service décrit tous ces éléments, cela permet à ses collaborateurs d'éviter d'invoquer les opérations dans un mauvais ordre et ainsi d'éviter certaines erreurs. De plus, si ce service bloque et qu'il prévient ses collaborateurs que dans ce cas un message particulier est envoyé, cela leur permettra de prendre leurs dispositions dans le cas où ils reçoivent un tel message. Ainsi la description WSCI permet de caractériser le comportement observable d'un service Web et non pas ses opérations internes. Considérons l'exemple suivant [BS02].

Exemple 2.1.2. *Dans cet exemple, nous représentons, en utilisant la description WSCI, les échanges de messages de la Figure 2.2. Le document représenté dans la Figure 2.3 est le document représenté dans la Figure 2.1 complété par une description dynamique de l'interface. Comme nous pouvons l'observer, la description WSCI est emboîtée dans la description WSDL (il n'est pas nécessaire de créer un nouveau fichier pour la description WSCI).*

La description des parties complexType, message et portType reste identique à celle de l'exemple précédent. La corrélation nommée itineraryCorrelation indique que les messages qui ont le même itineraryID représentent le même voyage (trip). Cela permettra au service de gérer des exécutions parallèles du même échange de messages. La partie interface contient la définition de tous les processus du service. Si le processus a l'attribut instantiation égal à message, cela signifie que l'exécution du processus sera déclenchée dès que la première action peut être effectuée. C'est-à-dire, dans notre exemple, dès réception du message tripOrderRequest. Dans le cas du BookSeats processus on a l'attribut instantiation égal à other, cela signifie que ce processus n'est pas instancié grâce à la réception d'un message mais lorsqu'il est explicitement invoqué. Dans notre exemple, il est invoqué dans la partie call de l'action ReceiveConfirmation. Chaque processus contient un ensemble d'actions. Ces actions peuvent être exécutées selon une séquence, un choix ou de façon répétée. Dans notre exemple, les actions ReceiveTripOrder, ReceiveConfirmation, et SendStatement doivent être effectuées en séquence.

Lorsque des services décident de collaborer, afin de réaliser une tâche complexe, qui ne peut être effectuée que partiellement par chaque service, il faut que ces services soient chorégraphiés. En d'autres termes, ces services doivent respecter certaines règles et contraintes concernant les messages observables qu'ils échangent. Pour décrire ces règles il faut utiliser un langage de description de chorégraphie comme WSCDL. Une fois la description WSCDL établie, il faut que la description WSCI de chaque membre de la chorégraphie respecte ces règles. Ainsi, dans la description WSCI de l'interface de chaque service nous avons la description d'une partie des échanges de messages et de leurs dépendances. Chaque service ne décrit que les échanges qui le concernent dans la chorégraphie. Lorsque la collaboration est effectuée entre différents partenaires, possédant chacun un ensemble de services, les règles vont être établies après un accord entre ces derniers.

Description BPEL4WS

Le rôle de la description BPEL4WS est de définir un nouveau service, en utilisant des services existants. Contrairement à WSCDL et WSCI qui ne sont pas exécutables, BPEL4WS est un langage d'implémentation de processus. Cependant, comme pour WSCDL et WSCI, la description de l'interface statique du service composé est décrite en utilisant WSDL. Le document BPEL4WS décrit le comportement interne du service, en plus de la description des échanges de messages entre le nouveau service et les

```
<? xml version = "1.0"?>
<wsdl :definitions name = "Travel Agent Dynamic Interface"
targetNamespace = "http ://example.com/consumer/TravelAgent"
⋮ >
<!-- WSDL complex types -->
<!-- WSDL message definitions -->
<!-- WSDL operations and port types -->
<correlation name = "itineraryCorrelation"
   property = "tns :itineraryID">
</correlation>
<interface name = "TravelAgent">
   <process name = "PlanAndBookTrip"
     instantiation = "message">
     <sequence>
       <action name = "ReceiveTripOrder"
         role = "tns :TravelAgent"
         operation = "tns :TAtoTraveler/OrderTrip">
       </action>
       <action name = "ReceiveConfirmation"
         role = "tns :TravelAgent"
         operation = "tns :TAtoTraveler/bookTickets">
        <correlate correlation="tns :itineraryCorrelation"/>
         <call process = "tns :BookSeats" />
       </action>
       <action name = "SendStatement"
         role = "tns :TravelAgent"
         operation = "tns :TAtoTraveler/SendStatement"/>
       </action>
     </sequence>
   </process>
   <process name = "BookSeats" instantiation = "other">
     <action name = "bookSeats"
       role = "tns :TravelAgent"
       operation = "tns :TAtoAirline/bookSeats">
     </action>
   </process>
</interface>
</wsdl :definitions>
```

FIG. 2.3 – Description, en utilisant WSCI, de la séquence d'échange de messages de la Figure 2.2.

services existants. Ce comportement interne n'est pas utile pour les utilisateurs du service. Cependant, il permet au service de réaliser ces fonctions. À la différence de WSCDL qui est un langage pour la chorégraphie de service, BPEL4WS est un langage d'orchestration de services. La différence est que le service décrit en BPEL4WS est celui qui orchestre les échanges de messages. En d'autres termes, il participe à tous les échanges de messages. Quant aux communications effectuées entre les services existants, elles ne sont pas considérées. Considérons l'exemple suivant[2].

Exemple 2.1.3. *Considérons le document représenté dans la Figure 2.4. Ce service autorise ou non ses utilisateurs à faire un prêt. Pour cela il fera appel au service Loan approver qui lui est capable de décider si un prêt doit être autorisé ou non. Ainsi, le processus de la Figure 2.4 n'aura qu'à transmettre la requête de son client au service Loan approver et ensuite à transmettre la réponse du Loan approver à son client. Par conséquent, il lui suffit de jouer le rôle de médiateur entre son client et son fournisseur. Bien-entendu, il est possible d'imaginer des exemples plus complexes, dans lesquels on a plusieurs fournisseurs, chacun permettant de réaliser une opération différente.*

Dans le fichier WSDL du service se trouve le détail de la description du PortType $loanApprovalPT$, des messages $CreditInformationMessage$ et $approvalMessage$, et du LinkType $loanApprovaLinkType$. Plus précisément, le Port Type $loanApprovalPT$ contient une seule opération appelée approve, cette opération a comme input message CreditInformationMessage [3] *et comme out put message $approvalMessage$*[4]*. En ce qui concerne la notion de Link Type, elle est utilisée pour lier le client ainsi que le fournisseur (Loan approver) au processus que nous décrivons. Dans le fichier WSDL du processus, le linktype $loanApprovaLinkType$ est défini de la façon suivante :*

```
<slnk :serviceLinkType name="loanApprovalLinkType">
   <slnk :role name="approver">
     <portType name="apns :loanApprovalPT"/>
   </slnk :role>
</slnk :serviceLinkType>
```

Cette définition permet de spécifier que le $loanApprovalLinkType$ concerne le port Type $loanApprovalPT$.

Considérons la définition d'un processus BPEL. Dans la description du

[2]Voir : http ://www.ibm.com/developersworks/library/ws-bpelcol1.

[3]Ce message contient, le prénom (FirstName), le nom (Name) et le montant du prix (Amount).

[4]Ce message contient la valeur de la donnée Accept.

processus il faut suivre les étapes suivantes :

1. *Il faut commencer par définir les différents partenaires du processus, en spécifiant leur nom, le service linkType ainsi que les différents rôles. My role désigne le rôle du processus et partner role celui du partenaire.*
2. *Une fois les partenaires définis, il faut définir les containers. Un container permet de stocker les données reçues par le service afin de les réutiliser. Par exemple, le message $CreditInformationMessage$ est reçu par le processus et ensuite transmit au fournisseur. Il faut donc le stocker dans un container (nommé request).*
3. *Maintenant, il est possible de décrire la séquence de messages entre le processus et les différents partenaires. La séquence contient une réception de message (receive), une invocation (invoke) et une réponse (reply). Concernant l'activité invoke, il faut indiquer le nom de l'activité, le partenaire invoqué (approver), le portype (loanApprovalPT) et plus précisément l'opération (approve) pour laquelle le partenaire est invoqué, le container qui contient les données que le processus envoie à son partenaire et le container dans lequel sera déposée la réponse du partenaire.*

Une représentation de l'échange effectué entre le processus, le client et le fournisseur est donnée dans la Figure 2.5

2.2 Composition de services

Les services Web peuvent être combinés afin de créer de nouveaux services capables de réaliser des tâches complexes. Le problème de la composition de services peut être énoncé comme suit :
étant donnés un ensemble de services disponibles et un but, est-il possible de combiner les services disponibles afin de réaliser le but ?

Dans ce qui suit, nous présentons brièvement, les travaux existants pour la résolution du problème de la composition. Ensuite, nous présentons notre approche afin de la comparer aux approches existantes présentées.

2.2.1 Approches existantes

Différents travaux de recherche s'intéressent au problème de la composition de service. Ces approches ont chacune des manières différentes de considérer les services, la notion de composition et la représentation du but. Nous allons décrire quelques-uns de ces travaux.

```
<process name="loanApprovalProcess"
targetNamespace="http ://acme.com/simpleloanprocessing"
xmlns="http ://schemas.xmlsoap.org/ws/2002/07/
business-process/"
xmlns :lns="http ://loans.org/wsdl/loan-approval"
xmlns :loandef="http ://tempuri.org/services/loandefinitions"
xmlns :apns="http ://tempuri.org/services/loanapprover">
⋮ >
<partners>
   <partner name="customer"
     serviceLinkType="lns :loanApprovaLinkType"
     myRole="approver"/>
   <partner name="approver"
     serviceLinkType="lns :loanApprovalLinkType"
     partnerRole="approver"/>
   </partners>
<containers>
   <container name="request"
   messageType="loandef :CreditInformationMessage"/>
   <container name="approvalInfo"
   messageType="apns :approvalMessage"/>
</containers>
<sequence>
   <receive name="receive1" partner="customer"
portType="apns :loanApprovalPT"
operation="approve" container="request"
createInstance="yes">
   </receive>
   <invoke name="invokeapprover"
     partner="approver"
     portType="apns :loanApprovalPT"
     operation="approve"
     inputContainer="request"
     outputContainer="approvalInfo">
   </invoke>
   <reply name="reply" partner="customer"
portType="apns :loanApprovalPT"
     operation="approve" container="approvalInfo">
   </reply>
</sequence>
</process>
```


Fig. 2.4 – Déscription BPEL du service Loan Approval.

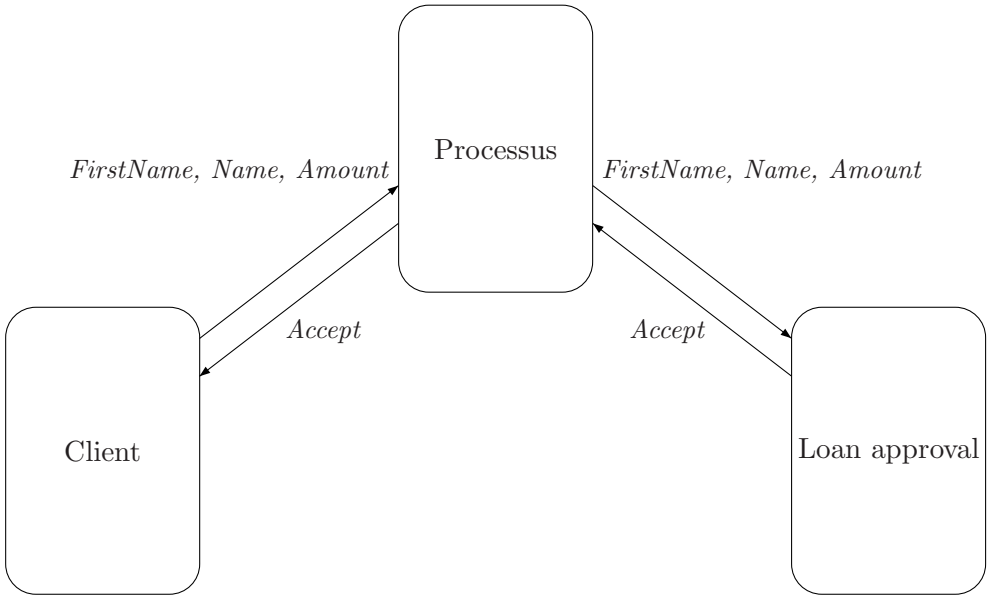

FIG. 2.5 – Séquence d'échange de messages entre le client, le processus et le fournisseur.

Approche de Pistore, Traverso et Bertoli

Dans l'approche proposée par Pistore, Traverso et Bertoli dans [PTB05] où celle proposée dans[PTBM05], les auteurs considèrent les services Web comme des documents BPEL. Ces documents sont formellement représentés par des systèmes de transition dans lesquels les actions exécutées sont l'émission et la réception de messages. Des actions internes sont également exécutables, mais elles sont représentées par des ϵ-transitions. C'est-à-dire que ces actions ne sont pas observables. De plus, les états du système de transition sont étiquetés par des ensembles de variables propositionnelles. Dans cette approche, le but est défini par une formule du langage EaGLe [LPT02] à satisfaire[5]. Intuitivement, cette formule peut, par exemple, exprimer le fait qu'aucun des services disponibles ne doit atteindre un état bloquant.

Dans cette approche, le problème de la composition est défini comme suit :
étant donnés un ensemble de services disponibles et un but, synthétiser un processus qui interagit de façon asynchrone avec les services disponibles et

[5]Le langage EaGle est une extension de CTL (Computational Tree Logic)

qui satisfait le but.

Pour résoudre ce problème, les auteurs le réduisent à un problème de planification. Rappelons, qu'étant donnés un domaine de planification et une formule à satisfaire, un planificateur permet de synthétiser un plan solution. Dans la réduction proposée par les auteurs, le domaine de planification est représenté par le produit asynchrone des services disponibles et la formule à satisfaire est représenté par la formule but. Le service composé est représenté par un système de transition, obtenu à partir du plan solution synthétisé par le planificateur.

Le produit synchrone, sur la communication, entre le service composé et les services disponibles permet d'obtenir un système qui satisfait le but. Intuitivement, cela veut dire que le but est satisfait lorsque le service composé communique avec les services disponibles.

Les auteurs présentent leurs résultats expérimentaux appliqués à différentes applications. Néanmoins, ils ne présentent pas de résultat théorique concernant les bornes inférieures et supérieures de complexité du problème de la composition.

Afin de réduire le temps de calcul nécessaire pour l'obtention d'une composition, les auteurs ont proposé [BPT06] une approche dans laquelle au lieu de construire le domaine de planification explicitement ils le construisent à la volée au moment où ils construisent le plan solution. Pour cela, ils utilisent des algorithmes de recherche de type et-ou [BCR01].

Dans d'autres articles, les auteurs ont également proposé des variantes de leur modèle en ajoutant aux services des bases de connaissances [PMBT05] ou en ajoutant des théories (conjonction de clauses) [HBP07]. Pour plus d'information concernant l'outil que les auteurs ont mis au point, afin de résoudre le problème de la composition, ainsi que son utilisation sur un scénario concret, il faut consulter les articles [TPC+05, MPT08, MPPT07].

Approche de Berardi, Calvanese, De Giacomo, Hull et Mecella

Dans l'approche proposée par Berardi, Calvanese, De Giacomo, Hull et Mecella [BCG+05], les auteurs considèrent les services Web comme des documents OWL-S. Ces documents sont formellement représentés par des automates conditionnels. Dans ces automates, il est possible d'exécuter des actions de communication ainsi que des actions internes, à la différence du modèle proposé par Pistore et al. Une autre différence est que la communication est effectuée à travers des ports, ces derniers peuvent être vus comme des files d'attente de messages. Dans les automates conditionnels, comme leur nom l'indique, des conditions sur les variables internes peuvent être

exigées pour l'exécution d'une action.

Le modèle COLOMBO proposé est constitué des éléments suivants : une base de données, des services disponibles qui partagent la base de données, un service client qui peut exécuter uniquement des actions de communication et qui représente l'utilisateur du service à composer, un service but qui représente la requête du client et un service médiateur qui représente le service à composer et qui a pour rôle de s'interposer entre le service client et les services disponibles.

Dans cette approche, le problème de la composition est défini comme suit :
étant donnés un service but, un service client et des services disponibles, sélectionner un ensemble de services, parmi les services disponibles et synthétiser un service médiateur tels que le produit asynchrone du service but et du service client est isomorphe au produit asynchrone du service client, du service médiateur et des services disponibles.

L'isomorphisme entre les deux systèmes est un isomorphisme modulo la communication entre les services disponibles. C'est-à-dire, que la communication entre les services disponibles est non observable.

Afin de résoudre ce problème, les auteurs proposent une réduction vers le problème de satisfiabilité d'une formule de la logique dynamique propositionnelle (PDL). Cependant, ils considèrent les restrictions suivantes :

- La médiation stricte : les services disponibles ne peuvent pas communiquer entre eux, ils ne peuvent communiquer qu'avec le médiateur,
- le service médiateur a un nombre borné de variables internes et d'états,
- les services disponibles sont déterministes,
- comportement bloquant : lorsqu'un service disponible envoie un message, deux cas sont possibles : le service se retrouve dans un état à partir duquel il ne peut plus effectuer d'action, ou bien l'unique action qu'il peut effectuer est la réception d'un message.
- accès borné : le nombre de messages envoyés par le client est borné par le nombre de messages qu'il reçoit. De plus, le nombre de recherches effectuées par le service but sur la base de données est borné par le nombre de messages envoyés par le client.

Lorsque toutes ces restrictions sont considérées, les auteurs ont prouvé que le problème de la composition peut être résolu en un temps doublement exponentiel.

Les auteurs se sont également intéressés au problème de la composition lorsque la relation de simulation est considérée [BFDP08]. Ils ont montré que le problème est dans EXPTIME. Néanmoins, le modèle considéré est plus abstrait que le modèle COLOMBO. En effet, dans ce modèle les auteurs

ne distinguent pas les actions internes des actions de communication et les transitions ne contiennent ni de préconditions ni d'effets.

Il existe d'autres variantes du problème de la composition auxquelles les auteurs se sont intéressés. Par exemple, dans [SPG08] le problème de la composition prend en considération le cas où les services peuvent devenir défaillants. Par exemple, si les services ou leur environnement changent d'états de façon inattendue. Pour résoudre ce problème, les auteurs proposent une réduction basée sur la relation de simulation entre automate non-déterministes. Ils proposent aussi des algorithmes, qui permettent la résolution du problème pour différentes failles possibles des systèmes.

Dans [GdLMP07], les auteurs s'intéressent au problème de la composition lorsque les services peuvent se déplacer. Cette composition est effectuée dans le contexte d'une gestion de situation d'urgence. Ce qui implique la prise en compte de contraintes d'environnement. Le problème de la composition dans cette approche consiste à synthétiser des médiateurs distribués au lieu d'un unique médiateur centralisé.

Approche de Pathak, Basu, Lutz et Honavar

Dans l'approche proposée par Pathak, Basu, Lutz et Honavar [PBLH06], les auteurs considèrent les services Web comme des documents BPEL. Cette approche est inspirée du modèle proposé par Berardi, Calvanese, De Giacomo, Hull et Mecella, décrit ci-dessous. Plus précisément, les services sont définis par des automates conditionnels qui peuvent effectuer des actions de communication ainsi que des actions internes.

Dans cette approche, le problème de la composition est défini comme suit :
étant donnés un service but est un ensemble de services disponibles, sélectionner un ensemble de services parmi les services disponibles et créer un service médiateur tels que la composition des services disponibles sélectionnés et du médiateur est bisimilaire au service but.

Composer des services est formellement représenté par un produit asynchrone pour les actions internes et synchrone pour les actions de communication.

Les auteurs proposent un algorithme correct et complet pour résoudre le problème de la composition. De plus, cette approche permet de déterminer les causes d'échec de la modélisation du médiateur. Une fois les causes déterminées il est possible d'aider l'utilisateur à reformuler sa spécification du service but. La complexité de l'algorithme proposé est exponentielle par rapport à la taille des services.

Approche de Mitra, Kumar et Basu

Dans les travaux de Mitra, Kumar et Basu [MKB07], les auteurs présentent une méthode pour résoudre le problème de la composition dans le cas où des interfaces de services sont considérées. Dans la pratique ces interfaces sont décrites par les langages BPEL, OWL-S et WSDL. Formellement, les services sont représentés par des automates d'entrée/sortie (I/O automata). Plus précisément, les actions effectuées par les services sont uniquement des actions de communication[6]. Le but est également représenté par un automate d'entrée/sortie.

Dans cette approche, le problème de la composition est défini comme suit :
étants donnés un service but et des services disponibles, vérifier l'existence d'un chorégraphe/médiateur qui communique avec les services disponibles tel que le service but est simulé par le produit asynchrone du chorégraphe et des services disponibles.

Les auteurs s'intéressent à deux types de chorégraphe, le chorégraphe simple et chorégraphe transducteur. Le chorégraphe simple est seulement capable de relier la sortie d'un service à l'entrée d'un autre service. Tandis que le chorégraphe transducteur est capable de stocker et de réutiliser les entrées et les sorties des autres services. Les auteurs proposent une méthode de résolution du problème de la composition dans ces deux cas. Néanmoins, deux restrictions sont considérées. Un service disponible ne peut communiquer qu'avec le chorégraphe[7] et le chorégraphe ne peut produire seul des entrées.

Afin de résoudre ce problème, les auteurs prouvent qu'il est nécessaire et suffisant de vérifier que le service but est simulé par la fermeture du produit asynchrone des services disponibles. Intuitivement, la fermeture de l'automate produit va contenir en plus des transitions effectuées par les servies disponibles, celles qui vont être effectuées par un service chorégraphe qui relie toutes les sorties d'un service avec les entrées d'un autre. Par conséquent, pour résoudre le problème de la composition, il suffit d'utiliser les procédures existantes pour la résolution du problème de la simulation entre automates. Cependant, les auteurs ne donnent pas de résultat concernant la complexité du problème.

[6] Les actions d'entrée sont les réceptions de messages et les actions de sortie sont les émissions de messages.

[7] Cette restriction est appelée stricte médiation dans le modèle COLOMBO.

Approche de Rao, Kungas et Matskin

Dans les travaux de Rao, Kungas et Matskin [RKM04], les auteurs proposent une méthode pour résoudre le problème de la composition dans le cas où des services Web sémantiques sont considérés. Les services sont formellement représentés par des axiomes de la logique linéaire et le but est représenté par un séquent de cette logique.

Dans cette approche, le problème de la composition est le suivant :
étant donnés un ensemble de services disponibles et un but, trouver une composition des services disponibles qui satisfasse le but.

Pour résoudre ce problème, les auteurs utilisent le prouveur de théorème de la logique linéaire. L'approche consiste à suivre les étapes suivantes :

1. Dans un premier temps, les services, décrit en langage DAML-S sont traduit en axiomes.
2. Ensuite, le but est spécifié sous forme de séquent à prouver.
3. Pour finir, le prouveur de théorème de la logique linéaire vérifie si le séquent peut être prouvé à partir des axiomes considérés. Dans le cas où le séquent a été prouvé et que la preuve est générée, un processus de calcul est extrait de la preuve. Ce processus de calcul représente le service composé, il sera traduit dans le langage DAMLS ou BEPEL4WS.

La méthode proposée est correcte et complète. Cela est dû au fait que le fragment de la logique utilisée est correct et complet.

Dans tous les travaux cité ci-dessus, pour composer des services, il est nécessaire d'avoir un but à satisfaire. Ce dernier peut être représenté par un automate ou par une formule logique. Néanmoins, il existe d'autres approches qui s'intéressent à la composition sans avoir à formaliser le but. Parmi ces approches nous citons celle proposée par **Hamadi et Benatallah** [HB03].

Dans cette approche les services sont modélisés par des réseaux de Petri. Composer des services revient à créer un nouveau service en utilisant des opérateurs algébriques sur les services existants. Les opérateurs algébriques permettent de représenter des notions comme la séquence, l'alternative, l'itération, etc. La sémantique de ces expressions algébriques est donnée par un réseau de Petri.

L'objectif des auteurs est de créer un cadre formel pour la définition d'un service composé. Ce cadre étant basé sur les réseaux de Pétri, cela permet l'utilisation des outils de vérification existants afin d'analyser le service. Avec

cette formalisation il est aussi possible de vérifier si un service peut être remplacé par un autre. Le problème revient à déterminer si les réseaux de Petri qui les représentent sont équivalents. Les auteurs proposent également des propriétés sur les opérations de leur algèbre. Ces propriétés permettent de transformer une expressions algébrique en une autre expression ayant la même sémantique, mais de taille inférieure.

2.2.2 Notre approche

Dans les approches que nous avons décrites précédemment, sauf celle proposée par Hamadi et Benatalah, il y a deux catégories.

Dans la première catégorie [PTB05, RKM04], l'objectif est de créer un orchestrateur/médiateur qui coordonne des services existants de sorte à satisfaire une formule d'une logique donnée. Cette formule peut représenter une exigence de l'utilisateur. Dans un exemple d'agence de voyage, cette exigence peut concerner la destination du voyageur ou l'heure de son arrivée. La formule à satisfaire peut aussi représenter une propriété que le système doit satisfaire. Par exemple, le système ne doit pas atteindre des états bloquants ou le système doit être vivace.

Dans la seconde catégorie [BCG$^+$05, MKB07], la composition est basée sur le modèle d'un service souhaité appelé aussi service but. L'objectif de ces approches est de créer un orchestrateur/médiateur qui coordonne des services existants de sorte à réaliser les fonctionnalités du service but. Formellement, cela sera représenté par une relation d'équivalence entre deux systèmes.

Notre approche se situe dans la seconde catégorie. Plus précisément, elle est inspirée par la méthode présentée dans [BCG$^+$05]. Notre modèle contient les éléments suivants :

- un ensemble de variables propositionnels ,
- un service client qui représente toute, les communications qui peuvent être effectuées avec un client potentiel,
- un service but qui représente le service que nous désirons réaliser, et qui ne peut communiquer qu'avec le service client,
- des services disponibles et
- un service médiateur.

Comme dans l'approche de Bérardi et al. [BCG$^+$05], nous représentons les services par des automates qui peuvent effectuer des actions internes ainsi que des actions de communication. Aussi, nous considérons des conditions et des effets dans les transitions.

Nous avons considéré trois variantes de notre modèle. La première va-

riante (voir Chapitre 5) est une abstraction de notre modèle initial. Plus précisément, dans ce modèle, les actions de communication sont considérées comme des actions internes [BCF07, CGM06, BFDP08]. Dans la seconde variante (voir Chapitre 6), nous considérons à nouveau la communication entre les services [BCF09, BCF08a, BCF08b]. Cependant, cette communication est effectuée à travers des ports bornés, comme dans [BCG$^+$05]. Dans la troisième variante (voir Chapitre 7), nous considérons une communication synchrone entre les services [BCHK09], comme dans [PTB05]. C'est-à-dire, qu'un service ne peut envoyer de message que s'il existe un autre service pour le recevoir. Réciproquement, un service ne peut recevoir de message que s'il existe un service prêt à l'envoyer.

Le problème de la composition est également différent pour les trois variantes.

Dans la première variante, le problème de la composition consiste à déterminer l'équivalence entre deux systèmes. Le premier système est composé du service but et le second système est composé des services disponibles.

Dans la deuxième variante, le problème de la composition consiste à synthétiser un service médiateur tel que le système composé du service client et du service but soit équivalent à celui composé du service client, du service médiateur et des services disponibles. Nous verrons dans le Chapitre 6, que le problème de la composition se réduit à un problème de satisfaction d'une formule du μ-calcul [BCF08a]. L'avantage de cette réduction est qu'elle nous permet d'exprimer en plus de l'équivalence entre deux systèmes, le fait qu'un système vérifie certaines propriétés. Une de ces propriétés peut être la vivacité du système ou sa tolérance aux fautes.

Dans la troisième variante, le problème de la composition consiste à synthétiser une formule caractéristique pour la composition. Cette formule est satisfaite ssi le système composé du service but est équivalent au système composé des services de la communauté. La formule caractéristique concerne des propriétés que le client doit satisfaire. Par exemple, l'âge du client ou le type de la carte de paiement qu'il possède.

Nous nous intéressons à l'étude de la complexité, en termes de bornes inférieures et supérieures, du problème de la composition pour les trois variantes. Pour comparer les systèmes, nous utilisons : l'inclusion de trace, l'équivalence de traces, la simulation et la bisimulation.

2.2.3 Comparaison de notre approche avec les approches existantes

Comme nous l'avons souligné précédemment, notre approche est différente des approches [PTB05, RKM04] car elles n'appartiennent pas à la même catégorie. Ainsi, les auteurs de [PTB05, RKM04] considèrent pas le problème de la composition de la même façon. Cependant, d'un point de vue formel, un problème d'équivalence entre des automates peut être réduit à un problème de satisfaction de formule. En effet, comme nous le verrons dans le Chapitre 6, afin de résoudre notre problème de la composition, pour la bisimulation, nous le réduisons au problème de la satisfaction d'une formule du μ-calcul. Également, le problème de la composition présenté dans [BCG$^+$05], se réduit au problème de la satisfaction d'une formule de la logique dynamique propositionnelle.

Notre approche est aussi différente de celle proposée dans [MKB07]. Dans cette approche, les auteurs ne considèrent une abstraction des services dans laquelle les actions internes ne sont pas considérées. Les auteurs ne s'intéressent qu'aux actions de communication. De plus, dans cette abstraction, les conditions et les effets sur les transitions ne sont pas considérés. Une autre différence avec notre modèle est qu'ils considèrent que les services disponibles ne peuvent pas communiquer entre eux. Ce qui n'est pas le cas dans la réalité.

Comparons notre modèle avec le modèle COLOMBO [BCG$^+$05]. Comme pour le modèle COLOMBO, nous considérons que les services Web peuvent effectuer des actions internes ainsi que des actions de communication. La communication dans les deux modèles s'effectue à travers des ports. Les notions de service client, service but, services disponibles et service médiateur sont identiques dans les deux approches.

L'une des différences entre notre modèle et le modèle COLOMBO est que nous ne considérons pas une base de données partagée entre les services. La raison est que nous considérons que les services sont indépendants et n'utilisent pas nécessairement les mêmes données. Néanmoins, nous avons un ensemble de variables propositionnelles, partagé par les services. Cet ensemble représente les données du client, mais il peut être vu comme une abstraction de la base de données.

Une autre différence est le fait que dans COLOMBO les services considérés sont déterministes. En effet, toutes les actions effectuées par les services sont supposées observables. Ce qui n'est pas le cas dans la réalité.

De plus, dans le modèle COLOMBO les auteurs considèrent certaines restrictions, pour résoudre le problème de la composition. Par exemple, il

considèrent que la taille du médiateur est initialement connue et que les services ne peuvent pas communiquer entre eux (la médiation stricte). D'autre restrictions spécifiques aux service Web ont été considérées comme : le comportement bloquant et l'accès borné. Ces restrictions permettent de diminuer le nombre de transitions possibles à partir d'un état donné.

Nous n'avons pas considéré ces restrictions, spécifiques aux services Web car nous voulons apporter des résultats qui peuvent être utilisés dans d'autres domaines. Effectivement, le problème de la composition n'est pas spécifique aux services Web. Dans le domaine des systèmes multiagents, il existe des travaux [Cha07, MKB06] qui s'intéressent à la composition d'agents. Il existe également des travaux sur la composition de service dans le domaine de l'intelligence ambiante [Opr07, MM07].

La dernière différence est que nous n'avons pas considéré la même relation d'équivalence. Dans [BCG+05], les auteurs considèrent un isomorphisme entre le système contenant le service but et celui qui contient les services disponibles et le médiateur. L'isomorphisme entre les automates est une relation plus restrictive que les relations de préordre et d'équivalence que nous considérons. En effet, l'isomorphisme peut être vu comme une égalité entre deux systèmes, modulo le renommage des états d'un des systèmes par les états de l'autre. Dans notre approche, l'objectif est de pouvoir exécuter les actions dans l'ordre exigé par l'utilisateur mais pas forcément avec le même nombre d'états que le service but.

Cependant, dans l'approche proposée par Mitra et al. les auteurs considèrent un modèle proche de celui de COLOMBO avec la bisimulation comme relation d'équivalence entre les automates. Mais ils considèrent aussi les restrictions citées ci-dessus. Ce qui permet de réduire la complexité de leur problème.

Soulignons que dans [BCG+05], le problème de la composition de services est ouvert, dans le cas où les ports sont non bornés. C'est pour cette raison que nous avons étudié ce cas, dans le Chapitre 4. Nous avons montré que, dans ce cas, le problème de la composition est indécidable pour la bisimulation et l'équivalence de traces et qu'il est décidable pour l'inclusion de traces et la simulation.

2.3 Problème de la composition versus synthèse de contrôleurs

Dans la section précédente, nous avons présenté différentes approches pour résoudre le problème de la composition. Ensuite, nous les avons com-

parées à notre approche. Dans cette section, nous nous intéressons à un problème dual au problème de la composition, à savoir le problème de la synthèse de contrôleurs, pour les systèmes à événements discrets. Nous commençons par présenter brièvement quelques travaux dans ce domaine. Ensuite, nous expliquerons où réside la dualité entre le problème de la composition et le problème de contrôle.

2.3.1 Synthèse de contrôleurs

La théorie du contrôle des systèmes à événements discrets a été introduite par Ramadge et Wonham [RW89] et a fait l'objet de différents travaux de recherche [KG95, CS06]. Dans cette théorie, les systèmes à événements discrets représentent des systèmes dynamiques qui évoluent selon l'occurrence de certains événements. Concrètement, ces systèmes peuvent représenter : un réseau routier, un réseau de communication ou un système robotique. Formellement, ces systèmes sont représentés par des automates finis sur une alphabet d'événements. Par ailleurs, le comportement d'un système est représenté par le langage reconnu par l'automate.

Dans l'approche proposée par Ramadge et Wonham, le problème de la synthèse de contrôleurs consiste à déterminer l'existence d'un système, nommé contrôleur, qui a pour objectif de restreindre le comportement/langage d'un système donné, appelé *plant*, afin que le comportement de ce dernier soit inclus dans un comportement dit admissible. Formellement, le contrôleur est également représenté par un automate fini. Quant à la notion de contrôle elle est représentée par le produit synchrone du contrôleur et du *plant*. Cependant, le contrôleur ne peut ni contrôler ni observer tous les événements. D'où les notions de contraintes de contrôlabilité et d'observabilité, que le contrôleur doit satisfaire[8]. Ramadge et Wonham se sont également intéressés à la synthèse de contrôleur maximal. Ce dernier permet de restreindre le comportement du *plant* de sorte à avoir un comportement équivalent à un comportement souhaité. Il existe une autre variante de ce problème, celle qui consiste à restreindre le comportement du *plant* de sorte à obtenir un comportement appartenant à un intervalle de comportements. L'objectif de cette variante est d'éviter d'avoir un contrôleur qui n'exécute aucun événement.

Il existe d'autres variantes du problème de la synthèse de contrôleurs. Parmi elles, nous citons celle proposée par Arnold, Vincent et Walukiewicz [AVW03]. Dans cette approche, l'objectif est de contrôler le *plant* afin

[8] Ces notions sont définies formellement dans le Chapitre 6. Plus précisément, dans la définition 6.5.2.

de satisfaire une formule du μ-calcul. Cette formule, peut représenter par exemple des propriétés de vivacité de système. Dans le cas de programmes informatiques, une propriété de vivacité serait que le système termine. D'autres approches basées sur la logique sont décrites dans [FP07, PR05b, JK06].

Dans l'approche proposée dans [AVW03], ainsi que l'approche de base proposée par Ramadge et Wonham, le *plant* est représenté par un automate déterministe. Le cas où le *plant* est non déterministe a été considéré dans [ZKJ06, BK06, PR05a]. La spécification non déterministe est utile, pour la description d'un système d'un haut niveau d'abstraction. Une autre raison serait que certains événement effectués par le *plant* ne soit pas modélisés, à cause d'un manque d'information. Dans [ZKJ06], le problème de la synthèse de contrôleurs devient : étant donnés un *plant* et une spécification, déterminer l'existence d'un contrôleur qui restreint le *plant* pour obtenir un système bisimilaire à la spécification.

2.3.2 Relation entre composition et contrôle

Dans la section ci-dessus, nous avons vu qu'il existe des approches dans lesquelles le problème de la synthèse de contrôleurs est décrit comme suit : étant donnés un automate $\mathcal{P}$, représentant le *plant*, un automate $\mathcal{S}$, représentant la spécification à réaliser et des contraintes de contrôlabilité et d'observabilité, déterminer l'existence d'un automate $\mathcal{C}$, représentant le contrôleur tel que : $\mathcal{C}$ satisfait les contraintes et le produit **synchrone** de $\mathcal{P}$ et $\mathcal{C}$ est équivalent à $\mathcal{S}$. Ici, l'équivalence peut être l'équivalence de traces ou la bisimulation. Il est également possible de considérer des relations de préordre comme l'inclusion de traces et la simulation.

Une description abstraite[9] du problème de la composition, que nous considérons, serait la suivante : étant donnés un automate $\mathcal{D}$, représentant le système disponible[10] et un automate $\mathcal{B}$, représentant le but à réaliser, déterminer l'existence d'un automate $\mathcal{M}$, représentant le médiateur[11] tel que : le produit **asynchrone** de $\mathcal{D}$ et $\mathcal{M}$ est équivalent à $\mathcal{B}$.

La différence évidente entre ces deux problèmes est le produit considéré. Dans la description du problème de la synthèse de contrôleurs, le produit synchrone permet de **restreindre** le comportement du *plant* en utilisant le

[9] Dans le modèle que nous considérons, nous distinguons les actions de communication et les actions internes des services. De plus, nous considérons des préconditions et des effets, dans les transitions.

[10] L'automate $\mathcal{D}$ représente un ensemble de n services disponibles. Formellement c'est un produit asynchrone de n automates.

[11] Dans notre modèle, le médiateur permet aux services disponibles de communiquer. Il est représenté par un automate qui ne peut effectuer que des actions de communication.

contrôleur. En revanche, dans la description du problème de la composition, le produit asynchrone permet **d'enrichir** le comportement du système disponible en utilisant le médiateur. C'est pour cette raison que nous pouvons dire que le problème de la composition est le dual du problème de la synthèse de contrôleurs.

Dans le Chapitre 6, nous donnons le détail de la réduction du problème de la composition vers le problème de la synthèse de contrôleurs, en considérant un problème de la composition moins abstrait que celui décrit ci-dessus. L'idée intuitive de la réduction est la suivante. Nous considérons un médiateur, appelé large médiateur. Ce dernier, peut exécuter toutes les actions à partir de son état initial. L'objectif est de contrôler le système composé de ce médiateur et du système disponible, en considérant que les actions effectuées par le large médiateur sont les seules actions observables et contrôlables.

2.4 Conclusion

Dans ce chapitre, nous avons d'abord présenté les différents langages utilisés pour la description de services Web, à savoir WSDL,WSCI,WSCDL et BPEL4WS. Ensuite, nous avons décrit les différentes approches formelles pour la résolution du problème de la composition de services. Plus précisément, pour chacune de ses approches, nous avons décrit la modélisation des services considérés, la formulation du but à satisfaire ainsi que le problème de la composition. Pour finir, nous avons comparé le problème de la composition avec le problème de la synthèse de contrôleurs, qui est largement étudié. Dans le Chapitre 6 nous donnons le détail de la réduction du problème de la composition vers le problème de la synthèse de contrôleurs.

Chapitre 3

Automates et réseaux de Petri

Le modèle que nous considérons pour représenter les services Web est basé sur les automates. Plus précisément, les services sont représentés par des automates communicants conditionnels (ACC) [BCF08b, BCG+05]. Cependant, nous verrons dans les Chapitres 4, 5 et 6, qu'il est possible d'associer aux ACC des réseaux de Petri ayant la même exécution[1]. Dans ce chapitre, nous faisons d'abord un rappel sur les automates et leur produits. Plus exactement, on s'intéresse aux produits synchrone et asynchrone. On s'intéresse également aux relations d'équivalences et de préordres suivantes : l'équivalence de traces, la bisimulation, l'inclusion de traces et la simulation. Ensuite, nous définissons les notions d'automate communicant conditionnel (ACC) et d'exécution d'ACC. Enfin, nous donnons des rappels sur les réseaux de Petri.

3.1 Les automates

Un automate, ou machine à états, est une machine abstraite utilisée en théorie de la calculabilité et dans l'étude des langages formels. Un automate est constitué d'états et de transitions [Koz97]. Un *état* de l'automate est une description instantanée de cet automate, elle donne les informations nécessaires pour déterminer comment l'automate évolue à partir de cet état. Les *transitions* sont des changements d'états, elles peuvent arriver spontanément ou comme réponse à des entrées externes. Considérons un

[1] Ici, exécution des réseaux veut dire graphe de marquage des réseaux.

ensemble fini d'actions Σ.

Définition 3.1.1 (**Automate**). Un *automate* sur Σ est une structure $\mathcal{A} = (Q, q_0, \rightarrow)$ telle que :
- Q est un ensemble d'états,
- q_0 est l'état initial et
- $\rightarrow \subseteq Q \times \Sigma \times Q$ est la relation de transition.

On dit que l'automate $\mathcal{A}$ est *fini* lorsque son ensemble d'états Q est fini. La taille d'un automate $\mathcal{A}$ sera la somme du cardinal de Q et du cardinal de $\rightarrow$. Plus précisément, $Taille(\mathcal{A}) = Card(Q) + Card(\rightarrow)$ avec $Card(\rightarrow) \leq Card(Q \times \Sigma \times Q)$. Pour tout $\Sigma' \subseteq \Sigma$, la relation $\rightarrow_{\mathcal{A}}^{\Sigma'} \subseteq Q \times Q$ décrit le changement d'état de l'automate suite à l'exécution d'une action de Σ'. Formellement, $(q, q') \in \rightarrow_{\mathcal{A}}^{\Sigma'}$ ssi il existe $a \in \Sigma'$ tel que $(q, a, q') \in \rightarrow$. On note $(q, q') \in \rightarrow_{\mathcal{A}}^{\Sigma'}$ par $q \rightarrow_{\mathcal{A}}^{\Sigma'} q'$. On note $\rightarrow_{\mathcal{A}}^{\Sigma'\star}$ la fermeture reflexive transitive de $\rightarrow_{\mathcal{A}}^{\Sigma'}$. Pour tout $\Sigma' \subseteq \Sigma$, nous dirons que $\mathcal{A}$ *boucle* sur Σ' ssi pour tout $a \in \Sigma'$, $\rightarrow_{\mathcal{A}}^{\{a\}} = Id_Q{}^2$. Un automate sera représenté par un graphe orienté. Les noeuds du graphe sont les états de l'automate et les arcs sont ses transitions. L'état initial sera représenté par un noeud particulier dans le graphe, désigné par un arc entrant. Plus précisément, cet arc n'aura aucun état de départ et son état d'arrivé sera l'état initial.

Remarque 3.1.2. Concernant les états finaux des automates, notons que :

1. dans la définition 3.1.1 l'ensemble des états finaux n'est pas représenté. Cela signifie que tous les états sont finaux. Cependant, dans les Chapitres 6, 7 il est nécessaire de représenter les états finaux. Dans ce cas nous ajoutons à la définition des automates, un ensemble $F \subseteq Q$. Soulignons, que dans le reste des chapitres, le fait de considérer ou non l'ensemble F ne change pas les résultats obtenus et
2. dans le graphe qui représente l'automate, les états finaux sont représentés par des états avec un arc sortant.

Définition 3.1.3 (**Automate déterministe**). Soit $\mathcal{A} = (Q, q_0, \rightarrow)$ un automate sur Σ. On dit que l'automate $\mathcal{A}$ est *déterministe* lorsque pour tout état $q \in Q$ et pour toute action $a \in \Sigma$, il existe au plus un état $q' \in Q$ tel que $(q, a, q') \in \rightarrow$.

Exemple 3.1.4. *Considérons l'automate fini et déterministe* $\mathcal{A}^1 = (\{q_0^1, q_1^1\}, q_0^1, \rightarrow_{\mathcal{A}^1})$ *sur* $\Sigma = \{a_1, a_2, a_3\}$ *tel que :* $\rightarrow_{\mathcal{A}^1} = \{(q_0^1, a_1, q_1^1), (q_1^1, a_2, q_1^1), (q_1^1, a_3, q_0^1)\}$.
L'automate $\mathcal{A}^1$ *est représenté par le graphe orienté de la Figure 3.1.*

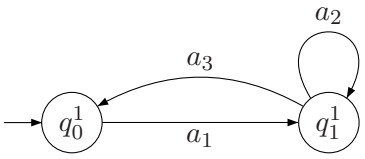

FIG. 3.1 – Automate fini $\mathcal{A}^1$

Définition 3.1.5 (Trace d'automate). Soient Σ' un ensemble fini d'actions et $\mathcal{A} = (Q, q_0, \rightarrow)$ un automate sur Σ. *Un chemin* pour $\mathcal{A}$ modulo Σ' est une séquence 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 :

$$q_i \rightarrow_{\mathcal{A}}^{\Sigma'^\star} \circ \rightarrow_{\mathcal{A}}^{\{a_{i+1}\}} \circ \rightarrow_{\mathcal{A}}^{\Sigma'^\star} q_{i+1} \ \forall i \in \{0, \ldots, n-1\}.$$

Le mot $a_1, a_2, \ldots, a_n$ est sa *trace*. Le mot vide est noté ϵ. L'ensemble de tous les mots qu'on peut construire à partir de Σ est noté Σ^*. L'ensemble des traces de tous les chemins pour $\mathcal{A}$ modulo Σ' est noté $Tr_{\Sigma'}(\mathcal{A})$. Sachant que dans notre définition d'automate tous les états sont finaux alors le mot vide appartient à l'ensemble des traces de tous les automates. Formellement, pour tout Σ', $\epsilon \in Tr_{\Sigma'}(\mathcal{A})$. Nous utilisons également la notation w^n, $w \in \Sigma^*$ et $n \in \mathbb{N}$ pour représenter un mot w^n construit par induction à partir du mot w de la façon suivante :

$$w^n = \begin{cases} \epsilon & \text{si } n = 0 \\ w^{n-1}w & \text{sinon} \end{cases}$$

Exemple 3.1.6. *Considérons l'automate $\mathcal{A}^1$ représenté dans la Figure 3.1. L'ensemble des traces de $\mathcal{A}^1$ modulo l'ensemble vide est $Tr_\emptyset(\mathcal{A}^1) = \{(a_1 a_2^n a_3)^m \mid n, m \in \mathbb{N}\}$. L'ensemble des traces de $\mathcal{A}^1$ modulo $\{a_1\}$ est $Tr_{\{a_1\}}(\mathcal{A}^1) = \{(a_2^n a_3)^m \mid n, m \in \mathbb{N}\}$.*

3.1.1 Produit d'automates

Le produit d'automates est utilisé pour représenter un système composé de plusieurs automates, définis sur un même ensemble d'actions. Ils existent deux types de produits : asynchrone et synchrone. Le produit asynchrone

[2] Id représente la relation identité

est utilisé lorsque les automates du système sont indépendants (ou concurrents). Le produit synchrone est, lui, utilisé lorsque les automates doivent se synchroniser les uns avec les autres, c'est -à-dire, qu'un automate ne peut effectuer une transition que si l'un des autres automates l'exécute aussi. Il est également possible d'avoir un produit dans lequel certaines actions seulement doivent être synchronisés, c'est ce qu'on appellera la synchronisation sur un ensemble d'actions. Soient $\mathcal{A}^1 = (Q^1, q_0^1, \rightarrow_{\mathcal{A}^1})$ et $\mathcal{A}^2 = (Q^2, q_0^2, \rightarrow_{\mathcal{A}^2})$ deux automates sur Σ.

Définition 3.1.7 (**Produit asynchrone d'automates**)**.** Nous notons par $\mathcal{A}^1 \otimes \mathcal{A}^2$ le *produit asynchrone* de $\mathcal{A}^1$ et $\mathcal{A}^2$, c'est-à-dire l'automate $\mathcal{A} = (Q, q_0, \rightarrow)$ sur Σ tel que :

- $Q = Q^1 \times Q^2$,
- $q_0 = (q_0^1, q_0^2)$ et
- $\rightarrow \subseteq Q \times \Sigma \times Q$ est la relation de transition définie par $((q_1^1, q_1^2),\ a,\ (q_2^1, q_2^2)) \in \rightarrow$ ssi $(q_1^1, a, q_2^1) \in \rightarrow_{A^1}$ et $q_2^2 = q_1^2$ ou $(q_1^2, a, q_2^2) \in \rightarrow_{A^2}$ et $q_2^1 = q_1^1$.

Le produit asynchrone d'automates est une opération associative.

Définition 3.1.8 (**Produit synchrone d'automates**)**.** Nous notons par $\mathcal{A}^1 \times \mathcal{A}^2$ le *produit synchrone* de $\mathcal{A}^1$ et $\mathcal{A}^2$, c'est-à-dire l'automate fini $\mathcal{A} = (Q, q_0, \rightarrow)$ sur Σ tel que :

- $Q = Q^1 \times Q^2$,
- $q_0 = (q_0^1, q_0^2)$ et
- $\rightarrow \subseteq Q \times \Sigma \times Q$ est la relation de transition définie par $((q_1^1, q_1^2),\ a,\ (q_2^1, q_2^2)) \in \rightarrow$ ssi $(q_1^1, a, q_2^1) \in \rightarrow_{A^1}$ et $(q_1^2, a, q_2^2) \in \rightarrow_{A^2}$.

Remarque 3.1.9. Il est possible que l'automate $\mathcal{A}^1 = (Q^{\mathcal{A}^1},\ q_0^{\mathcal{A}^1},\ \rightarrow_{\mathcal{A}^1})$ soit défini sur Σ^1 et l'automate $\mathcal{A}^2 = (Q^{\mathcal{A}^2},\ q_0^{\mathcal{A}^2},\ \rightarrow_{\mathcal{A}^2})$ sur Σ^2 tels que $\Sigma^1 \neq \Sigma^2$.

1. dans cette thèse, les produits synchrone et asynchrone des deux automates seront définis en considérant $\mathcal{A}^1$ comme un automate sur $\Sigma^1 \cup \Sigma^2$ tel que pour toute action $a \in \Sigma^2$, $(q, a, q') \notin \rightarrow_{\mathcal{A}^1}$. De la même façon, $\mathcal{A}^2$ sera considéré comme un automate sur $\Sigma^1 \cup \Sigma^2$ tel que pour toute action $a \in \Sigma^1$, $(q, a, q') \notin \rightarrow_{\mathcal{A}^2}$.
2. Pour certains auteurs [LS00], si $\Sigma^1 \neq \Sigma^2$ alors pour toute action $a \in \Sigma^1 \setminus \Sigma^2$, $((q_1^1, q_1^2),\ a,\ (q_2^1, q_2^2)) \in \rightarrow_{\mathcal{A}^1 \times \mathcal{A}^2}$ ssi $(q_1^1, a, q_2^1) \in \rightarrow_{A^1}$ et $q_2^2 = q_1^2$, pour toute action $a \in \Sigma^2 \setminus \Sigma^1$, $((q_1^1, q_1^2),\ a,\ (q_2^1, q_2^2)) \in \rightarrow_{\mathcal{A}^1 \times \mathcal{A}^2}$ ssi $(q_1^2, a, q_2^2) \in \rightarrow_{A^2}$ et $q_2^1 = q_1^1$. et pour toute action $a \in \Sigma^1 \cap \Sigma^2$, $((q_1^1, q_1^2),\ a,\ (q_2^1, q_2^2)) \in \rightarrow$ ssi $(q_1^1, a, q_2^1) \in \rightarrow_{A^1}$ et $(q_1^2, a, q_2^2) \in \rightarrow_{A^2}$.

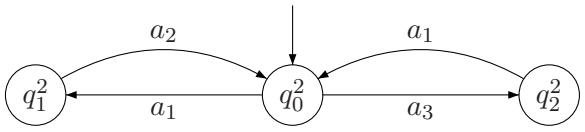

FIG. 3.2 – Automate fini $\mathcal{A}^2$

Il est possible d'avoir un produit d'automates synchrone pour un ensemble d'actions et asynchrone pour d'autres.

Définition 3.1.10 (Produit synchrone sur Σ'). Soit un ensemble d'actions $\Sigma' \subseteq \Sigma$, nous notons par $\mathcal{A}^1 \times^{\Sigma'} \mathcal{A}^2$ le *produit synchrone sur* Σ' de $\mathcal{A}^1$ et $\mathcal{A}^2$, c'est-à-dire l'automate fini $\mathcal{A} = (Q, q_0, \rightarrow)$ sur Σ tel que :

- $Q = Q^1 \times Q^2$,
- $q_0 = (q_0^1, q_0^2)$ et
- $\rightarrow \subseteq Q \times \Sigma \times Q$ est la relation de transition définie par $((q_1^1, q_1^2), a, (q_2^1, q_2^2)) \in \rightarrow$ ssi une des trois conditions suivantes est satisfaite :
  (1) $a \in \Sigma'$, $(q_1^1, a, q_2^1) \in \rightarrow_{A^1}$ et $(q_1^2, a, q_2^2) \in \rightarrow_{A^2}$,
  (2) $a \in \Sigma \setminus \Sigma'$,$(q_1^1, a, q_2^1) \in \rightarrow_{A^1}$ et $q_2^2 = q_1^2$ et
  (3) $a \in \Sigma \setminus \Sigma'$, $(q_1^2, a, q_2^2) \in \rightarrow_{A^2}$ et $q_2^1 = q_1^1$.

Exemple 3.1.11. *Considérons l'automate fini $\mathcal{A}^1$ représenté dans la Figure 3.1 et l'automate fini $\mathcal{A}^2$ représenté dans la Figure 3.2. Le produit asynchrone de ces deux automates est représenté dans la Figure 3.3. Dans cette figure les arcs pointillés représentent les transitions effectuées par l'automate $\mathcal{A}^1$ et les arcs pleins représentent les transitions effectuées par l'automate $\mathcal{A}^2$. Le produit synchrone de $\mathcal{A}^1$ et $\mathcal{A}^2$ est représenté dans la Figure 3.4.*

3.1.2 Equivalences et préordres

Comme nous l'avons précisé dans le Chapitre 2, pour définir le problème de la composition de services nous devons comparer le comportement de deux systèmes. D'une part nous considérons le système qui représente le but du client et d'autre part nous considérons le système qui représente les services disponibles. Il est donc nécessaire de définir formellement le type d'équivalences et/ou préordres que nous considérons. Dans la théorie des

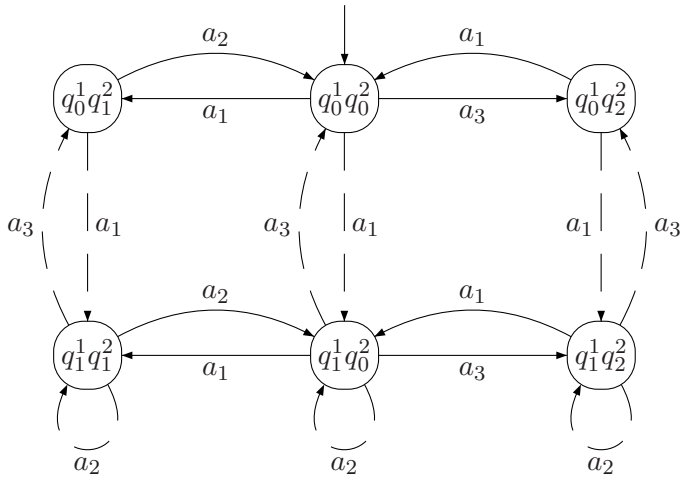

FIG. 3.3 – Automate fini $\mathcal{A}^1 \otimes \mathcal{A}^2$

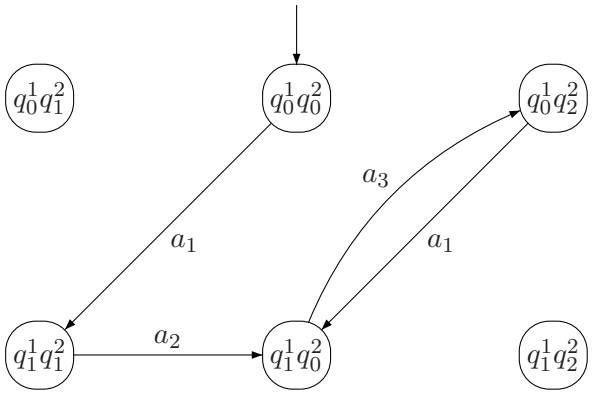

FIG. 3.4 – Automate fini $\mathcal{A}^1 \times \mathcal{A}^2$

systèmes concurrents, dans le calcul de processus ainsi que dans la vérification automatique de systèmes, certaines relations de préordre et d'équivalences entre processus ont été considérées [vG90, GV92, BIM95]. De telles relations sont par exemple utilisées pour établir qu'une implémentation d'un système est correcte relativement à une spécification. Le problème de la composition que nous traitons, peut être considéré avec tout type d'équivalence, comme par exemple toutes celles répertoriées et organisées sous forme de hiérarchie dans [vG90]. Cependant, nous nous intéresserons dans ce document aux équivalences et aux préordres les plus étudiés : l'équivalence de traces, l'inclusion de traces, la bisimulation et la simulation.

Inclusion et équivalence de traces

Selon [Hoa78, Hoa85], la trace du comportement d'un processus est une séquence de symboles représentant l'enregistrement des événements que le processus avait engagé à un moments donnée. On peut imaginer un observateur avec un bloc-notes qui observe le processus et note le nom de chaque événement lorsqu'il se produit. Si deux événements se produisent simultanément l'observateur enregistrera un des deux en premier ensuite le second. Soient $\mathcal{A}^1 = (Q^1, q_0^1, \rightarrow_{\mathcal{A}^1})$ et $\mathcal{A}^2=(Q^2,q_0^2,\rightarrow_{\mathcal{A}^2})$ des automates sur Σ.

Définition 3.1.12 (**Inclusion de traces**)**.** Soit un ensemble d'actions $\Sigma' \subseteq \Sigma$, nous dirons que $\mathcal{A}^1$ est *inclus pour la trace* dans $\mathcal{A}^2$ *modulo* Σ', noté $\mathcal{A}^1 \subseteq_{tr} \mathcal{A}^2 \quad (\Sigma')$, lorsque $Tr_{\Sigma'}(\mathcal{A}^1) \subseteq Tr_{\Sigma'}(\mathcal{A}^2)$. Lorsque $\Sigma' = \emptyset$, nous dirons que $\mathcal{A}^1$ est *inclus pour la trace* dans $\mathcal{A}^2$, noté $\mathcal{A}^1 \subseteq_{tr} \mathcal{A}^2$.

Exemple 3.1.13. *Considérons les automates $\mathcal{A}^1$ et $\mathcal{A}^2$ représentés dans la Figure 3.5. Dans ce cas, $Tr_\emptyset(\mathcal{A}^1) = \{\epsilon, a, a.b\}$ et $Tr_\emptyset(\mathcal{A}^2) = \{\epsilon, a, c, a.b\}$. Par conséquent, $Tr_\emptyset(\mathcal{A}^1) \subseteq Tr_\emptyset(\mathcal{A}^2)$. Ainsi, l'automate $\mathcal{A}^1$ est inclus pour la trace dans $\mathcal{A}^2$.*

Définition 3.1.14 (**Equivalence de traces**)**.** Soit un ensemble d'actions $\Sigma' \subseteq \Sigma$, nous dirons que $\mathcal{A}^1$ et $\mathcal{A}^2$ sont *équivalents pour la trace* modulo Σ', noté $\mathcal{A}^1 \equiv_{tr} \mathcal{A}^2 \quad (\Sigma')$, lorsque $Tr_{\Sigma'}(\mathcal{A}^1) = Tr_{\Sigma'}(\mathcal{A}^2)$. Lorsque $\Sigma' = \emptyset$, nous dirons que $\mathcal{A}^1$ et $\mathcal{A}^2$ sont *équivalents pour la trace*, noté $\mathcal{A}^1 \equiv_{tr} \mathcal{A}^2$.

Exemple 3.1.15. *Considérons les automates finis $\mathcal{A}^1$ et $\mathcal{A}^2$ représentés par la Figure 3.6. Les deux automates sont équivalents pour la trace. En effet, $Tr_\emptyset(\mathcal{A}_1) = Tr_\emptyset(\mathcal{A}_2) = \{\epsilon, a, ab\}$.*

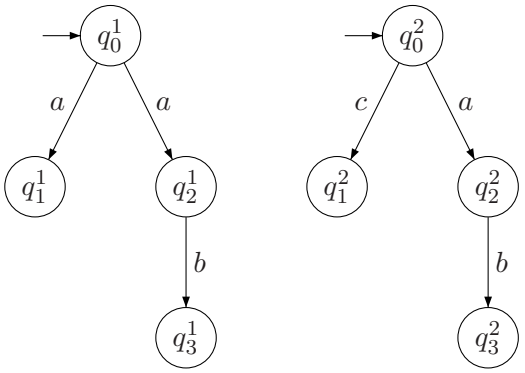

FIG. 3.5 – De gauche à droite, automates $\mathcal{A}^1$ et $\mathcal{A}^2$

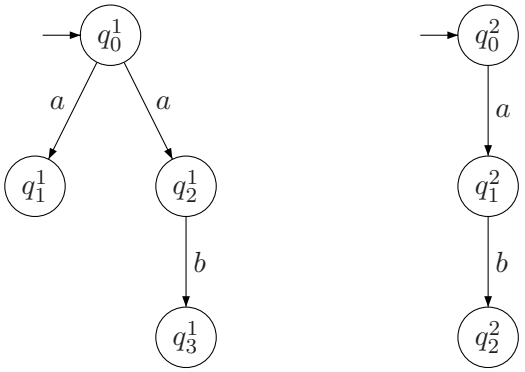

FIG. 3.6 – De gauche à droite, automates $\mathcal{A}^1$ et $\mathcal{A}^2$

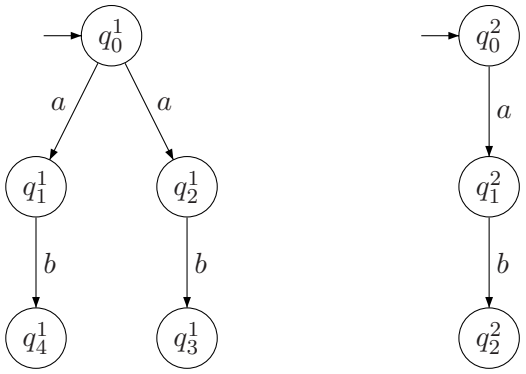

FIG. 3.7 – De gauche à droite, automates $\mathcal{A}^1$ et $\mathcal{A}^2$

Remarque 3.1.16. L'équivalence de traces est une relation d'équivalence (réflexive, symétrique et transitive), tandis que l'inclusion de traces est une relation de préordre (réflexive et transitive).

Simulation et bisimulation

Les notions de simulation et de bisimulation [Par81, Mil95], à la différence des notions d'inclusion de traces et d'équivalence de traces, se basent sur les états des systèmes à comparer. Soient $\mathcal{A}^1 = (Q^1, q_0^1, \rightarrow_{\mathcal{A}^1})$ et $\mathcal{A}^2 = (Q^2, q_0^2, \rightarrow_{\mathcal{A}^2})$ des automates sur Σ.

Définition 3.1.17 (Simulation). Soit un ensemble d'actions $\Sigma' \subseteq \Sigma$, une relation $Z \subseteq Q^1 \times Q^2$ telle que $(q_0^1, q_0^2) \in Z$ (aussi noté $q_0^1 \; Z \; q_0^2$) est appelée *simulation* entre $\mathcal{A}^1$ et $\mathcal{A}^2$ modulo Σ' ssi les conditions suivantes sont satisfaites pour tout $(q_1^1, q_1^2) \in Z$ et pour tout $a \in \Sigma \setminus \Sigma'$:

pour tout $q_2^1 \in Q^1$, si $q_1^1 \rightarrow_{\mathcal{A}^1}^{\Sigma'\star} \circ \rightarrow_{\mathcal{A}^1}^{\{a\}} \circ \rightarrow_{\mathcal{A}^1}^{\Sigma'\star} q_2^1$ alors il existe $q_2^2 \in Q^2$ tel que $q_1^2 \rightarrow_{\mathcal{A}^2}^{\Sigma'\star} \circ \rightarrow_{\mathcal{A}^2}^{\{a\}} \circ \rightarrow_{\mathcal{A}^2}^{\Sigma'\star} q_2^2$ et $(q_2^1, q_2^2) \in Z$,

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 $\mathcal{A}^1 \leq_{si} \mathcal{A}^2 \quad (\Sigma')$ et nous dirons que $\mathcal{A}^1$ est *simulé* par $\mathcal{A}^2$ modulo Σ'. Lorsque $\Sigma' = \emptyset$, nous dirons que Z est une *simulation* entre $\mathcal{A}^1$ et $\mathcal{A}^2$ et que $\mathcal{A}^1$ est *simulé* par $\mathcal{A}^2$, noté $\mathcal{A}^1 \leq_{si} \mathcal{A}^2$.

Définition 3.1.18 (Bisimulation). Une relation $Z \subseteq Q^1 \times Q^2$ telle que $(q_0^1, q_0^2) \in Z$ est appelée *bisimulation* entre $\mathcal{A}^1$ et $\mathcal{A}^2$ modulo Σ' ssi Z est une

simulations entre $\mathcal{A}^1$ et $\mathcal{A}^2$ modulo Σ' et Z^{-1} est une simulations entre $\mathcal{A}^2$ et $\mathcal{A}^1$ modulo Σ'.

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 $\mathcal{A}^1 \longleftrightarrow_{bi} \mathcal{A}^2 \quad (\Sigma')$ et nous dirons que $\mathcal{A}^1$ et $\mathcal{A}^2$ sont *bisimilaires* modulo Σ'. Lorsque $\Sigma' = \emptyset$, nous dirons que Z est une *bisimulation* entre $\mathcal{A}^1$ et $\mathcal{A}^2$ et que $\mathcal{A}^1$ et $\mathcal{A}^2$ sont *bisimilaires*, noté $\mathcal{A}^1 \longleftrightarrow_{bi} \mathcal{A}^2$.

Remarque 3.1.19. Il existe un ensemble d'actions $\Sigma' \subseteq \Sigma$ et des automates $\mathcal{A}^1$ et $\mathcal{A}^2$ tels que $\mathcal{A}^1 \leq_{si} A^2 \quad (\Sigma')$, $\mathcal{A}^2 \leq_{si} A^1 \quad ((\Sigma'))$ et non $\mathcal{A}^1 \longleftrightarrow_{bi} A^2 \quad (\Sigma')$.

En effet, il suffit de considérer $\Sigma' = \emptyset$ et les automates finis $\mathcal{A}^1$ et $\mathcal{A}^2$ représentés par la Figure 3.6.

Remarque 3.1.20. Notons que la bisimulation est une relation d'équivalence tandis que la simulation est une relation de préordre.

Rappelons que dans le cas où les automates finis $\mathcal{A}^1$ et $\mathcal{A}^2$ sont déterministes, l'inclusion de traces (resp. équivalence de traces) 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) et plus restrictive que celle de l'inclusion de traces (resp. l'équivalence de traces).

Exemple 3.1.21. *Dans la Figure 3.6, l'automate $\mathcal{A}^1$ est simulé par 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 3.7, les automates $\mathcal{A}^1$ et $\mathcal{A}^2$ sont bisimilaires.*

Pour certaines preuves, dans les chapitres suivants, nous aurons besoin de comparer des automates en termes d'isomorphisme. C'est pour cette raison que nous donnons ci-dessous la définition de cette relation d'équivalence. Cette relation est plus forte que la bisimulation (deux automates isomorphes sont bisimilaires). De ce fait, elle est plus forte que l'inclusion de trace, l'équivalence de trace et la simulation.

Définition 3.1.22 (Isomorphisme). Nous dirons que $\mathcal{A}^1$ et $\mathcal{A}^2$ sont *isomorphes*, noté $\mathcal{A}^1 \simeq_{iso} \mathcal{A}^2$, ssi il existe une bijection $g : Q^1 \to Q^2$ telle que $g(q_0^1) = q_0^2$ et pour tout état q_1^1, $q_2^1 \in Q^1$ et pour toute action $a \in \Sigma$, $q_1^1 \to_{\mathcal{A}^1}^{\{a\}} q_2^1$ ssi $g(q_1^1) \to_{\mathcal{A}^2}^{\{a\}} g(q_2^1)$.

Définition 3.1.23 (Automate observable modulo Σ'). Soient $\mathcal{A} = (Q, q_0, \rightarrow)$ un automate sur Σ et $\Sigma' \subseteq \Sigma$ un ensemble fini d'actions. Soit $R \subseteq Q \times Q$ la relation définie par $q_1 R q_2$ ssi il existe $m \in \mathbb{N}$, s_0, ..., s_m $\in Q$ tels que $s_0 = q_1$, $s_m = q_2$ et pour tout $i \in \{1, \ldots, m\}$, $s_{i-1} \rightarrow_{\mathcal{A}}^{\Sigma'\star} s_i$ ou $s_i \rightarrow_{\mathcal{A}}^{\Sigma'\star} s_{i-1}$. La *classe d'équivalence* d'un élément $q_1 \in Q$ est l'ensemble $R(q_1) = \{q_2 \in Q \mid q_1 R q_2\}$. Nous appelons *automate observable* de $\mathcal{A}$ modulo Σ', l'automate fini $Obs_{\Sigma'}(\mathcal{A}) = (Q', q_0', \rightarrow')$ sur Σ tel que :

- $Q' = \{R(q) \mid q \in Q\}$
- $q_0' = R(q_0)$
- $\rightarrow' \subseteq Q' \times \Sigma \times Q'$ est la relation de transition définie par $(q_1', a, q_2') \in \rightarrow'$ ssi il existe des états q_1 et $q_2 \in Q$ tels que $q_1' = R(q_1)$, $q_2' = R(q_2)$ et $q_1 \rightarrow_{\mathcal{A}}^{\{a\}} q_2$.

Définition 3.1.24 (Isomorphisme modulo Σ'). Pour tout $\Sigma' \subseteq \Sigma$, nous dirons que $\mathcal{A}^1$ et $\mathcal{A}^2$ sont *isomorphes modulo* Σ', noté $\mathcal{A}^1 \simeq_{iso} \mathcal{A}^2 \quad (\Sigma')$, ssi $Obs_{\Sigma'}(\mathcal{A}^1)$ et $Obs_{\Sigma'}(\mathcal{A}^2)$ sont isomorphes.

3.2 Automates communicants conditionnels

Le modèle que nous avons choisi pour représenter les services Web est basé sur les automates communicants conditionnels (ACC) [BCF08b, BCF08a]. Ces derniers sont des automates dans lesquels des actions de communication peuvent être exécutées. De plus, la transition d'un état à un autre peut nécessiter que des conditions soient satisfaites et que des effets soit appliqués. Les conditions et les effets sont représentés par des ensembles de littéraux sur un ensemble fini de formules atomiques, noté At. La notion de port que nous considérons est celle où les ports peuvent être vus comme des files. Nous noterons par $Port$, l'ensemble fini des ports que les services peuvent utiliser. Nous considérons deux types d'actions de communication : l'action de réception de message dans un port π, notée $?\pi$ et l'action d'envoi de message dans un port π, notée $!\pi$.

Définition 3.2.1 (Littéraux). L'ensemble des *littéraux* sur At est défini par $Li(At) = At \cup \{\neg p : p \in At\}$.

Définition 3.2.2 (Ensemble consistant). Un ensemble $J \in 2^{Li(At)}$ est *consistant* ssi pour tout $p \in At$, $p \notin J$ ou $\neg p \notin J$.

Définition 3.2.3 (Ensemble maximal). Soit $J, J' \in 2^{Li(At)}$. L'ensemble J est *maximal* pour J' ssi pour tout $r \in J$, $r \in J'$ ou $\neg r \in J$. Dans le cas où $J' = At$, nous dirons que J est maximal.

Définition 3.2.4 (**Automate communicant conditionnel**). Un automate communicant conditionnel sur $\Sigma \cup (\{!, ?\} \times Port)$ et At est une structure $\mathcal{A}c = (Q, q_0, \delta)$ telle que :

- Q est un ensemble d'états,
- q_0 est l'état initial,
- $\delta \subseteq 2^{Li(At)} \times Q \times (\Sigma \cup (\{!, ?\} \times Port)) \times Q \times 2^{Li(At)}$ est une relation de transition.

Dans tout le document les ensembles dans $2^{Li(At)}$ que nous considérons sont consistants. C'est à dire, si $(J, q, a, q', J') \in \delta$ alors J et J' sont consistants. Nous dirons que l'automate communicant conditionnel $\mathcal{A}c$ est *fini* ssi l'ensemble de ses état Q est fini. La taille d'un automate communicant conditionnel $\mathcal{A}c$ est la somme du cardinal de Q et du cardinal de δ. Plus précisément, $Taille(\mathcal{A}c) = Card(Q) + Card(\delta)$ avec $Card(\delta) \leq 2^{4 \times Card(At)} \times Card(Q)^2 \times (Card(\Sigma) + 2 \times Card(Port))$. Pour tout ensemble de littéraux $J, J' \in 2^{Li(At)}$, pour tout état $q, q' \in Q$ et pour toute action $a \in \Sigma \cup (\{!, ?\} \times Port)$, $(J, q, a, q', J') \in \delta$ signifie que les causes (préconditions) de l'exécution de l'action a à partir de l'état q pour atteindre l'état q' sont représentées par l'ensemble J et que les effets (postconditions) de l'exécution de cette action sont représentés par l'ensemble J'.

Pour simplifier, l'élément $(?, \pi)$ (resp. $(!, \pi)$) dans $\{!, ?\} \times Port$ est noté $?\pi$ (resp. $!\pi$).

Définition 3.2.5 (**ACC déterministe**). Soit $\mathcal{A}c = (Q, q_0, \delta)$ un automate communicant conditionnel sur $\Sigma \cup (\{!, ?\} \times Port)$ et At. Nous dirons que $\mathcal{A}c$ est *déterministe* lorsque pour tout état $q_1 \in Q$, pour toute action $a \in \Sigma$, et pour tout ensemble de littéraux $J_1, J_2 \in 2^{Li(At)}$, si $(J_1, q_1, a, q_2, J_2) \in \delta$ alors il n'existe pas d'état q_2' et d'ensemble de littéraux $J_1', J_2' \in 2^{Li(At)}$ tels que $q_2 \neq q_2'$, $J_1 \cup J_1'$ est consistant et que $(J_1', q_1, a, q_2', J_2') \in \delta$.

Exemple 3.2.6. *Nous considérons l'ensemble d'actions $\Sigma = \{a_1, a_2\}$, l'ensemble de ports $Port = \{\pi\}$, l'ensemble des formules atomiques $At = \{p_1, p_2\}$ et l'automate communicant conditionnel $\mathcal{A}c^1$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At, représenté dans la Figure 3.8. Cet automate est fini et déterministe.*

Remarque 3.2.7.

1. Observons que dans le cas où $At = \emptyset$, l'automate communicant conditionnel $\mathcal{A}c$ n'est autre qu'un automate défini sur l'ensemble d'actions $\Sigma \cup (\{!, ?\} \times Port)$. Formellement, soit $\mathcal{A}c = (Q, q_0, \delta)$ un automate communicant conditionnel sur $\Sigma \cup (\{!, ?\} \times Port)$ et $At = \emptyset$. Nous associons à $\mathcal{A}c$ l'automate $Auto(\mathcal{A}c) = (Q', q_0', \rightarrow')$ sur $\Sigma \cup (\{!, ?\} \times Port)$ tel que :

- $Q' = Q$,
- $q'_0 = q_0$ et
- $\rightarrow' \subseteq Q' \times (\Sigma \cup (\{!, ?\} \times Port)) \times Q'$ est la relation de transition définie par $(q, a, q') \in \rightarrow'$ ssi $(\emptyset, q, a, q', \emptyset) \in \delta$.

Ainsi, $Auto(\mathcal{A}c)$ a les mêmes états et transitions que $\mathcal{A}c$. La seul différence est le formalisme utilisé. Plus précisément, $Auto(\mathcal{A}c)$ est un automate et $\mathcal{A}c$ est un automate communicant conditionnel.

2. Réciproquement, dans le cas où $\mathcal{A}$ est un automate défini sur $\Sigma \cup (\{!, ?\} \times Port)$, il peut être vu comme un automate communicant conditionnel défini sur $\Sigma \cup (\{!, ?\} \times Port)$ et $At = \emptyset$. Formellement, nous associons à $\mathcal{A} = (Q, q_0, \rightarrow)$ un automate communicant conditionnel $Auto^{-1}(\mathcal{A}) = (Q', q'_0, \delta')$ tel que :
   - $Q' = Q$,
   - $q'_0 = q_0$ et
   - $\delta' \subseteq 2^{\emptyset} \times Q' \times (\Sigma \cup (\{!, ?\} \times Port)) \times Q' \times 2^{\emptyset}$ est la relation de transition définie par $(\emptyset, q, a, q', \emptyset) \in \delta'$ ssi $(q, a, q') \in \rightarrow$.
3. Il est clair que $Auto^{-1}(Auto(\mathcal{A}c)) = \mathcal{A}c$ et $Auto(Auto^{-1}(\mathcal{A})) = \mathcal{A}$.

3.2.1 Produit d'automates communicants conditionnels

Nous allons définir le produit synchrone ainsi que le produit asynchrone des automates communicants conditionnels. Ces deux types de produits sont une généralisation des produits synchrone et asynchrone que nous avons défini pour les automates. Soient $\mathcal{A}c^1 = (Q^1, q_0^1, \delta^1)$ et $\mathcal{A}c^2 = (Q^2, q_0^2, \delta^2)$ des automates communicants conditionnels sur $\Sigma \cup (\{!, ?\} \times Port)$ et At.

Définition 3.2.8 (**Produit asynchrone des ACC**)**.** Nous notons par $\mathcal{A}c^1 \circledast \mathcal{A}c^2$ le *produit asynchrone* de $\mathcal{A}c^1$ et $\mathcal{A}c^2$, c'est-à-dire l'automate communicant conditionnel $\mathcal{A}c = (Q, q_0, \delta)$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At tel que :

- $Q = Q^1 \times Q^2$,
- $q_0 = (q_0^1, q_0^2)$ et
- $\delta \subseteq 2^{Li(At)} \times Q \times (\Sigma \cup (\{!, ?\} \times Port)) \times Q \times 2^{Li(At)}$ est la relation de transition définie par $(J, (q_1^1, q_1^2), a, (q_2^1, q_2^2), J') \in \delta$ ssi $(J, q_1^1, a, q_2^1, J') \in \delta^1$ et $q_2^2 = q_1^2$ ou $(J, q_1^2, a, q_2^2, J') \in \delta^2$ et $q_2^1 = q_1^1$.

Le produit asynchrone d'automates communicants conditionnels est une opération associative.

Définition 3.2.9 (**Produit synchrone des ACC**)**.** Nous notons par $\mathcal{A}c^1 * \mathcal{A}c^2$,le *produit synchrone* de $\mathcal{A}c^1$ et $\mathcal{A}c^2$, c.-à-d. l'automate communicant conditionnel $\mathcal{A}c = (Q, q_0, \delta)$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At tel que :

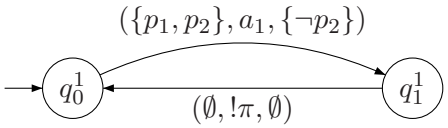

FIG. 3.8 – Automate communicant conditionnel $\mathcal{A}c^1$

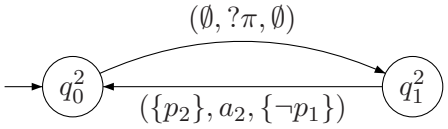

FIG. 3.9 – Automate communicant conditionnel $\mathcal{A}c^2$

- $Q = Q^1 \times Q^2$,
- $q_0 = (q_0^1, q_0^2)$ et
- $\delta \subseteq 2^{Li(At)} \times Q \times (\Sigma \cup (\{!, ?\} \times Port)) \times Q \times 2^{Li(At)}$ est la relation de transition définie par $(J, (q_1^1, q_1^2), a, (q_2^1, q_2^2), J') \in \delta$ ssi il existe J_1^1,J_2^1,J_1^2 et J_2^2 tels que $J_1^1 \cup J_1^2$ est consistant, $J_2^1 \cup J_2^2$ est consistant, $J = J_1^1 \cup J_1^2$, $J' = J_2^1 \cup J_2^2$, $(J_1^1, q_1^1, a, q_2^1, J_2^1) \in \delta^1$ et $(J_1^2, q_1^2, a, q_2^2, J_2^2) \in \delta^2$.

Exemple 3.2.10. *Nous considérons l'ensemble d'actions $\Sigma = \{a_1, a_2\}$, l'ensemble de ports $Port = \{\pi\}$, l'ensemble des formules atomiques $At = \{p_1, p_2\}$ et les automates communicants conditionnels $\mathcal{A}c^1$ et $\mathcal{A}c^2$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At, représentés respectivement dans la Figure 3.8 et la Figure 3.9. Le produit asynchrone de $\mathcal{A}c^1$ et $\mathcal{A}c^2$ est représenté dans la Figure 3.10. Les arcs pointillés représentent les transitions effectuées par $\mathcal{A}c^1$ et les arcs pleins les transitions effectuées par $\mathcal{A}c^2$. Quant au produit synchrone, des deux automates, il est représenté par l'automate constitué d'un seul état (son état initial) et d'aucune transition. En effet, l'ensemble des actions qui étiquettent les transitions de $\mathcal{A}c^1$ sont $\{a_1, !\pi\}$. Cet ensemble est disjoint de l'ensemble des actions qui étiquettent les transitions de $\mathcal{A}c^2$ et qui est $\{a_2, ?\pi\}$.*

Remarque 3.2.11. Comme pour les automates (voir remarque 3.1.9), il est possible que l'automate communicant $\mathcal{A}c^1$ soit défini sur $\Sigma^1 \cup (\{!, ?\} \times Port^1)$ et At^1 et l'automate $\mathcal{A}c^2$ sur $\Sigma^2 \cup (\{!, ?\} \times Port^2)$ et At^2, tels que $\Sigma^1 \neq \Sigma^2$,

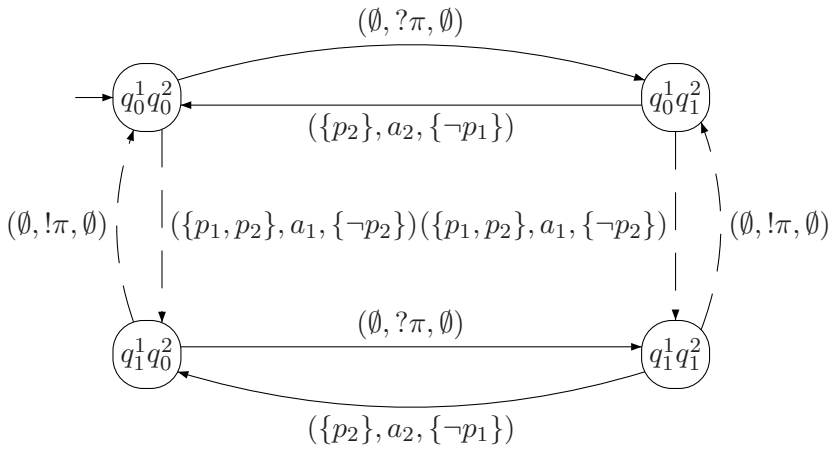

FIG. 3.10 – Automate communicant conditionnel $\mathcal{A}c^1 \circledast \mathcal{A}c^2$

$Port^1 \neq Port^2$ et $At^1 \neq At^2$. Dans cette thèse, les produits synchrone et asynchrone des deux automates seront définis en considérant $\mathcal{A}c^1 = (Q^1, q_0^1, \delta^1)$ comme un automate communicant conditionnel sur $(\Sigma^1 \cup \Sigma^2) \cup (\{!, ?\} \times (Port^1 \cup Port^2))$ et $At^1 \cup At^2$ tel que pour toute action $a \in \Sigma^2 \cup (\{!, ?\} \times Port^2)$, $(I_1^1, q_1^1, a, q_2^1, I_2^1) \notin \delta^1$. De la même façon, $\mathcal{A}^2 = (Q^2, q_0^2, \delta^2)$ est considéré comme un automate sur $(\Sigma^1 \cup \Sigma^2) \cup (\{!, ?\} \times (Port^1 \cup Port^2))$ et $At^1 \cup At^2$ tel que pour toute action $a \in \Sigma^1 \cup (\{!, ?\} \times Port^1)$, $(I_1^2, q_1^2, a, q_2^2, I_2^2) \notin \delta^2$.

3.2.2 Exécution des ACC

Pour représenter l'aspect dynamique des ACC, c'est-à-dire, l'évolution des formules atomiques et du contenu des ports, nous construisons des automates qui peuvent éventuellement être infinis. Dans la pratique, les ports n'ont pas une capacité illimitée : un port a un nombre maximal de messages qu'il peut stocker. Dans notre modèle, nous supposerons que tous les ports peuvent contenir au plus $k \in \mathbb{N}^* \cup \{\infty\}$ messages à la fois. Si $k \in \mathbb{N}^*$ alors le port est une file bornée de taille k, sinon le port est une file non bornée. De plus, nous supposerons que les ports sont initialement vides. Quant à l'initialisation des formules atomiques elle sera représentée par un ensemble maximal consistant de littéraux $I_0 \subseteq Li(At)$. Dans la suite du document, nous verrons que l'initialisation des ports ne modifie ni la décidabilité ni la

complexité des problèmes que nous traitons. Pour représenter le contenu des ports, nous utilisons les fonctions $\gamma : Port \to \mathbb{N}$. L'ensemble de toutes ces fonctions est noté Γ.

Définition 3.2.12 (Exécution des ACC). Soit $\mathcal{A}c = (Q, q_0, \delta)$ un automate communicant conditionnel sur $\Sigma \cup (\{!, ?\} \times Port)$ et At. Nous appelons *exécution* de $\mathcal{A}c$, l'automate $Exec(\mathcal{A}c, I_0) = (Q', q'_0, \to')$ défini sur $\Sigma \cup (\{!, ?\} \times Port)$ tel que :

- $Q' = \{(q, \gamma, I) \mid q \in Q, \gamma \in \Gamma$ et $I \subseteq Li(At)$ est une partie maximale consistante$\}$,
- $q'_0 = (q_0, \gamma_0, I_0)$ où $\forall \pi \in Port, \gamma_0(\pi) = 0$ et
- $\to' \subseteq Q' \times \Sigma \cup (\{!, ?\} \times Port) \times Q'$ est la relation de transition définie par $((q, \gamma, I), a, (q', \gamma', I')) \in \to'$ ssi il existe une transition $(J, q, a, q', J') \in \delta$ telle 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 que l'une des trois conditions suivantes est satisfaite :
  (1) $a \in \Sigma$ et $\gamma = \gamma'$,
  (2) $a = ?\pi$, $\gamma(\pi) > 0$, $\gamma'(\pi) = \gamma(\pi) - 1$ et $\gamma'(\pi') = \gamma(\pi)$ pour $\pi' \neq \pi$ et
  (3) $a = !\pi$, $\gamma(\pi) < k$, $\gamma'(\pi) = \gamma(\pi) + 1$ et $\gamma'(\pi') = \gamma(\pi)$ pour $\pi' \neq \pi$.

Intuitivement, dans la définition de $Exec(\mathcal{A}c, I_0)$, pour tout port π, $\gamma(\pi)$ représente le nombre actuel de messages dans le port π et I représente la valeur actuelle des formules atomiques. Concernant les transitions à partir d'un état (q, γ, I) il n'est possible d'effectuer une transition vers un état (q', γ', I'), étiquetée par a que si d'une part les préconditions de l'action a sont satisfaites et que les effets de l'actions sont pris en compte dans l'état (q', γ', I'). D'autre part, des conditions implicites sur les actions de communications sont à satisfaire. Plus précisément, pour effectuer une réception de message par un port π il est nécessaire d'avoir au moins un message dans ce port. Une fois que le message est reçu, le nombre de messages dans le port π est décrémenté. De façon symétrique, pour envoyer un message dans un port π il est nécessaire que le port n'est pas atteint sa capacité maximale. Une fois que le message est envoyé le nombre de messages dans le port est incrémenté.

Remarque 3.2.13. Lorsque la taille des ports est bornée, un temps exponentiel par rapport à la taille de At et de $Port$ est suffisant pour construire l'automate fini $Exec(\mathcal{A}c, I_0)$. De plus la taille de l'automate fini $Exec(\mathcal{A}c, I_0)$ est exponentielle par rapport à la taille de At et de $Port$.

Exemple 3.2.14. *Nous considérons l'automate communicant conditionnel* $\mathcal{A}c^1$ *de la Figure 3.8 et nous supposons que* $k = 1$. *L'automate fini qui*

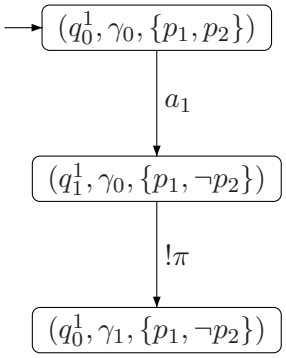

FIG. 3.11 – Automate fini $Exec(\mathcal{A}c^1, I_0)$

représente l'exécution de $\mathcal{A}c^1$ est représenté dans la Figure 3.11. Dans cette figure, $\gamma_0(\pi) = 0$ et $\gamma_1(\pi) = 1$ et initialement I_0 contient p_1 et p_2. Les actions a_1 et $!\pi$ ne peuvent être exécutées qu'une seule fois. La raison est que, la précondition de a_1 n'est plus satisfaite après son exécution et que l'exécution de $!\pi$ dépend de celle de a_1. Notons, que les états non accessibles à partir de l'état initial ne sont pas représentés dans la figure et que le nombre total d'états est égal à 2^4.

Remarque 3.2.15. Soit $\mathcal{A}c = (Q, q_0, \delta)$ un automate communicant conditionnel sur $\Sigma \cup (\{!, ?\} \times Port)$ et $At = \emptyset$. Rappelons que dans la remarque 3.2.7, nous avons associé à $\mathcal{A}c$ l'automate $Auto(\mathcal{A}c) = \mathcal{A} = (Q, q_0, \rightarrow)$ sur $\Sigma \cup (\{!, ?\} \times Port)$. Par abus de notation, nous considèrerons $Exec(\mathcal{A})$ au lieu de $Exec(\mathcal{A}c, \emptyset)$. De plus, $Exec(\mathcal{A}) = (Q', q'_0, \rightarrow')$ est un automate sur $\Sigma \cup (\{!, ?\} \times Port)$ défini comme suite :

- $Q' = \{(q, \gamma) \mid q \in Q \text{ et } \gamma : Port \rightarrow \mathbb{N}\}$,
- $q'_0 = (q_0, \gamma_0)$ où $\forall \pi \in Port, \gamma_0(\pi) = 0$ et
- $\rightarrow' \subseteq Q' \times \Sigma \cup (\{!, ?\} \times Port) \times Q'$ est la relation de transition définie par $((q, \gamma), a, (q', \gamma')) \in \rightarrow'$ ssi $(q, a, q') \in \rightarrow$ et que l'une des trois conditions suivantes est satisfaite :
  (1) $a \in \Sigma$ and $\gamma = \gamma'$,
  (2) $a = ?\pi$, $\gamma(\pi) > 0$, $\gamma'(\pi) = \gamma(\pi) - 1$ et $\gamma'(\pi') = \gamma(\pi)$ pour $\pi' \neq \pi$ et
  (3) $a = !\pi$, $\gamma(\pi) < k$, $\gamma'(\pi) = \gamma(\pi) + 1$ et $\gamma'(\pi') = \gamma(\pi)$ pour $\pi' \neq \pi$.

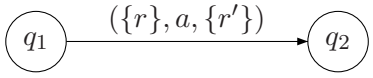

FIG. 3.12 – Une transition qui n'est pas de forme canonique.

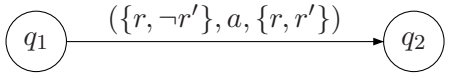

FIG. 3.13 – Une transition de forme canonique.

3.2.3 Forme canonique des ACC

Nous définissons un type particulier d'automates communicants conditionnels, les ACC de forme canonique. Dans ces automates, les conditions et les effets d'une transition, concernent les même variables propositionnelles. En d'autre terme, pour chaque transition, un littéral ou sa négation appartient à ses conditions ssi lui ou sa négation appartient aux effets. Dans ce qui suit, nous donnons la définition formelle d'un ACC de forme canonique et nous prouvons que tout ACC peut être associé à un ACC de forme canonique qui lui est isomorphe.

Définition 3.2.16 (**Transition de forme canonique**)**.** Soient les ensembles finis Σ, $Port$, At, I_0 et un automate communicant conditionnel fini $\mathcal{A}c = (Q, q_0, \delta)$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At. Nous dirons qu'une transition $(J, q, a, q', J') \in \delta$ est de *forme canonique* lorsque $J \subseteq (J' \cup \neg J')$ et $J' \subseteq (J \cup \neg J)$.

Exemple 3.2.17. *La transition* $(\{r\}, q_1, a, q_2, \{r'\})$ *de la Figure 3.12, n'est pas canonique car* $r \in J$ *et* $r \notin (J' \cup \neg J')$ *et* $r' \in J'$ *et* $r' \notin (J \cup \neg J)$. *La transition* $(\{r, \neg r'\}, q_1, a, q_2, \{r, r'\})$ *de la Figure 3.13 est canonique car* $(J \cup \neg J) = (J' \cup \neg J') = \{r, \neg r, r', \neg r'\}$.

Définition 3.2.18 (**ACC de forme canonique**)**.** un automate communicant conditionnel fini $\mathcal{A}c$ est de *forme canonique* lorsque toute ses transitions

sont de forme canonique.

Proposition 3.2.19. *Soient les ensembles finis Σ, $Port$, At, I_0 et un automate communicant conditionnel fini $\mathcal{A}c$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At. Il existe un automate communicant conditionnel $\mathcal{A}c'$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At, de forme canonique et de taille exponentielle par rapport à la taille de At tel que :*

$$Exec(\mathcal{A}c, I_0) \simeq_{iso} Exec(\mathcal{A}c', I_0)$$

Démonstration. Soient $J, J' \in 2^{Li(At)}$, $Completer(J, J') = J' \setminus (J \cup \neg J)$ et $Combinaison(J, J') = \{J'' \in 2^{Li(At)} \mid J''$ consistant et J'' maximal pour $Completer(J, J')\}$. Soit $I \in 2^{Li(At)}$ un ensemble maximal consistant. L'ensemble $J^I = \{r \in I \mid r \in Completer(J, J')$ ou $\neg r \in Completer(J, J')\}$ est l'unique ensemble J'' dans $Combinaison(J, J')$ tel que $J'' \subseteq I$. Soit $\mathcal{A}c = (Q, q_0, \delta)$ un automate communicant conditionnel sur $\Sigma \cup (\{!,?\} \times Port)$ et At. Nous construisons, à partir de $\mathcal{A}c$, l'automate communicant conditionnel $\mathcal{A}c' = (Q', q'_0, \delta')$ comme suit :

- $Q' = Q$,
- $q'_0 = q_0$,
- $\delta' \subseteq 2^{Li(At)} \times Q' \times (\Sigma \cup (\{!,?\} \times Port)) \times Q' \times 2^{Li(At)}$ est la relation définie par : $(J'_1, q_1, a, q_2, J'_2) \in \delta'$ ssi il existe une transition $(J_1, q_1, a, q_2, J_2) \in \delta$ et il existe $J \in Combinaison(J_1, J_2)$ avec :
  - $J'_1 = J_1 \cup J$ et
  - $J'_2 = J_2 \cup Completer(J_2, J_1)$.

Il est facile de vérifier que toute transition $(J'_1, q_1, a, q_2, J'_2) \in \delta'$ est canonique. Montrons que (1) $J'_1 \subseteq (J'_2 \cup \neg J'_2)$ et (2) $J'_2 \subseteq (J'_1 \cup \neg J'_1)$. Concernant le point (1). Deux cas sont possibles pour J_1 : soit $J_1 \subseteq (J_2 \cup \neg J_2)$ ou $J_1 \not\subseteq (J_2 \cup \neg J_2)$.

- Considérons le cas où $J_1 \subseteq (J_2 \cup \neg J_2)$. Dans ce cas, $Completer(J_1, J_2) = \emptyset$. Cela implique $J'_1 \subseteq (J_2 \cup \neg J_2)$. Sachant que $J_2 \subseteq J'_2$ on a $J'_1 \subseteq (J'_2 \cup \neg J'_2)$.
- Considérons le cas où $J_1 \not\subseteq (J_2 \cup \neg J_2)$. Dans ce cas, soit $r \in J_1$ et $r \notin (J_2 \cup \neg J_2)$. Donc, $r \in Completer(J_1, J_2)$. Ainsi, $r \in J'_2$. Ce qui implique $r \in (J'_2 \cup \neg J'_2)$. Par conséquent, $J'_1 \subseteq (J'_2 \cup \neg J'_2)$.

Concernant le point (2). Deux cas sont possibles pour J_2 : soit $J_2 \subseteq (J_1 \cup \neg J_1)$ ou $J_2 \not\subseteq (J_1 \cup \neg J_1)$. Dans le cas où $J_2 \subseteq (J_1 \cup \neg J_1)$, le raisonnement est le même que dans le cas où $J_1 \subseteq (J_2 \cup \neg J_2)$. Considérons le cas où $J_2 \not\subseteq (J_1 \cup \neg J_1)$. Dans ce cas, soit $r \in J_2$ et $r \notin (J_1 \cup \neg J_1)$. Donc, $r \in Completer(J_2, J_1)$. Sachant que J est maximal pour $Completer(J_2, J_1)$ on a $r \in J \cup \neg J$. Sachant que $J \subseteq J'_1$ on a $r \in (J'_1 \cup \neg J'_1)$. Par conséquent, $J'_2 \subseteq (J'_1 \cup \neg J'_1)$.

La taille de $\mathcal{A}c$ est égale à :

$card(Q)+ (card(Q)^2 \times card(\Sigma \cup (\{!,?\} \times Port))\times\ k^{card(At)})$ avec $k \in \mathbb{N}$

et la taille de $\mathcal{A}c'$ est égale à :

$$card(Q)+ (card(\delta)\times\ 2^{card(At)}).$$

Il s'en suit que la taille de $\mathcal{A}c'$ est exponentielle par rapport à la taille de At et polynomiale par rapport à la taille de Ac. Nous devons montrer que :

$$Exec(\mathcal{A}c, I_0) \simeq_{iso} Exec(\mathcal{A}c', I_0)$$

Soit g la fonction identité sur $Q \times \Gamma \times 2^{Li(At)}$. Concernant les états initiaux, il est évident que $g((q_0, \gamma_0, I_0)) = (q_0, \gamma_0, I_0)$. Nous allons prouver que pour toute action $a \in \Sigma \cup (\{!,?\} \times Port)$ et pour tout état $(q_1, \gamma_1, I_1), (q_2, \gamma_2, I_2) \in Q \times \Gamma \times 2^{Li(At)}$:

$$(q_1, \gamma_1, I_1) \rightarrow^a_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$$
$$\text{ssi}$$
$$(q_1, \gamma_1, I_1) \rightarrow^a_{Exec(\mathcal{A}c', I_0)} (q_2, \gamma_2, I_2)$$

($\Rightarrow$) Considérons l'implication de gauche à droite. Supposons que (q_1, γ_1, I_1) $\rightarrow^a_{Exec(\mathcal{A}c, I_0)}$ (q_2, γ_2, I_2). Deux cas sont possibles pour a : $a \in \Sigma$ ou $a \in \{!,?\} \times Port$. Considérons le cas où $a \in \Sigma$. Puisque $(q_1, \gamma_1, I_1) \rightarrow^a_{Exec(\mathcal{A}c, I_0)}$ (q_2, γ_2, I_2) alors il existe $J_1, J_2 \in 2^{Li(At)}$ tel que $(J_1, q_1, a, q_2, J_2) \in \delta$ avec $J_1 \subseteq I_1$, $I_2 = (I_1 \setminus \neg J_2) \cup J_2$ et $\gamma_2 = \gamma_1$. Par construction de $\mathcal{A}c'$, (J'_1, q_1, a, q_2, J'_2) $\in \delta'$ avec :

- $J'_1 = J_1 \cup J^I$ et
- $J'_2 = J_2 \cup Completer(J_2, J_1)$.

Puisque $J_1 \subseteq I_1$ et $J^I \subseteq I_1$ alors $J'_1 \subseteq I_1$. Sachant que $J_1 \subseteq I_1$ et que I_1 est consistant on a $Completer(J_2, J_1) \subseteq I_1$ et $\neg Completer(J_2, J_1) \cap \subseteq I_1 = \emptyset$. Par conséquent, $(I_1 \setminus \neg J_2) \cup J_2 = (I_1 \setminus \neg(J_2 \cup Completer(J_2, J_1))) \cup (J_2 \cup Completer(J_2, J_1))$. Donc $I_2 = (I_1 \setminus \neg J'_2) \cup J'_2$. Il en résulte par construction de $Exec(\mathcal{A}c', I_0)$ que $(q_1, \gamma_1, I_1) \rightarrow^a_{Exec(\mathcal{A}c', I_0)} (q_2, \gamma_2, I_2)$.

Dans le cas où $a = ?\pi, \pi \in Port$, le raisonnement est le même. L'unique différence est que $(q_1, \gamma_1, I_1) \rightarrow^{?\pi}_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$ implique que $\gamma_1(\pi) > 0$, $\gamma_2(\pi) = \gamma_1(\pi) - 1$ et pour tout $\pi' \neq \pi$, $\gamma_2(\pi') = \gamma_2(\pi')$. Par construction de $Exec(\mathcal{A}c', I_0)$ on a $(q_1, \gamma_1, I_1) \rightarrow^{?\pi}_{Exec(\mathcal{A}c', I_0)} (q_2, \gamma_2, I_2)$.

Dans le cas où $a = !\pi, \pi \in Port$, le raisonnement est le même. L'unique différence est que $\gamma_2(\pi) = \gamma_1(\pi) + 1$ et pour tout $\pi' \neq \pi$, $\gamma_2(\pi') = \gamma_2(\pi')$. Par construction de $Exec(\mathcal{A}c', I_0)$ on a $(q_1, \gamma_1, I_1) \rightarrow^{?\pi}_{Exec(\mathcal{A}c', I_0)} (q_2, \gamma_2, I_2)$.

($\Leftarrow$) Considérons l'implication de droite à gauche. Supposons que $(q_1, \gamma_1, I_1) \rightarrow^{a}_{Exec(\mathcal{A}c', I_0)} (q_2, \gamma_2, I_2)$. Deux cas sont possibles pour a : $a \in \Sigma$ ou $a \in \{!, ?\} \times Port$. Considérons le cas où $a \in \Sigma$. Puisque $(q_1, \gamma_1, I_1) \rightarrow^{a}_{Exec(\mathcal{A}c', I_0)} (q_2, \gamma_2, I_2)$ alors il existe $J'_1, J'_2 \in 2^{Li(At)}$ tel que $(J'_1, q_1, a, q_2, J'_2) \in \delta'$ avec $J'_1 \subseteq I_1$, $I_2 = (I_1 \setminus \neg J'_2) \cup J'_2$ et $\gamma_2 = \gamma_1$.

Par construction de $\mathcal{A}c'$, $(J'_1, q_1, a, q_2, J'_2) \in \delta'$ il existe $J_1, J_2 \in 2^{Li(At)}$ et il existe $J \in Combinaison(J_1, J_2)$ tels que :

- $J'_1 = J_1 \cup J$ et
- $J'_2 = J_2 \cup Completer(J_2, J_1)$.

Sachant que $J'_1 \subseteq I_1$ alors $J_1 \subseteq I_1$. Puisque $I_2 = (I_1 \setminus \neg J'_2) \cup J'_2$ et $J'_2 = J_2 \cup Completer(J_2, J_1)$ alors $I_2 = (I_1 \setminus \neg(J_2 \cup Completer(J_2, J_1))) \cup (J_2 \cup Completer(J_2, J_1))$. De plus, sachant que $J_1 \subseteq I_1$ et que I_1 est consistant on a $Completer(J_2, J_1) \subseteq I_1$ et $\neg Completer(J_2, J_1) \cap \subseteq I_1 = \emptyset$. Par conséquent, $I_2 = (I_1 \setminus \neg J_2) \cup J_2$. Par construction de $Exec(\mathcal{A}c, I_0)$ on a $(q_1, \gamma_1, I_1) \rightarrow^{a}_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$.

Dans le cas où $a = ?\pi, \pi \in Port$, le raisonnement est le même. L'unique différence est que $(q_1, \gamma_1, I_1) \rightarrow^{?\pi}_{Exec(\mathcal{A}c', I_0)} (q_2, \gamma_2, I_2)$ implique que $\gamma_1(\pi) > 0$, $\gamma_2(\pi) = \gamma_1(\pi) - 1$ et pour tout $\pi' \neq \pi$, $\gamma_2(\pi') = \gamma_2(\pi')$. Par construction de $Exec(\mathcal{A}c, I_0)$ on a $(q_1, \gamma_1, I_1) \rightarrow^{?\pi}_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$.

Dans le cas où $a = !\pi, \pi \in Port$, le raisonnement est le même. L'unique différence est que $\gamma_2(\pi) = \gamma_1(\pi) + 1$ et pour tout $\pi' \neq \pi$, $\gamma_2(\pi') = \gamma_2(\pi')$. Par construction de $Exec(\mathcal{A}c, I_0)$ on a $(q_1, \gamma_1, I_1) \rightarrow^{?\pi}_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$. □

3.3 Réseaux de Petri

Les réseaux de Petri ont été introduit dans [Pet62]. Dans cette thèse, nous somme intéressée par les réseaux de Petri étiquetés. Plus particulièrement par les réseaux de Petri ordinaires. En effet, comme nous le verrons dans les Chapitres 4, 5 et 6, il est possible d'associer aux ACC des Réseaux de Petri ordinaire ayant la même exécution[3]. Dans cette section, nous ne définissons que les notions nécessaires pour la compréhension du reste du document.

Définition 3.3.1 (**Réseau de Petri étiqueté**). Soit Σ un ensemble fini d'actions. Un *réseau de Petri étiqueté* sur Σ est une structure de la forme $\mathcal{N} = (P, T, F, W, l)$ où

- P est un ensemble fini de places,
- T est un ensemble fini de transitions,

[3]Ici, exécution des réseaux veux dire graphe de marquage des réseaux.

- $F \subseteq (P \times T) \cup (T \times P)$ est un ensemble fini d'arcs définissant *la relation de flot*,
- $W : F \to \mathbb{N}$ est une fonction de pondération pour les arcs et
- $l : T \to \Sigma$ est une fonction d'étiquetage des transitions.

Les Réseaux de Petri sont des graphes biparties, contenant deux types de sommets : les places et les transitions. Dans ce graphe, les arcs doivent relier des sommets de types différents.

Définition 3.3.2 (**Réseau de Petri ordinaire**). Un réseau de Petri $\mathcal{N} = (P, T, F, W, l)$ est dit *ordinaire* ssi pour tout arc $f \in F$, $W(f) = 1$. Un réseau ordinaire est représenté par la structure $\mathcal{N} = (P, T, F, l)$.

Pour chaque transition $t \in T$, nous définissons deux sous-ensembles de places : l'ensemble ${}^\bullet t = \{p : p \in P \text{ et } (p, t) \in F\}$ qui représente les places en *entrée* de la transition t et l'ensemble $t^\bullet = \{p : p \in P \text{ et } (t, p) \in F\}$ qui représente les places en *s*orties de la transitions t .

Les places du réseau $\mathcal{N}$ contiennent des *jetons*. Nous utiliserons une fonction $m : P \to \mathbb{N}$ pour représenter le marquage des places dans le réseau et nous noterons $M = \{m \mid m : P \to \mathbb{N}\}$, l'ensemble de tous les marquages possibles pour le réseau de Petri. Un réseau de Petri est appelé *réseau de Petri initialisé* lorsqu'un marquage initial lui est attribué.

Une transition $t \in T$ est dite *active* (resp. *inactive*) au marquage m si pour toute place $p \in {}^\bullet t$, $W(p, t) \leq m(p)$ (resp. il existe une place $p \in {}^\bullet t$ telle que $W(t, p) > m(p)$). Le marquage m d'un réseau est modifié par le *tir* d'une transition. Pour effectuer le tir, il faut que la transition soit active au marquage m. Le résultat du tir de la transition est l'obtention d'un nouveau marquage.

Exemple 3.3.3. *Considérons le réseau ordinaire $\mathcal{N}$ de la Figure 3.14. Dans cette figure, les places sont représenté par des cercles et les transitions par des rectangles.*

Définition 3.3.4 (**Tir d'une transition**). Lorsqu'une transition $t \in T$ est active pour un marquage m, le *tir* d'une transition est l'action qui consiste à modifier le marquage m en un marquage m' telle que pour toute place $p \in P$, $m'(p) = m(p) - W(p, t) + W(t, p)$. 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.

La notion de tir d'une transition peut être étendu à la notion de *séquence de tir*. Soit $v \in T^*$ tel que $v = t_1 \dot{t}_2 \dots t_n$ nous posons $m[v\rangle m'$ ssi il existe des

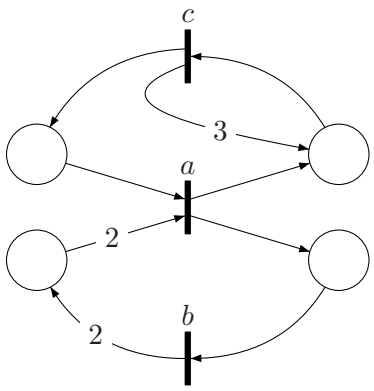

FIG. 3.14 – Résau de Petri $\mathcal{N}$.

marquages $m_1, \ldots m_{n-1}$ tels que $m[t_1\rangle m_1[t_2\rangle \ldots m_{n-1}[t_n\rangle m'$. Un marquage m de $\mathcal{N}$ est *accessible* depuis le marquage initial m_0 lorsqu'il existe une séquence de tir v telle que $m[v\rangle m$. Pour représenter tout les marquages accessibles par un réseau à partir du marquage initial, nous utilisons un automate fini. Dans la littérature [Mur89], cet automate est appelé graphe de marquage.

Définition 3.3.5 (**Exécution des réseaux de Petri**). Soient $\mathcal{N} = (P, T, F, l)$ un réseau de Petri étiqueté sur Σ et m_0 un marquage initial pour $\mathcal{N}$. Nous appelons *exécution* de $\mathcal{N}$, l'automate $Exec(\mathcal{N}, m_0)$=$(Q^{\mathcal{N}}, q_0^{\mathcal{N}}, \rightarrow_{\mathcal{N}})$, sur Σ tel que :

- $Q^{\mathcal{N}} = M$,
- $q_0^{\mathcal{N}} = m_0$ et
- $\rightarrow_{\mathcal{N}} \subseteq Q^{\mathcal{N}} \times \Sigma \cup (\{!, ?\} \times Port) \times Q^{\mathcal{N}}$ est une fonction de transition définie par : $(m, a, m') \in \rightarrow_{\mathcal{N}}$ ssi il existe $t \in T$ tel que $l(t) = a$ et $m[t\rangle m'$

Exemple 3.3.6. *Considérons le réseau de Petri ordinaire et initialisé de la Figure 3.15. Si nous considérons qu'initialement les places p_1 et p_2 sont les seules places contenant des jetons, alors nous obtenons l'exécution du réseau représentée dans la Figure 3.16. Dans cette figure, un état représente, dans l'ordre, les valeurs de $m(p_1)$, $m(p_2)$, $m(p_3)$ et $m(p_4)$, où m représente le marquage courant des places.*

Dans le Chapitre 4, pour établir certains résultats nous avons besoin d'utiliser la notion de places complémentaires.

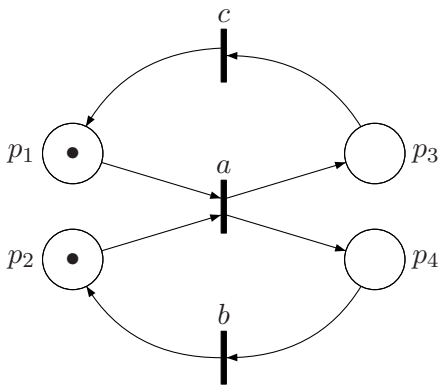

FIG. 3.15 – Résau de Petri ordinaire et initialisé.

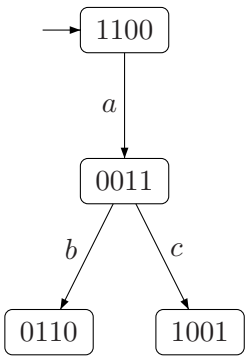

FIG. 3.16 – Exécution du réseau de Petri de la Figure 3.15.

Définition 3.3.7 (**Places complémentaires**). Soient un ensemble fini Σ et un réseau de Petri ordinaire $\mathcal{N} = (P, T, F, l)$ sur Σ. Deux places $p, q \in P$ sont *complémentaires* ssi pour toute transition $t \in T$:

- si $p \in {}^\bullet t$ alors $q \notin {}^\bullet t$, $q \in t^\bullet$ et $p \notin t^\bullet$ et
- si $q \in {}^\bullet t$ alors $p \notin {}^\bullet t$, $p \in t^\bullet$ et $q \notin t^\bullet$.

Remarque 3.3.8. *Lorsque deux places p et q sont complémentaires, si $m_0(p) = 1$ et $m_0(q) = 0$ alors pour tout marquage m accessible depuis m_0, $m(p) + m(q) = 1$.*

3.4 Conclusion

Nous avons rappelé dans ce chapitre des notions de bases sur les automates et les réseaux de Petri. Nous avons également définis la notion d'automate communicant conditionnel qui est une généralisation des automates. Dans les chapitres suivants, afin d'établir des résultats de décidabilité et de complexité, nous aurons besoin d'associer à un ACC un réseau de Petri dont l'exécution est isomorphe à celle de l'ACC (voir Chapitre 4). Réciproquement, nous devrons associer à un réseaux de Petri un ACC dont l'exécution est isomorphe à celle du réseau de Petri (voir Chapitre 5, 6). De plus, Nous avons introduit la notion d'ACC de forme canonique, car il est plus simple de leurs associer des réseaux de Petri.

Chapitre 4

Modèle général pour les services Web

Dans le Chapitre 2, nous avons présenté et comparé les différentes approches existantes pour résoudre le problème de la composition de services. Dans ce chapitre, nous commencerons par la présentation du modèle que nous considérons pour représenter les services et par la description du problème de leur composition [BCF07]. Nous verrons que notre modèle s'inspire de [BCG+05]. Ensuite, nous donnerons la définition formelle des services et du problème de la composition. Nous donnerons des résultats concernant la décidabilité de ce problème. Enfin, nous discuterons les différences entre notre modèle et celui de [BCG+05].

4.1 Modèle des services Web

Les services Web représentent un mécanisme de communication entre applications distantes qui s'effectue à travers le réseau internet. Ce mécanisme est indépendant de tout langage de programmation et de toute plate-forme d'exécution. Cependant, nous considérons dans cette thèse que les communications s'effectuent à travers des ports/canaux de communication et le protocole utilisé pour recevoir les messages est First In First Out (FIFO).

Dans le cadre d'une communication dans un réseau ouvert, comme c'est le cas pour le réseau internet, les fournisseurs de services et leurs clients ne se connaissent pas forcément. De ce fait, il est nécessaire de trouver un moyen pour établir une confiance entre eux. Le fournisseur, par exemple, doit avoir la garantie d'être payé pour le service rendu. Le client quant à lui, doit avoir la garantie que les informations confidentielles qu'il a fournies

au service ne seront pas réutilisées sans sa permission. Parmi les moyens utilisés pour établir la confiance client/fournisseur, il y a l'utilisation de certificats. Les certificats représentent des assertions concernant un service qu'il soit client ou fournisseur. Ces certificats sont émis par une autorité de certification qui les signera en utilisant sa clé privée, ce qui permettra leur authentification par l'utilisation de la clé publique de leur émetteur. Notre modèle est constitué des éléments suivants :

- une communauté de services,
- un service client muni d'un ensemble de certificats,
- un service but et
- un service médiateur qui s'interpose entre le client et les services de la communauté.

4.1.1 Les services

Les services effectuent des opérations de calcul en exécutant des actions, ils obtiennent également des informations au moyen d'échanges de messages avec les autres services. Pour exécuter les actions ou communiquer avec les autres services, des conditions peuvent être exigées. Ces conditions peuvent porter sur la valeur d'une variable interne au service ou sur la valeur d'un attribut d'un des certificats du client. Une fois que l'action est exécutée il est possible que des modifications se produisent sur les variables internes au service ou sur les attributs d'un certificat dans le cas où on considère que les services peuvent être des autorités de certification [CGM06]. Dans la littérature, différents formalismes ont été utilisés pour représenter les services. Parmi ces formalismes nous citons les réseaux de Petri [HB03], les automates d'entrées/sorties [PTB05] et les automates communicants conditionnels [BCG$^+$05]. Pour plus de détail voir le Chapitre 2.

Avec les hypothèses que nous considérons, les réseaux de Petri et les automates communicants conditionnels sont les plus adaptés pour représenter les services. Etant donné que notre travail s'inspire de [BCG$^+$05] et que nous considérons une problématique similaire à la leur, nous avons choisi d'utiliser les automates communicants conditionnels pour représenter les services. Pour les détails concernant les définitions et rappels sur les automates communicants conditionnels, voir le Chapitre 3

Nous présentons maintenant notre modèle formel des services. Dans ce modèle, nous considérons certaines restrictions, pour simplifier l'analyse du problème. Une de ces restrictions consiste à ne pas considérer la structure des messages envoyés par les services ni leur contenu. Il est suffisant de dire qu'un service donné a envoyé/reçu un message dans un port donné, sans préciser de

quel message il s'agit. De plus, nous considérons que les variables locales des services et les attributs des certificats sont des propositions booléennes. De ce fait, nous ne tenons compte d'aucune structure algébrique. Notre modèle est ainsi une abstraction d'un modèle réaliste.

Définition 4.1.1 (Certificat). Un *certificat* est une structure $C = (Attr, Se)$ telle que :
- $Attr$ est un attribut (nom, prénom, date de naissance, etc) et
- Se est le service qui a émis le certificat.

L'ensemble de tous les certificats est noté $\mathcal{C}ert$. Bien entendu, dans la pratique les certificats peuvent avoir plusieurs attributs. Pour simplifier le modèle, nous n'avons considéré qu'un seul attribut [1].

Exemple 4.1.2. *Considérons l'ensemble des certificats $\mathcal{C}ert$ tel que $C_1 = (MasterCr, SGBanque)$ et $C_2 = (VisaCr, CLBanque)$. Ces certificats sont des cartes bancaires. Elles ont comme attribut le type de la carte et comme service émetteur une banque. Dans cet exemple, la première carte est de type Master Card, émise par la banque SGBanque. La seconde carte et de type Visa Card, émise par la banque CLBanque.*

Remarque 4.1.3. Les services qui peuvent émettre les certificats peuvent ne pas appartenir à la communauté de services.

Dans notre représentation formelle des services, nous utilisons un ensemble fini d'actions Σ, un ensemble fini de ports $Port$ et un ensemble fini de formules atomiques At. L'ensemble Σ représente toutes les actions qu'un service peut effectuer, ces actions sont différentes des actions de communication. L'ensemble $Port$ représente tous les ports qu'un service peut utiliser pour communiquer avec un autre service. Les actions de communication seront représentées par l'ensemble $\{!, ?\} \times Port$. L'ensemble At représente les formules atomiques dont les littéraux seront les conditions ou les effets des transitions d'un service. Plus précisément, l'ensemble At contient tous les attributs et les émetteurs des certificats ainsi que toutes les variables booléennes que les services peuvent utiliser.

Exemple 4.1.4. *On s'intéresse à l'achat et à la livraison de livres. L'ensemble des actions $\Sigma = \{$VerifierDispo, Paiement, Annuler, Finaliser$\}$ est tel que :*
- *VerifierDispo : vérifier la disponibilité de l'ouvrage.*

[1] Considérer un nombre fini d'attributs ne change pas nos résultats, car At restera un ensemble fini de formules atomiques.

- *Payment : accepter le paiement.*
- *Annuler : annuler la transaction.*
- *Finaliser : finaliser la transaction.*

L'ensemble des ports $Port = \{$*ChoisirPr, InfoO, EnvoiPrix, PrixO, RepAchat, ConfAchat, Echec, EchecO, Reussit, Descrption, InfoL, PrixLivraison, PrixL, RepLivraison, ConfL, SucsLivraison*$\}$ *tel que les ports sont utilisés pour :*

- *ChoisirPr, InfoO : envoyer/recevoir l'identifiant de l'ouvrage.*
- *EnvoiPrix, PrixO : envoyer/recevoir le prix de l'ouvrage.*
- *RepAchat, ConfAchat : envoyer/recevoir la confirmation de l'achat d'un ouvrage.*
- *Echec, EchecO : envoyer/recevoir un message d'échec de transaction.*
- *Reussit : envoyer/recevoir un message de réussite de transaction.*
- *Descrption, InfoL : envoyer/recevoir le type du produit à livrer, la date et le lieu de sa livraison.*
- *PrixLivraison, PrixL : envoyer/recevoir le prix de livraison d'un produit.*
- *RepLivraison, ConfL : envoyer/recevoir la confirmation d'acceptation du prix.*
- *SucsLivraison : envoyer/recevoir un message de réussite de la transaction.*

On considère les certificats de l'exemple 4.1.2 ainsi que l'ensemble des formules atomiques $At = \{$*Dispo, Acpt, MasterCr, VisaCr, SGBanque, CLBanque, Encaisse, AcptLivraison*$\}$ *tels que :*

- *Dispo : vaut "vrai" si l'ouvrage est disponible.*
- *Acpt : vaut "vrai" si le client accepte de payer l'ouvrage.*
- *MasterCr : vaut "vrai" si le client a une carte de type Master Card.*
- *VisaCr : vaut "vrai" si le client a une carte de type Visa Card.*
- *SGbanque : vaut "vrai" si l'émetteur de la carte est la banque SGBanque.*
- *CLbanque : vaut "vrai" si l'émetteur de la carte est la banque CLBanque.*
- *Encaisse : vaut "vrai" si le paiement de l'ouvrage est accepté.*
- *AcptLivraison : vaut "vrai" si le client accepte de payer la livraison.*

Définition 4.1.5 (Service). Un *service* est un automate communicant conditionnel $\mathcal{A}c = (Q, q_0, \delta)$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At.

Le lecteur remarquera que l'ensemble des états finaux ne figure pas parmi les composantes de l'automate communicant conditionnel. La raison de cette omission est la simplification des preuves de décidabilité et d'indécidabilité,

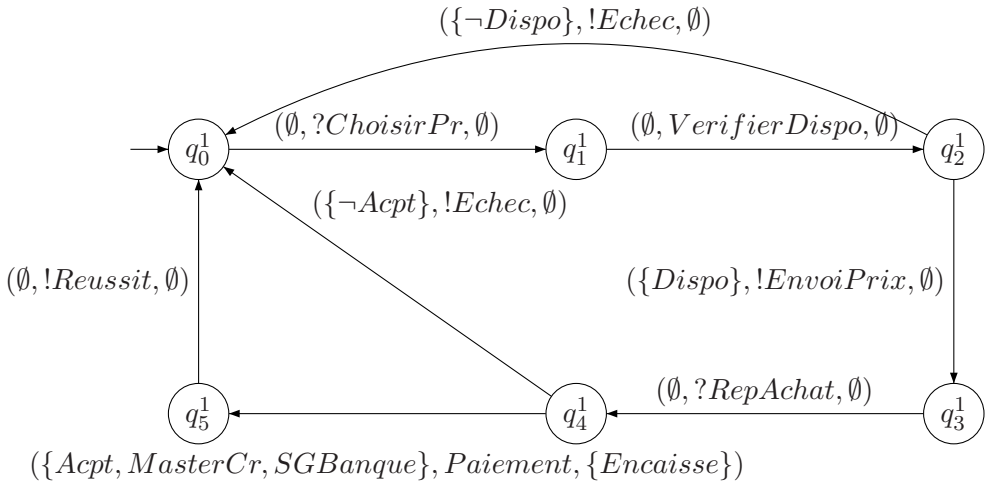

FIG. 4.1 – Service VenteDeLivre

dans les sections suivantes. Cependant, les preuves restent applicables dans le cas général.

Exemple 4.1.6. *Considérons un service qui propose des livres à vendre et qui est intitulé $VenteDeLivre$. Nous représentons ce service par l'automate communicant conditionnel $\mathcal{A}c^1$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At de l'exemple 4.1.4. L'automate communicant conditionnel est représenté dans la Figure 4.1. Dans le service $VenteDeLivre$, pour exécuter certaines transitions des conditions, sont à satisfaire et des effets sont à considérer. Par exemple, pour exécuter l'action $Paiement$, il est nécessaire que le client soit d'accord pour payer, qu'il possède une carte bancaire de type Master Card et que l'émetteur de cette carte soit la banque SGBanque. C'est à dire que les formules atomiques Acpt, MasterCr et SGBanque soient satisfaites. Une fois le paiement accepté, la formule atomique Encaisse devient vraie.*

4.1.2 Communauté de services

Lorsqu'un service est créé, le but est qu'il soit accessible par le maximum d'utilisateurs, qu'ils soient des individus ou des applications. Pour que cela soit possible, il faut que ce service puisse être facilement retrouvé. Le même principe que celui utilisé pour rechercher le numéro de téléphone d'un individu, en utilisant l'annuaire téléphonique, est utilisé pour rechercher l'adresse et la description d'un service. En effet, les services sont répertoriés

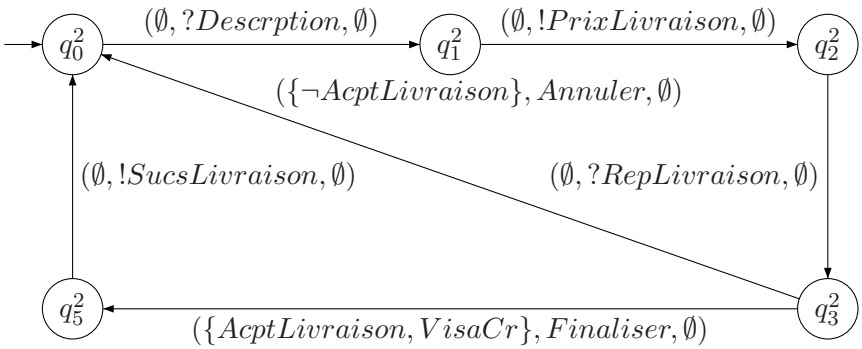

FIG. 4.2 – Service Livraison

dans un référentiel/annuaire appelé UDDI (Universel Description, Discovery, and Integration). Dans le UDDI on ne trouve pas seulement l'adresse du service mais aussi son fichier WSDL (Web Service Description Language). Ce fichier contient la description de l'ensemble des opérations que le service peut effectuer (voir le Chapitre 2).

Dans notre modèle, la *communauté de services* (notée Com) représente l'ensemble des services disponibles référencés dans le UDDI. Ces services sont indépendants les uns des autres. C'est à dire, si un service doit effectuer une action et que les conditions pour effectuer cette action sont satisfaites, alors le service peut l'exécuter sans avoir à se synchroniser avec un autre service. De ce fait, le produit asynchrone est le plus adapté pour représenter le comportement simultané des services.

Définition 4.1.7 (**Communauté de services**). Soit $Com = \{\mathcal{A}c^1, \ldots, \mathcal{A}c^n\}$ une communauté de services représentés par les automates communicants conditionnels finis $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At, nous associons à la communauté Com l'automate communicant conditionnel $\mathcal{A}c^{Com} = \mathcal{A}c^1 \otimes \ldots \otimes \mathcal{A}c^n$.

Exemple 4.1.8. *Considérons une communauté* $Com = \{$*VenteDeLivre, Livraison*$\}$*. Le service VenteDeLivre est celui de l'exemple 5.1.2. Quant au service intitulé Livraison il permet la livraison de différents types de produits à différents lieux. Nous représentons ce service par l'automate communicant conditionnel* $\mathcal{A}c^2$ *sur* $\Sigma \cup (\{!, ?\} \times Port)$ *et* At *de l'exemple 4.1.4. L'automate communicant conditionnel est représenté dans la Figure 4.2.*

4.1.3 Client des services

Les services de la communauté, tels que nous les avons définis, seront utilisés par des humains ou par d'autres services. Dans notre modèle, les clients des services seront définis par trois éléments, le service client, le service but et un ensemble de certificats. Comme dans [BCG+05], dans notre problématique, nous n'avons pas besoin de connaître la description du service qui va utiliser les services disponibles. Il est suffisant de supposer que ce service peut uniquement émettre des requêtes et attendre des réponses en ayant éventuellement des conditions sur la valeur de ses variables internes. Ainsi, il suffira de représenter les utilisateurs des services de la communauté par des services particuliers (ne faisant pas partie de la communauté) qui ne peuvent qu'émettre et/ou recevoir des messages. Ces services seront appelés *services clients*. Soient Σ un ensemble fini d'actions, $Port$ un ensemble fini de port et At un ensemble fini de formules atomiques.

Définition 4.1.9 (**Service client**). Un *service client* est un automate communicant conditionnel $\mathcal{A}c^{client} = (Q^{client}, q_0^{client}, \delta^{client})$ sur $\{!, ?\} \times Port$ et At, tel que $Q^{client} = \{q_0^{client}, q_1^{client}\}$ est une paire d'états et pour tout port $\pi \in Port$ et pour tout ensemble $J, J' \subseteq Li(At)$ on a :
- $(J, q_1^{client}, !\pi, q_i^{client}, J') \notin \delta^{client}$ où $q_i^{client} \in Q^{client}$, $i \in \{0, 1\}$,
- $(J, q_0^{client}, ?\pi, q_i^{client}, J') \notin \delta^{client}$ où $q_i^{client} \in Q^{client}$, $i \in \{0, 1\}$ et

Exemple 4.1.10. *Considérons l'exemple précédent 4.1.8 et un client qui veut acheter un livre et se le faire livrer. Ce client possède les certificats de l'exemple 4.1.2. Comme service client, nous pouvons considérer le service représenté par l'automate communicant conditionnel de la Figure 4.3. Afin de ne pas encombrer la Figure 4.3, nous n'avons gardé qu'une seul transition de q_0^{client} vers q_1^{client} (resp. de q_1^{client} vers q_0^{client}). Par exemple, l'étiquette (∅, !InfoO,∅)/ (∅, !InfoL,∅) / (∅, !ConfAchat,∅)/ (∅, !ConfL,∅) entre q_0^{client} et q_1^{client} représente trois transitions différentes.*

Le service client désire réaliser un certain nombre d'opérations. Afin de réaliser son but, ce dernier souhaiterait interagir avec un service qui pourrait lui effectuer ces opérations. La requête du service client peut ainsi être représentée par un service appelé *service but* qui interagira uniquement avec le service client, qui pourra exécuter des opérations et qui aura des conditions qui portent sur ses variables internes.

Définition 4.1.11 (**Service but**). Un *service but* est un automate communicant conditionnel $\mathcal{A}c^{but} = (Q^{but}, q_0^{but}, \delta^{but})$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At.

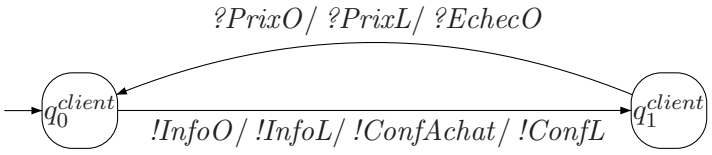

FIG. 4.3 – Service client

Exemple 4.1.12. *Nous considérons le service but représenté dans la Figure 4.4. Ce service but représente les opérations que le client de la Figure 4.3 veut effectuer.*

Le service but est un service fictif qui ne fait pas partie des services de la communauté. Cela dit, il est possible qu'un des services disponibles ait la même spécification que le service but, ce qui sera l'idéal.

Remarque 4.1.13. Comme nous l'avons dit précédemment, les services peuvent exiger que des certificats leur soient présentés avant d'exécuter des opérations. Dans notre modèle, seuls les services clients possèdent un ensemble de certificats. De ce fait, si une condition porte sur la valeur d'un des attributs d'un certificat, il s'agira de celui du service client.

4.1.4 Service médiateur

Le rôle des services médiateurs est de s'interposer entre le service client et les services de la communauté. Ces services sont parfois nécessaires pour établir la communication entre le service client et les services de la communauté. Ce qui est le cas lorsque le service client et les services de la communauté ne communiquent pas sur les mêmes ports/canaux. Dans ce cas, c'est le médiateur qui se charge de faire un pont entre le client et les services disponibles en retransmettant les informations aux uns et aux autres. Les *services médiateurs* effectuent uniquement des échanges de messages. De plus, les conditions des opérations ne concernent que les valeurs de ses variables locales. De plus, le service médiateur ne peut pas modifier la valeur des variables.

Définition 4.1.14 (Service médiateur). Un *service médiateur* est un automate communicant conditionnel $\mathcal{A}c^{med} = (Q^{med}, q_0^{med}, \delta^{med})$ sur $\{!,?\} \times Port$ et At tel que $\delta^{med} \subseteq 2^{Li(At)} \times Q^{med} \times (\{!,?\} \times Port) \times Q^{med} \times 2^{\emptyset}$.

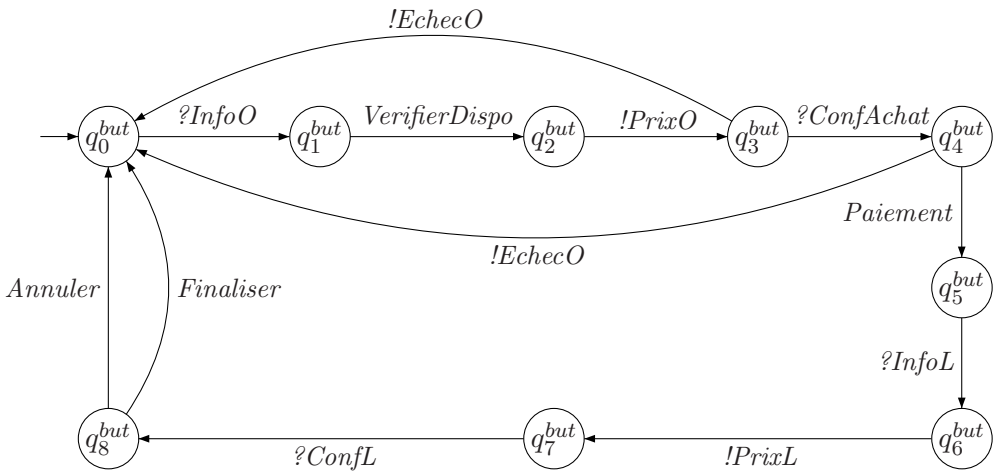

FIG. 4.4 – Service but

Dans le reste du document, nous aurons besoin d'utiliser un service médiateur particulier appelé *service médiateur large*. Ce dernier pourra effectuer toute les communications possibles à travers un ensemble de ports donné. Ce service sera représenté par un automate communicant conditionnel sur un ensemble vide de formules atomiques et contenant un seul état.

Exemple 4.1.15. *Pour l'exemple 4.1.8, nous considérons le service médiateur représenté par l'automate communicant conditionnel de la Figure 4.5. Dans cette figure, les arcs pleins représentent les actions de communication effectuées entre le service client et le service médiateur. Les arcs pointillés représentent les actions de communication effectuées entre le service médiateur et les services de la communauté, ces actions sont invisibles du point de vue du client.*

Définition 4.1.16 (**Service médiateur large**)**.** Un *service médiateur large* pour $Port$ est un automate communicant conditionnel $\mathcal{A}c^{L_{Port}} = (Q^{L_{Port}}, q_0^{L_{Port}}, \delta^{L_{Port}})$ sur $\{!,?\} \times Port$ et At tel que :

- $Q^{L_{Port}} = \{q_0^{L_{Port}}\}$ et
- $\delta^{L_{Port}} = \{(\emptyset, q_0^{L_{Port}}, a, q_0^{L_{Port}}, \emptyset) \mid a \in \{!,?\} \times Port\}$.

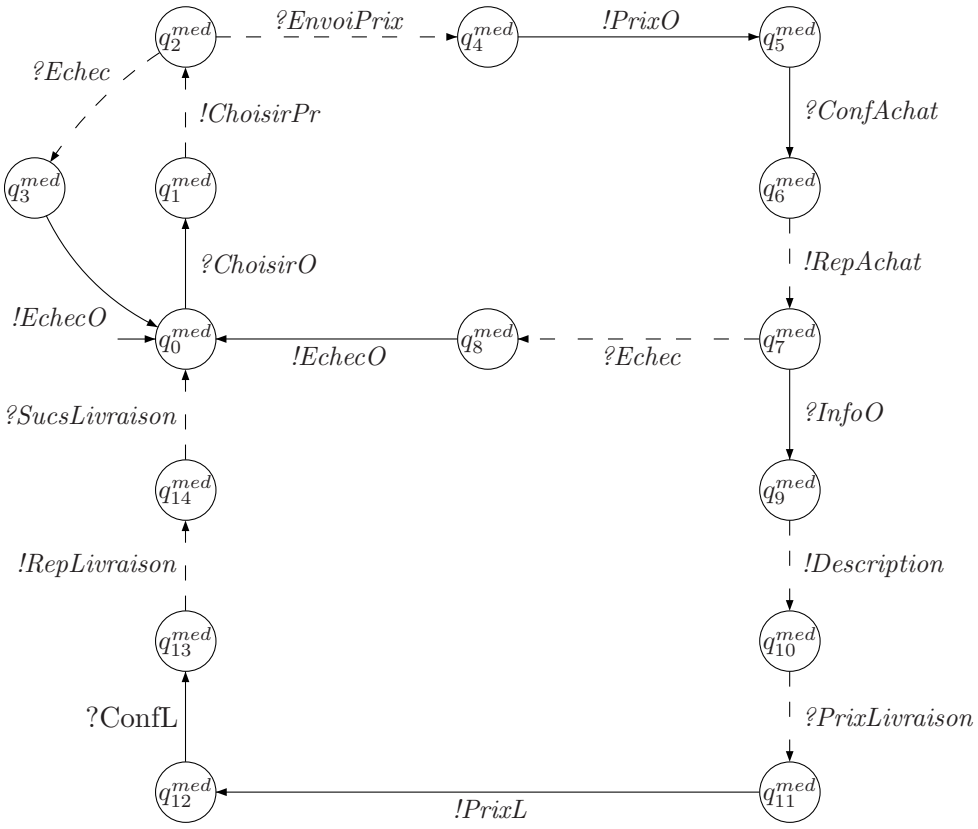

FIG. 4.5 – Service médiateur

Exemple 4.1.17. *Pour l'ensemble Port de l'exemple 4.1.4, nous représentons le service médiateur large dans la Figure 4.6.*

$$!ChoisirPr, ?ChoisirPr, \ldots, !SucsLivraison, ?SucsLivraison$$

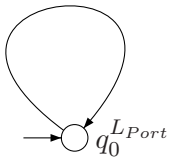

FIG. 4.6 – Service médiateur large.

4.1.5 Problème de la composition de services

Lorsqu'un client veut effectuer certaines opérations et qu'aucun des services de la communauté ne peut réaliser seul ces opérations, 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. Parfois, un service médiateur est nécessaire pour permettre ce lien. Certains travaux [PTB05], comme nous l'avons précisé dans le Chapitre 2, s'intéressent à lier les services afin d'obtenir un nouveau service qui doit vérifier certaines propriétés comme"ne pas atteindre un état bloquant". D'autre travaux comme ceux de [BCG$^+$05], s'intéressent à lier les services afin d'obtenir un nouveau service qui aurait un comportement similaire ou équivalent à celui d'un service prédéfini. Notre travail s'inscrit dans la seconde catégorie. Plus précisement, notre problème de la composition est le problème de décision défini de la façon suivante :

Problème $\mathcal{PC}(\approx)$ **:** problème de la composition

Instance : un ensemble fini d'action Σ, des ensembles finis de ports $Port$ et $Port' \subseteq Port$, un ensemble fini de formules atomiques At, un ensemble maximal consistant de littéraux I_0 sur At un automate communicant conditionnel $\mathcal{A}c^{client}$ sur $\{!, ?\} \times Port$ et At et des automates communicants conditionnels finis $\mathcal{A}c^{but}$, $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At.

Question : existe-t-il un automate communicant conditionnel $\mathcal{A}c^{med}$ sur $\{!, ?\} \times Port$ et At, tel que

$$Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, I_0)$$
$$\approx$$
$$Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{1} \circledast \ldots \circledast \mathcal{A}c^{n} \circledast \mathcal{A}c^{med}, I_0)(\{!,?\} \times Port')\,?$$

Dans cette définition, I_0 représente la valeur initiale des formules atomiques dans At et $Exec(\mathcal{A}c, I_0)$ l'exécution ou le comportement d'un automate communicant conditionnel $\mathcal{A}c$. Pour plus de détail, veuillez consulter le Chapitre 3.

Intuitivement, le problème de la composition est le suivant : étant donnés un service client, un service but pour ce client et des services disponibles, déterminer s'il existe un service médiateur qui établira une liaison entre le service client et les services disponibles. La liaison doit être de sorte que le comportement du système composé des services disponibles communicants avec le service client, à travers le service médiateur, soit équivalent au comportement du système composé du service client communicant avec le service but. Ces deux systèmes doivent être équivalents du point de vue du client. En d'autres termes, nous voulons que les deux systèmes soient équivalents lorsqu'on ne considère pas les communications faites sur des ports qui ne concernent pas le client. Ces ports seront représentés par l'ensemble $Port' \subseteq Port$. Ainsi, nous allons considérer des équivalences modulo $\{!,?\} \times Port'$, en ayant les communications entre les services de la communauté ou entre le médiateur et ces services comme actions non observables.

Le problème $\mathcal{PC}$ représente quatre problèmes différents suivant que $\approx$ est égale à $\subseteq_{tr}$, $\equiv_{tr}$, $\leq_{si}$ ou $\longleftrightarrow_{bi}$ [2].

Intuitivement, vouloir que le comportement du système composé du service but et du service client soit inclus dans celui du système composé du service client, des services de la communauté et du service médiateur, signifie que toute séquence d'actions effectuée par le premier système peut également être effectuée par le second (en permettant éventuellement aux second système d'effectuer des actions supplémentaires).

Dans le cas où l'équivalence de trace est considérée, le second système ne peut pas effectuer de séquence d'actions que le premier système ne peut pas effectuer. Par exemple, cela est utile pour garantir la sécurité du système en interdisant des séquences d'actions potentiellement dangereuses. En effet, certaines séquences d'actions peuvent permêtre la déduction d'informations confidentielles.

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

[2]Toutes les définitions concernant les équivalences et préordres se trouvent dans le Chapitre 3

systèmes à comparer. Lorsque les systèmes sont non déterministes, la bisimulation (resp. simulation) est une relation plus restrictive que l'équivalence de traces (resp. l'inclusion de traces). En effet, lorsque deux systèmes sont bisimilaires, tous les états accessibles dans le premier système sont bisimilaires à des états accessibles dans le second système. Par conséquent, toutes les actions effectuées à partir des états accessibles, dans le premier système, peuvent être effectuées à partir des états correspondant, dans le second système. La réciproque est également vraie, la relation de bisimulation étant symétrique. Concrètement, cela signifie que si un des deux systèmes ne contient pas d'état bloquant accessible à partir de l'état initial, alors l'autre système vérifie également cette propriété.

Exemple 4.1.18. *Considérons les services VenteDeLivre, Livraison, Client et But représentés respectivement dans les Figures 4.1, 4.2, 4.3 et 4.4. Le service médiateur représenté dans la Figure 4.5 permet de répondre "oui" au problème de décision $\mathcal{PC}(\longleftrightarrow_{bi})$. Dans la Figure 4.7 sont représentés tous les services avec leurs différentes interactions* [3]*. Les arc pointillés sont ceux que le service client ne peut pas voir.*

4.2 Résultats

Dans cette section nous donnons des résultats concernant la décidabilité du problème de la composition de services. Précédemment, nous avons défini formellement quatre problèmes, $\mathcal{PC}(\subseteq_{tr})$, $\mathcal{PC}(\equiv_{tr})$, $\mathcal{PC}(\leq_{si})$ et $\mathcal{PC}(\longleftrightarrow_{bi})$. Dans notre modèle, nous considérons une communication qui s'effectue à travers des ports. Dans ce qui suit, nous montrons que dans le cas où les ports sont non bornés, les problèmes $\mathcal{PC}(\subseteq_{tr})$ et $\mathcal{PC}(\leq_{si})$ sont décidables alors que les problèmes $\mathcal{PC}(\equiv_{tr})$ et $\mathcal{PC}(\longleftrightarrow_{bi})$ ne le sont pas. Pour prouver la décidabilité de $\mathcal{PC}(\subseteq_{tr})$ et $\mathcal{PC}(\leq_{si})$ nous utiliserons des résultats connus concernant la comparaison d'un réseau de Petri étiqueté avec un automate fini. Concernant l'indécidabilité de $\mathcal{PC}(\equiv_{tr})$ et $\mathcal{PC}(\longleftrightarrow_{bi})$ nous utiliserons des résultats connus à propos de la comparaison de deux réseaux de Petri étiquetés. Avant d'énoncer les théorèmes concernant les résultats de décidabilité ou d'indécidabilité de ces quatre problèmes, nous donnons des résultats préliminaires sous la forme de propositions.

Proposition 4.2.1. *Soient Σ un ensemble fini d'actions, Port un ensemble fini de ports et $\mathcal{A}$ un automate fini sur $\Sigma \cup (\{!, ?\} \times Port)$.*

[3] Cette figure représente une vue d'un observateur extérieur au modèle.

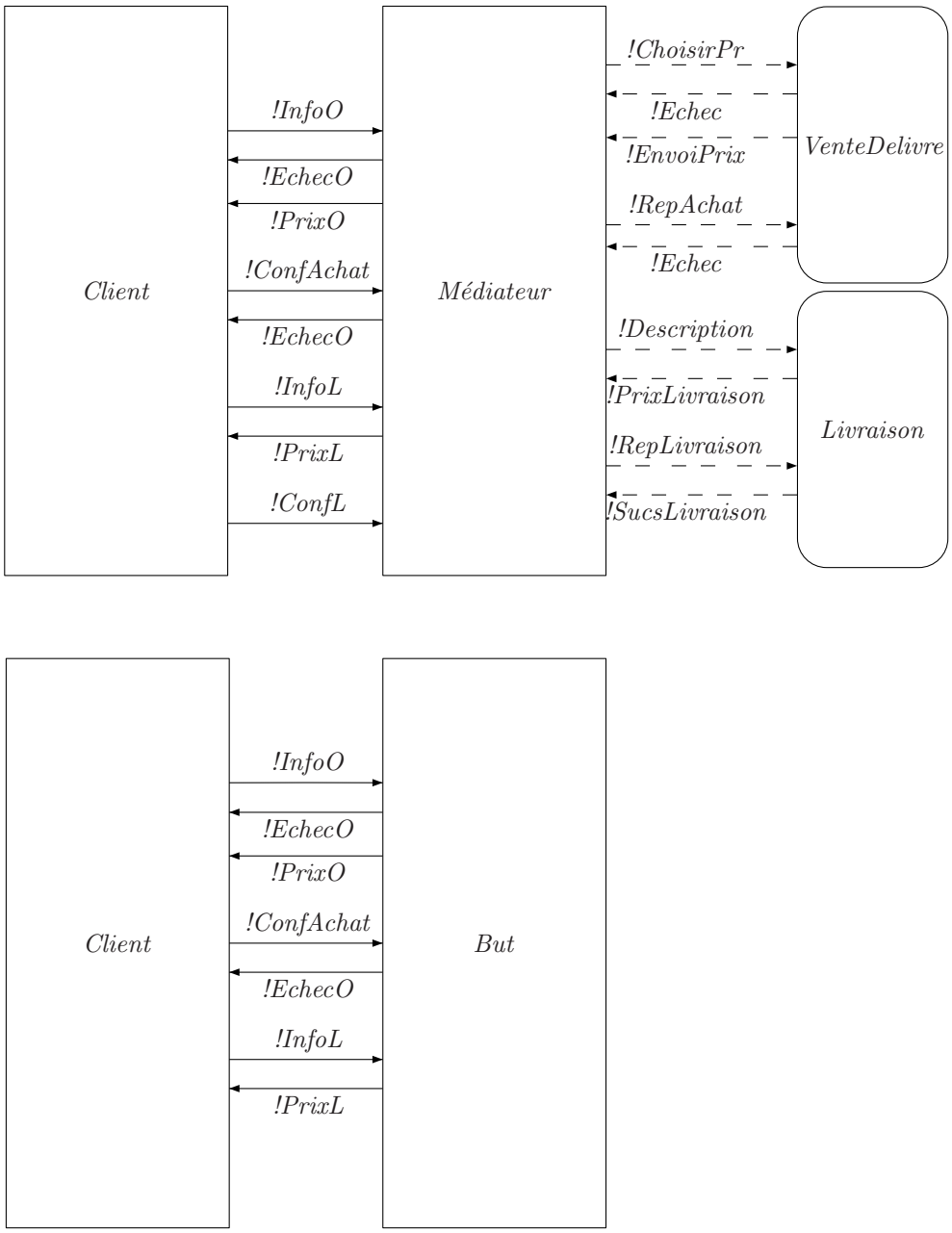

FIG. 4.7 – Le modèle vu par un observateur extérieur

$$Exec(\mathcal{A}) \text{ et } Exec(Auto^{-1}(\mathcal{A}), \emptyset) \text{ sont isomorphes.}$$

Dans la remarque 3.2.7, nous avons associé un automate communicant conditionnel $Auto^{-1}(\mathcal{A})$, à un automate $\mathcal{A}$. Dans la proposition ci-dessus, on ajoute le fait que l'exécution de l'automate est isomorphe à celle de l'automate communicant conditionnel.

Démonstration. Considérons un ensemble fini d'actions Σ, un ensemble fini de ports $Port$ et un automate fini $\mathcal{A} = (Q, q, \rightarrow)$ sur $\Sigma \cup (\{!, ?\} \times Port)$. Considérons l'ensemble Γ de toutes les fonctions $\gamma : Port \rightarrow \mathbb{N}$ et la bijection $g : Q \times \Gamma \rightarrow Q \times \Gamma \times 2^{\emptyset}$ telle que $g(q, \gamma) = (q, \gamma, \emptyset)$. Il est clair que pour tout état (q, γ), $(q', \gamma') \in Q \times \Gamma$ et pour toute action $a \in \{!, ?\} \times Port$ on a :

$$((q, \gamma), a, (q', \gamma')) \in \rightarrow_{Exec(\mathcal{A})} \text{ ssi } (g(q, \gamma), a, g(q', \gamma')) \in \rightarrow_{Exec(Auto^{-1}(\mathcal{A}),\emptyset)}.$$

□

Proposition 4.2.2. *Soient Σ un ensemble fini d'actions, Port un ensemble fini de ports et $\mathcal{A}^1$ et $\mathcal{A}^2$ des automates finis sur $\Sigma \cup (\{!, ?\} \times Port)$.*

$$Auto^{-1}(\mathcal{A}^1 \otimes \mathcal{A}^2) = Auto^{-1}(\mathcal{A}^1) \circledast Auto^{-1}(\mathcal{A}^2).$$

Ci-dessous, nous montrons que l'automate communicant conditionnel associé au produit asynchrone de deux automates, n'est autre que le produit asynchrone des automates communicants conditionnels associés à ces deux automates.

Démonstration. Considérons un ensemble fini d'actions Σ, un ensemble fini de ports $Port$ et des automates finis $\mathcal{A}^1 = (Q^1, q_0^1, \rightarrow_{\mathcal{A}^1})$ et $\mathcal{A}^2 = (Q^2, q_0^2, \rightarrow_{\mathcal{A}^2})$ sur $\Sigma \cup (\{!, ?\} \times Port)$. Egalement, considérons $Auto^{-1}(\mathcal{A}^1 \otimes \mathcal{A}^2) = (Q, q_0, \delta)$ et $Auto^{-1}(\mathcal{A}^1) \circledast Auto^{-1}(\mathcal{A}^2) = (Q', q_0', \delta')$. D'après la remarque 3.2.7 et la définition du produit asynchrone entre automates 3.3 et entre automates communicants conditionnels 3.2.8, on a :

$$Q = Q' = Q^1 \times Q^2 \text{ et } q_0 = q_0' = (q_0^1, q_0^2).$$

Ainsi, pour que $Auto^{-1}(\mathcal{A}^1 \otimes \mathcal{A}^2) = Auto^{-1}(\mathcal{A}^1) \circledast Auto^{-1}(\mathcal{A}^2)$, il suffit de monter que $\delta = \delta'$. Plus précisément, il faut prouver que pour tout état $q_1^1, q_2^1 \in Q^1$ et $q_1^2, q_2^2 \in Q^2$ et toute action $a \in \Sigma \cup (\{!, ?\} \times Port)$ on a :

$$(\emptyset, (q_1^1, q_1^2), a, (q_2^1, q_2^2), \emptyset) \in \delta \text{ ssi } (\emptyset, (q_1^1, q_1^2), a, (q_2^1, q_2^2), \emptyset) \in \delta'.$$

D'après la remarque 3.2.7 on a :

$$(\emptyset, (q_1^1, q_1^2), a, (q_2^1, q_2^2), \emptyset) \in \delta \text{ ssi } ((q_1^1, q_1^2), a, (q_2^1, q_2^2)) \in \rightarrow_{\mathcal{A}^1 \otimes \mathcal{A}^2}.$$

D'après la définition 3.3 on a :

$$((q_1^1, q_1^2), a, (q_2^1, q_2^2)) \in \rightarrow_{\mathcal{A}^1 \otimes \mathcal{A}^2} \text{ ssi } (q_1^1, a, q_2^1) \in \rightarrow_{\mathcal{A}^1} \text{ ou } (q_1^2, a, q_2^2) \in \rightarrow_{\mathcal{A}^2}.$$

D'après la remarque 3.2.7 on a :

$$(q_1^1, a, q_2^1) \in \rightarrow_{\mathcal{A}^1} \text{ ou } (q_1^2, a, q_2^2) \in \rightarrow_{\mathcal{A}^2} \text{ ssi } (\emptyset, q_1^1, a, q_2^1, \emptyset) \in \delta^{Auto^{-1}(\mathcal{A}^1)} \text{ et } q_1^2 = q_2^2 \text{ ou } (\emptyset, q_1^2, a, q_2^2, \emptyset) \in \delta^{Auto^{-1}(\mathcal{A}^2)} \text{ et } q_1^1 = q_2^1.$$

D'après la définition 3.2.8 on a :

$$(\emptyset, q_1^1, a, q_2^1, \emptyset) \in \delta^{Auto^{-1}(\mathcal{A}^1)} \text{ et } q_1^2 = q_2^2 \text{ ou } (\emptyset, q_1^2, a, q_2^2, \emptyset) \in \delta^{Auto^{-1}(\mathcal{A}^2)} \text{ ssi } (\emptyset, (q_1^1, q_1^2), a, (q_2^1, q_2^2), \emptyset) \in \delta' \text{ et } q_1^1 = q_2^1.$$

On n'en déduit que

$$(\emptyset, (q_1^1, q_1^2), a, (q_2^1, q_2^2), \emptyset) \in \delta \text{ ssi } (\emptyset, (q_1^1, q_1^2), a, (q_2^1, q_2^2), \emptyset) \in \delta'.$$

Par conséquent, $Auto^{-1}(\mathcal{A}^1 \otimes \mathcal{A}^2) = Auto^{-1}(\mathcal{A}^1) \circledast Auto^{-1}(\mathcal{A}^2)$.

□

4.2.1 Résultats de décidabilité

Afin de montrer que les problèmes de la composition $\mathcal{PC}(\subseteq_{tr})$ et $\mathcal{PC}(\leq_{si})$ sont décidables, nous allons réduire ces problèmes à des problèmes plus abstraits, qui seront à leur tour réduits à des problèmes connus comme étant décidables. Plus précisément, dans un premier temps, nous allons considérer les problèmes de décisions $\mathcal{PEL}(\subseteq_{tr})$ et $\mathcal{PEL}(\leq_{si})$. Ensuite, nous allons réduire le problème $\mathcal{PEL}(\subseteq_{tr})$ (resp. $\mathcal{PEL}(\leq_{si})$) au problème de l'inclusion de trace (resp. simulation) entre un réseau de Petri étiqueté et un automate fini. Ce dernier est un problème décidable [Jan94]. Le problème $\mathcal{PEL}(\approx)$, avec $\approx \in \{\subseteq_{tr}, \equiv_{tr}, \leq_{si}, \longleftrightarrow_{bi}\}$ est un problème qui permet, étant donnés deux automates communicants conditionnels finis $\mathcal{A}c$ et $\mathcal{A}c'$ définis sur un ensemble fini de ports $Port$, de décider si l'exécution de $\mathcal{A}c$ est équivalente à celle du système composé de $\mathcal{A}c'$ et du médiateur large $\mathcal{A}c^{L_{Port}}$. Soulignons que dans ce problème nous ne cherchons pas à savoir s'il est possible de synthétiser le médiateur, pour obtenir l'équivalence, mais a vérifier si avec le médiateur large l'équivalence est vérifiée. Ci-dessous, nous décrivons formellement le problème $\mathcal{PEL}(\approx)$, avec $\approx \in \{\subseteq_{tr}, \equiv_{tr}, \leq_{si}, \longleftrightarrow_{bi}\}$.

Problème $\mathcal{PEL}(\approx)$: problème d'équivalence avec le médiateur large

Instance : un ensemble fini d'actions Σ, des ensembles finis de ports $Port$ et $Port' \subseteq Port$, un ensemble fini de formules atomiques At, un ensemble maximal consistant de littéraux I_0 sur At et des automates

communicants conditionnels finis $\mathcal{A}c$ et $\mathcal{A}c'$ sur $\Sigma \cup (\{!,?\} \times Port)$, et At.

Question : est-ce-que

$$Exec(\mathcal{A}c, I_0) \approx Exec(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port}}, I_0)(\{!,?\} \times Port'), \text{ où } \approx \in \{\subseteq_{tr}, \equiv_{tr}, \leq_{si}, \longleftrightarrow_{bi}\} ?$$

Afin de réduire le problème de la composition $\mathcal{PC}(\sqsubseteq)$, avec $\sqsubseteq \in \{\leq_{si}, \subseteq_{tr}\}$ au problème de l'équivalence avec le médiateur large $\mathcal{PEL}(\sqsubseteq)$, avec $\sqsubseteq \in \{\leq_{si}, \subseteq_{tr}\}$, nous avons besoin du lemme suivant.

Lemme 4.2.3. *Soient Σ un ensemble fini d'actions, $Port$ un ensemble fini de ports, At un ensemble fini de formules atomiques, I_0 un ensemble maximal consistant de littéraux sur At, $\mathcal{A}c$ un automate communicant conditionnel fini sur $\Sigma \cup (\{!,?\} \times Port)$ et At et $\mathcal{A}c^{med}$ un automate communicant conditionnel fini sur $\{!,?\} \times Port$ et At.*

$$Exec(\mathcal{A}c \circledast \mathcal{A}c^{med}, I_0) \leq_{si} Exec(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port}}, I_0)(\emptyset).$$

Démonstration. Considérons un ensemble fini d'actions Σ, un ensemble fini de ports $Port$, un ensemble fini de formules atomiques At, un ensemble maximal consistant de littéraux I_0 sur At, les automates communicants conditionnels finis $\mathcal{A}c = (Q, q_0, \delta)$ et $\mathcal{A}c^{med} = (Q^{med}, q_0^{med}, \delta^{med})$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At et le médiateur large $Ac^{L_{Port}}$=$(Q^{L_{Port}}, q_0^{L_{Port}}, \delta^{L_{Port}})$ où $Q^{L_{Port}} = \{q_0^{L_{Port}}\}$. Rappelons que l'ensemble de toutes les fonctions de la forme $\gamma : Port \to \mathbb{N}$ sera représenté par Γ et que l'ensemble des littéraux de At sera représenté par $Li(At)$.

Pour rendre la preuve plus claire, nous utilisons l'automate $\mathcal{A} = (Q^{\mathcal{A}}, q_0^{\mathcal{A}}, \rightarrow_{\mathcal{A}})$ sur $\Sigma \cup (\{!,?\} \times Port)$ pour représenter l'automate $Exec(\mathcal{A}c \circledast \mathcal{A}c^{med}, I_0)$. De la même façon, nous utilisons l'automate $\mathcal{B} = (Q^{\mathcal{B}}, q_0^{\mathcal{B}}, \rightarrow_{\mathcal{B}})$ sur $\Sigma \cup (\{!,?\} \times Port)$ pour représenter l'automate $Exec(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port}}, I_0)$. Ainsi, $Q^{\mathcal{A}} = (Q \times Q^{med}) \times \Gamma \times 2^{Li(At)}$ et $Q^{\mathcal{B}} = (Q \times Q^{L_{Port}}) \times \Gamma \times 2^{Li(At)}$.

Soit la relation $Z \subseteq Q^{\mathcal{A}} \times Q^{\mathcal{B}}$ telle que $((q, q^{med}), \gamma, I)\ Z\ ((q, q_0^{L_{Port}}), \gamma, I)$. Nous devons prouver que Z est une simulation telle que $q_0^{\mathcal{A}}\ Z\ q_0^{\mathcal{B}}$.

Concernant les états initiaux, l'état initial de $\mathcal{A}$ est $((q_0, q_0^{med}), \gamma_0, I_0)$ où $\gamma_0(\pi) = 0$, pour tout $\pi \in Port$. L'état initial de $\mathcal{B}$ est $((q_0, q_0^{L_{Port}}), \gamma_0, I_0)$. Il est clair que $((q_0, q_0^{med}), \gamma_0, I_0)\ Z\ ((q_0, q_0^{L_{Port}}), \gamma_0, I_0)$.

Soient les états $((q_1, q_1^{med}), \gamma_1, I_1) \in Q^{\mathcal{A}}$ et $((q_1', q_0^{L_{Port}}), \gamma_1', I_1') \in Q^{\mathcal{B}}$ tels que $((q_1, q_1^{med}), \gamma_1, I_1)\ Z\ ((q_1', q_0^{L_{Port}}), \gamma_1', I_1')$. Supposons qu'il existe un état $((q_2, q_2^{med}), \gamma_2, I_2) \in Q^{\mathcal{A}}$ et une action $a \in \Sigma \cup (\{!,?\} \times Port)$ tels que $((q_1, q_1^{med}), \gamma_1, I_1) \rightarrow_{\mathcal{A}}^{\{a\}} ((q_2, q_2^{med}), \gamma_2, I_2)$. Deux cas sont possibles pour a ; $a \in \Sigma$ ou $a \in \{!,?\} \times Port$.

Considérons le cas $a \in \Sigma$, $((q_1,\ q_1^{med}),\ \gamma_1,\ I_1) \rightarrow_{\mathcal{A}}^{\{a\}} ((q_2,\ q_2^{med}),\ \gamma_2,\ I_2)$ implique qu'il existe $J, J' \subseteq Li(At)$ tel que $(J, q_1, a, q_2, J') \in \delta$, $J \subseteq I_1$, $I_2 = (I \setminus \neg J') \cup J'$, que $q_2^{med} = q_1^{med}$ et que $\gamma_2 = \gamma_1$. Sachant que $((q_1, q_1^{med}), \gamma_1, I_1)$ Z $((q_1', q_0^{L_{Port}}), \gamma_1', I_1')$ on a $q_1' = q_1$, $\gamma_1' = \gamma_1$ et $I_1' = I_1$. Ainsi, pour $q_2' = q_2$, $\gamma_2' = \gamma_2$ et $I_2' = I_2$, il est clair que $((q_1',\ q_0^{L_{Port}}),\ \gamma_1',\ I_1') \rightarrow_{\mathcal{B}}^{a} ((q_2',\ q_0^{L_{Port}}),\ \gamma_2',\ I_2')$ et que $((q_2,\ q_1^{med}),\ \gamma_2,\ I_2)$ Z $((q_2',\ q_0^{L_{Port}}),\ \gamma_2',\ I_2')$.

Considérons le cas $a = ?\pi$, $\pi \in Port$ (le raisonnement est le même pour $a = !\pi$), l'action $?\pi$ peut être exécutée par $\mathcal{A}c$ ou par $\mathcal{A}c^{med}$. Dans le cas où l'action est effectuée par $\mathcal{A}c$, on utilise le même raisonnement que celui utilisé dans le cas $a \in \Sigma$. Supposons que $\mathcal{A}c^{med}$ exécute $?\pi$. Donc il existe un ensemble $J \subseteq Li(At)$ tel que $(J, q_1^{med}, ?\pi, q_2^{med}, \emptyset) \in \delta^{med}$, $J \subseteq I_1$, $I_2 = I_1$, $\gamma_2(\pi) = \gamma_1(\pi) - 1$ et pour tout $\pi' \neq \pi$, $\gamma_2(\pi') = \gamma_1(\pi')$. De plus, $q_2 = q_1$. Sachant que $((q_1, q_1^{med}), \gamma_1, I_1)$ Z $((q_1', q_0^{L_{Port}}), \gamma_1', I_1')$ on a $q_1' = q_1$, $\gamma_1' = \gamma_1$ et $I_1' = I_1$. Nous savons que $(\emptyset,\ q_0^{L_{Port}},\ ?\pi,\ q_0^{L_{Port}},\ \emptyset) \in \delta^{L_{Port}}$. Ainsi, pour $q_2' = q_2$, $\gamma_2' = \gamma_2$ et $I_2' = I_2$, il est clair que $((q_1',\ q_0^{L_{Port}}),\ \gamma_1',\ I_1') \rightarrow_{\mathcal{B}}^{?\pi} ((q_2',\ q_0^{L_{Port}}),\ \gamma_2',\ I_2')$ et que $((q_2,\ q_1^{med}),\ \gamma_2,\ I_2)$ Z $((q_2',\ q_0^{L_{Port}}),\ \gamma_2',\ I_2')$.

Par conséquent Z est une simulation. Ainsi $Exec(\mathcal{A}c \circledast \mathcal{A}c^{med}, I_0) \leq_{si} Exec(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port}}, I_0)(\emptyset)$. □

Soulignons que ce lemme reste correct dans le cas où la simulation modulo un ensemble non vide ou l'inclusion modulo un ensemble non vide sont considérées. Cela est dû au fait que la simulation implique ces deux autres préordres. Maintenant, nous montrons que, pour la simulation et l'inclusion de traces, le problème de l'équivalence avec le médiateur large et le problème de la composition sont équivalents. Dans un premier temps, nous prouvons le lemme concernant la réduction du problème de l'équivalence avec le médiateur large au problème de la composition[4]. Dans un deuxième temps, nous prouvons le lemme concernant la réduction du problème de la composition au problème de l'équivalence avec le médiateur large.

Lemme 4.2.4. *Le problème $\mathcal{PEL}(\sqsubseteq)$, $\sqsubseteq \in \{\leq_{si}, \subseteq_{tr}\}$ se réduit au problème $\mathcal{PC}(\sqsubseteq)$, en utilisant un espace logarithmique.*

Démonstration. Considérons une instance du problème $\mathcal{PEL}(\sqsubseteq)$. Soient Σ un ensemble fini d'actions, $Port$ et $Port' \subseteq Port$ des ensembles finis de ports, At un ensemble fini de formules atomiques, I_0 un ensemble maximal consistant de littéraux sur At, des automates communicants conditionnels finis $\mathcal{A}c$ et $\mathcal{A}c'$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At. L'instance $\rho(\Sigma,\ Port,\ Port',\ At,\ I_0,\ \mathcal{A}c,\ \mathcal{A}c')$ de $\mathcal{PC}(\sqsubseteq)$ que nous construisons est donnée par les ensembles

[4]Ce résultat sera utilisé dans le Chapitre 6, plus précisément dans la Section 6.3.1.

$\Sigma_e = \Sigma$, $Port_e = Port$, $Port'_e = Port'$, $At_e = At$ et $I_0^e = I_0$, le service client $\mathcal{A}c^{client}$ tel que $Q^{client} = \emptyset$, le service but $\mathcal{A}c^{but} = \mathcal{A}c$ et un unique service disponible $\mathcal{A}c^1 = \mathcal{A}c'$. Deux point sont à vérifier :

(1) ρ est calculable par une machine de Turing déterministe en utilisant un espace logarithmique,

(2) $Exec(\mathcal{A}c, I_0) \sqsubseteq Exec(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port}}, I_0)(\{!, ?\} \times Port')$ ssi il existe un automate communicant conditionnel $\mathcal{A}c^{med}$ sur $\{!, ?\} \times Port$ et At, tel que $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c, I_0) \sqsubseteq Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0)(\{!, ?\} \times Port')$.

Concernant le point (1), la machine de Turing qui calcule ρ va d'abord écrire sur son ruban de sortie les ensemble Σ, $Port$, $Port'$, At et I_0, qu'elle lira sur son ruban d'entrée. Ensuite, elle écrira $\mathcal{A}c^{client}$, $\mathcal{A}c^{but}$ et $\mathcal{A}c^1$. La machine n'aura pas besoin d'utiliser un ruban de travail. Par conséquent ρ est calculable par une machine de Turing déterministe en utilisant un espace logarithmique.

Concernant le point (2), sachant que $Q^{client} = \emptyset$, il est clair que : $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c, I_0)$ est isomorphe à $Exec(\mathcal{A}c, I_0)$. De la même façon, pour tout automate communicant conditionnel Ac^{med}, il est clair que : $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0)$ est isomorphe à $Exec(\mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0)$. Il nous suffira donc de prouver que :

$$Exec(\mathcal{A}c, I_0) \sqsubseteq Exec(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port}}, I_0)(\{!, ?\} \times Port')$$
$$\Longleftrightarrow$$
il existe un automate communicant conditionnel $\mathcal{A}c^{med}$ sur $\{!, ?\} \times Port$ et At, tel que $Exec(\mathcal{A}c, I_0) \sqsubseteq Exec(\mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0)(\{!, ?\} \times Port')$.

($\Rightarrow$) Concernant l'implication de gauche à droite, elle est triviale. En effet, supposons que :

$$Exec(\mathcal{A}c, I_0) \sqsubseteq Exec(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port}}, I_0)(\{!, ?\} \times Port').$$

Dans ce cas, il suffit de considérer $\mathcal{A}c^{med} = \mathcal{A}c^{L_{Port}}$ pour avoir :

$$Exec(\mathcal{A}c, I_0) \sqsubseteq Exec(\mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0)(\{!, ?\} \times Port').$$

($\Leftarrow$) Considérons l'implication de droite à gauche. Supposons qu'il existe un automate communicant conditionnel $\mathcal{A}c^{med}$ sur $\{!, ?\} \times Port$ et At tel que :

$$Exec(\mathcal{A}c, I_0) \sqsubseteq Exec(\mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0)(\{!, ?\} \times Port'). \quad (4.1)$$

D'après le lemme 4.2.3, on a :

$$Exec(\mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0) \leq_{si} Exec(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port}}, I_0)(\emptyset). \quad (4.2)$$

A partir de 4.1 et 4.2 on a :

$$Exec(\mathcal{A}c, I_0) \sqsubseteq Exec(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port}}, I_0)(\{!,?\} \times Port').$$

Ceci conclut la preuve. □

Lemme 4.2.5. *Le problème $\mathcal{PC}(\sqsubseteq)$, $\sqsubseteq \in \{\leq_{si}, \subseteq_{tr}\}$ se réduit au problème $\mathcal{PEL}(\sqsubseteq)$.*

Démonstration. Considérons une instance du problème $\mathcal{PC}(\sqsubseteq)$. Soient Σ un ensemble fini d'actions, $Port$ et $Port' \subseteq Port$ des ensembles finis de ports, At un ensemble fini de formules atomiques, I_0 un ensemble maximal consistant de littéraux sur At, un automate communicant conditionnel fini $\mathcal{A}c^{client}$ sur $\{!,?\} \times Port$ et At et des automates communicants conditionnels finis $\mathcal{A}c^{but}$, $\mathcal{A}c^1$, ..., $\mathcal{A}c^n$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At. L'instance $\rho(\Sigma,\ Port,\ Port',\ At,\ I_0,\ \mathcal{A}c^{client},\ \mathcal{A}c^{but},\ \mathcal{A}c^1, \ldots\ \mathcal{A}c^n)$ de $\mathcal{PEL}(\sqsubseteq)$, que nous construisons est donnée par l'ensemble fini d'actions Σ_e, les ensembles finis de ports $Port_e$ et $Port'_e \subseteq Port_e$, un ensemble fini de formules atomiques At_e, un ensemble maximal consistant de littéraux I_{0e} sur At_e et les automates communicants conditionnels finis $\mathcal{A}c_e$ et $\mathcal{A}c'_e$ sur Σ_e, $Port_e$ et At_e. Plus précisément, cette instance Σ_e, $Port_e$, $Port'_e$, At_e, I_{0e}, $\mathcal{A}c_e$ et $\mathcal{A}c'_e$ est construite comme suit :

- Σ_e=Σ,
- $Port_e$=$Port$,
- $Port'_e$=$Port'$,
- $At_e = At$,
- $I_{0e} = I_0$,
- $\mathcal{A}c_e = (Q, q_0, \delta)$, tel que $\mathcal{A}c_e = \mathcal{A}c^{client} \circledast \mathcal{A}c^{but}$,
- $\mathcal{A}c'_e = (Q', q'_0, \delta')$, tel que $\mathcal{A}c'_e = \mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n$.

Nous allons montrer qu' il existe un automate communicant conditionnel $\mathcal{A}c^{med}$ sur $\{!,?\} \times Port$ et At, tel que $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, I_0) \sqsubseteq Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast \mathcal{A}c^{med}, I_0)(\{!,?\} \times Port')$ ssi $Exec(\mathcal{A}c_e, I_{0e}) \sqsubseteq Exec(\mathcal{A}c'_e \circledast Ac^{L_{Port}},\ I_{0e})$ $(\{!,?\} \times Port'_e)$, avec $\sqsubseteq \in \{\leq_{si}, \subseteq_{tr}\}$.

Etant donné que $\mathcal{A}c_e = \mathcal{A}c^{client} \circledast \mathcal{A}c^{but}$, $\mathcal{A}c'_e = \mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n$, $I_{0e} = I_0$ et $Port'_e = Port'$, il est clair que pour prouver l'équivalence ci-dessus il suffit de prouver l'équivalence suivante : Il existe un automate communicant conditionnel $\mathcal{A}c^{med}$ sur $\{!,?\} \times Port$ et At tel que $Exec(\mathcal{A}c_e, I_0) \sqsubseteq Exec(\mathcal{A}c'_e \circledast Ac^{med},\ I_0)$ $(\{!,?\} \times Port')$ ssi $Exec(\mathcal{A}c_e, I_0) \sqsubseteq Exec(\mathcal{A}c'_e \circledast Ac^{L_{Port}},\ I_0)$ $(\{!,?\} \times Port')$, avec $\sqsubseteq \in \{\leq_{si}, \subseteq_{tr}\}$.

Concernant l'implication de gauche à droite, supposons qu'il existe un automate communicant conditionnel $\mathcal{A}c^{med}$ sur $\{!,?\} \times Port$ et At tel que :

$$Exec(\mathcal{A}c_e, I_0) \sqsubseteq Exec(\mathcal{A}c'_e \circledast Ac^{med}, I_0)(\{!,?\} \times Port'), \sqsubseteq \in \{\leq_{si}, \subseteq_{tr}\}. \tag{4.3}$$

D'après le le lemme 4.2.3 on a :

$$Exec(\mathcal{A}c'_e \circledast Ac^{med}, I_0) \leq_{si} Exec(\mathcal{A}c'_e \circledast Ac^{L_{Port}}, I_0)(\emptyset) \tag{4.4}$$

A partir de 4.3 et 4.4 on obtient :

$$Exec(\mathcal{A}c_e, I_0) \sqsubseteq Exec(\mathcal{A}c'_e \circledast Ac^{L_{Port}}, I_0)(\{!,?\} \times Port'), \sqsubseteq \in \{\leq_{si}, \subseteq_{tr}\}. \tag{4.5}$$

Concernant l'implication de droite à gauche, elle est triviale, supposons que :

$$Exec(\mathcal{A}c_e, I_0) \sqsubseteq Exec(\mathcal{A}c'_e \circledast Ac^{L_{Port}}, I_0)(\{!,?\} \times Port'), \sqsubseteq \in \{\leq_{si}, \subseteq_{tr}\}.$$

Dans ce cas, il suffit de considérer $Ac^{med} = Ac^{L_{Port}}$ pour avoir :

$$Exec(\mathcal{A}c_e, I_0) \sqsubseteq Exec(\mathcal{A}c'_e \circledast Ac^{med}, I_0)(\{!,?\} \times Port'), \sqsubseteq \in \{\leq_{si}, \subseteq_{tr}\}.$$

Ceci conclut la preuve. □

Nous avons montré que le problème de la composition de services pour la simulation et l'inclusion de traces pouvait se réduire au problème de l'équivalence avec le médiateur large lorsque la simulation et l'inclusion de traces sont considérées. Dans ce qui suit, nous montrons que le problème de l'équivalence avec le médiateur large pour ces deux relations est décidable. Cependant, nous avons besoin de la proposition ci-dessous. Dans cette proposition, nous considérons un ensemble maximal consistant $I_0 \in 2^{Li(At)}$ et des automates communicants conditionnels $\mathcal{A}c = (Q, q_0, \delta)$ et $\mathcal{A}c' = (Q', q'_0, \delta')$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At. De plus, nous associons à $\mathcal{A}c'$ un automate $AutoBoucle(\mathcal{A}c')$. Ce dernier est construit de sorte que toute transition exécutable par $Exec(\mathcal{A}c', I_0)$ soit exécutable par $AutoBoucle(\mathcal{A}c')$. Aussi, dans tout ces états l'automate $AutoBoucle(\mathcal{A}c')$ boucle sur toute les actions de communication pour simuler le médiateur large. Cependant, il faut souligner que dans l'automate $AutoBoucle(\mathcal{A}c')$ les actions de communication n'ont pas de conditions concernant la présence ou pas de messages dans les ports. Pour une meilleur compréhension, considérons l'exemple suivant.

Exemple 4.2.6. *Considérons $\Sigma = \{a\}$, $Port = \{\pi_1, \pi_2\}$ et $Port' = \{\pi_2\}$, $At = \{p\}$, $I_0 = \{\neg p\}$ et l'automate communicant conditionnel $\mathcal{A}c'$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At représenté dans la Figure 4.8. Le médiateur large $\mathcal{A}c^{L_{Port}}$ et l'automate $AutoBoucle(\mathcal{A}c')$ associé à $\mathcal{A}c$ sont représentés dans la Figure 4.9.*

Nous définissons formellement l'automate $AutoBoucle(\mathcal{A}c')$ comme suit.

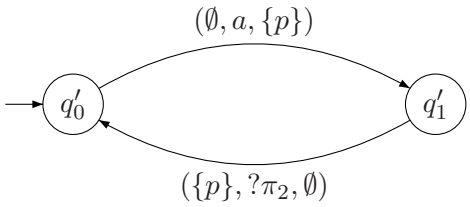

FIG. 4.8 – Automate communicant conditionnel $\mathcal{A}c'$.

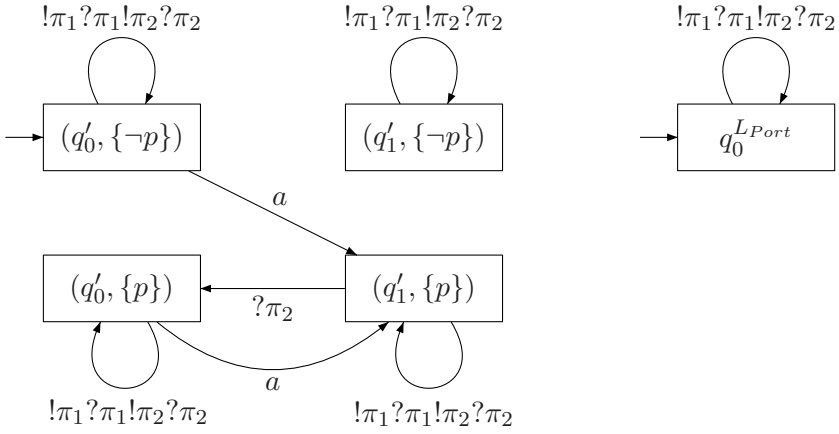

FIG. 4.9 – À gauche l'automate $AutoBoucle(\mathcal{A}c')$ et à droite le médiateur large $\mathcal{A}c^{L_{Port}}$.

Définition 4.2.7. Soit $I_0 \in 2^{Li(At)}$ un ensemble maximal consistant et un automate communicant conditionnel fini $\mathcal{A}c' = (Q', q'_0, \delta')$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At. Soit l'automate fini $AutoBoucle(\mathcal{A}c') = (Q, q_0, \rightarrow)$ sur $\Sigma \cup (\{!,?\} \times Port)$ tel que :

- $Q = \{(q', I) \mid q' \in Q' \text{ et } I \subseteq Li(At) \text{ une partie maximale consistante}\}$,
- $q_0 = (q'_0, I_0)$,
- $\rightarrow \subseteq Q \times (\Sigma \cup (\{!,?\} \times Port)) \times Q$ est la relation de transition définie par $((q'_1, I_1), a, (q'_2, I_2)) \in \rightarrow_{\mathcal{A}}$ ssi $a \in \{!,?\} \times Port$ et $(q'_2, I_2) = (q'_1, I_1)$ ou il existe une transition $(J, q'_1, a, q'_2, J') \in \delta'$ telle que $J \subseteq I_1$ et $I_2 = (I_1 \setminus \neg J') \cup J'$ où $\neg J' = \{p : \neg p \in J'\} \cup \{\neg p : p \in J'\}$.

Dans les preuves des propositions suivantes, nous utilisons l'automate $\mathcal{A}^{Port}$. Dans cet automate, un état représente une fonction $\gamma \in \Gamma$, qui associe à chaque port le nombre de messages qu'il contient. Dans $\mathcal{A}^{Port}$, les actions qui peuvent être exécutées sont uniquement des actions de communication. Les actions de la forme $?\pi$, $\pi \in Port$, ne peuvent être exécutées qu'à partir des états tels que le port π contient au moins un message. Lorsque l'action $?\pi$ est effectuée, le nombre de messages dans le port π est décrémenté de 1. Réciproquement, lorsque l'action $!\pi$ est effectuée, le nombre de messages dans le port π est incrémenté de 1.

Définition 4.2.8. L'automate $\mathcal{A}^{Port} = (\Gamma, \gamma_0, \rightarrow_{\mathcal{A}^{Port}})$ sur $\{!,?\} \times Port$ est défini comme suit :

- $\Gamma = \{\gamma : Port \rightarrow \mathbb{N}\}$,
- pour tout port $\pi \in Port$, $\gamma_0(\pi) = 0$ et
- $\rightarrow_{\mathcal{A}^{Port}} \subseteq \Gamma \times (\{!,?\} \times Port) \times \Gamma$ la relation de transition définie par $\gamma \rightarrow^a_{\mathcal{A}^{Port}} \gamma'$ ssi pour tout $\pi' \neq \pi$, $\gamma'(\pi') = \gamma(\pi')$ et
  - $a = ?\pi$, $\gamma(\pi) > 0$ et $\gamma'(\pi) = \gamma(\pi) - 1$ ou
  - $a = !\pi$ et $\gamma'(\pi) = \gamma(\pi) + 1$.

Exemple 4.2.9. *Un exemple de l'automate $\mathcal{A}^{Port}$ est illustré dans la Figure 4.10. Dans cet exemple, $Port = \{\pi_1, \pi_2\}$ et $n \in \mathbb{N}$.*

Pour rendre les preuves des propositions suivantes plus claires, nous utilisons l'automate $\mathcal{A} = (Q^{\mathcal{A}}, q_0^{\mathcal{A}}, \rightarrow_{\mathcal{A}})$ pour représenter l'automate $AutoBoucle(\mathcal{A}c')$, l'automate $\mathcal{A}^1 = (Q^1, q_0^1, \rightarrow_{\mathcal{A}^1})$ sur $\Sigma \cup (\{!,?\} \times Port)$ pour représenter l'automate $Exec(\mathcal{A}c, I_0)$. De la même façon, nous utilisons l'automate $\mathcal{A}^2 = (Q^2, q_0^2, \rightarrow_{\mathcal{A}^2})$ sur $\Sigma \cup (\{!,?\} \times Port)$ pour représenter l'automate $Exec(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port}}, I_0)$. Ainsi, $Q^1 = Q \times \Gamma \times 2^{Li(At)}$ et $Q^2 = (Q' \times Q^{L_{Port}}) \times \Gamma \times 2^{Li(At)}$.

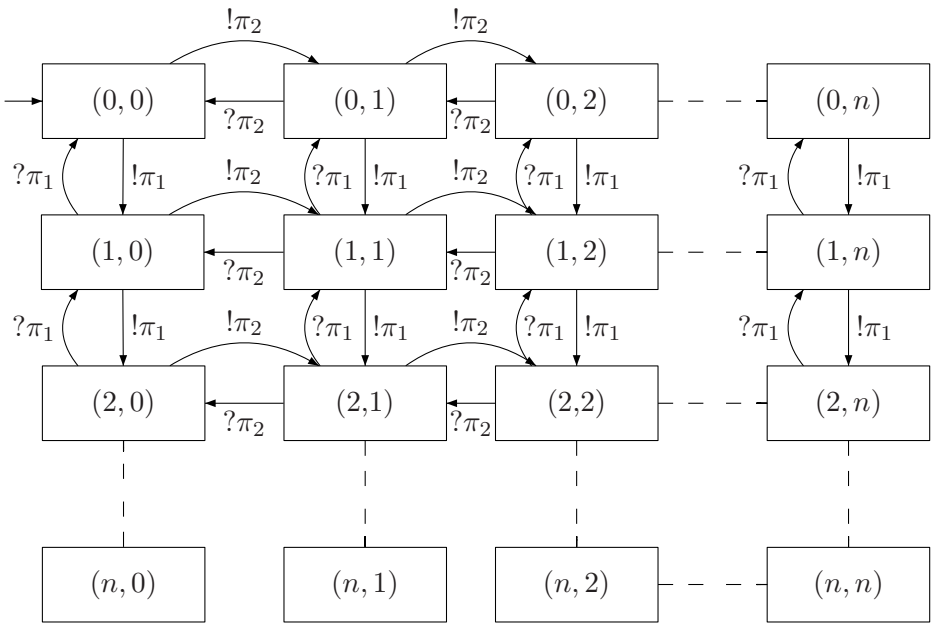

FIG. 4.10 – Automate $\mathcal{A}^{Port}$ avec $n \in \mathbb{N}$.

Proposition 4.2.10.

$\mathcal{A} \times^{\{!,?\}\times Port} \mathcal{A}^{Port} \leq_{si} Exec(\mathcal{A}c' \circledast Ac^{L_{Port}}, I_0)(\{!,?\} \times Port')$.

Démonstration. Nous allons montrer que $\mathcal{A} \times^{\{!,?\}\times Port} \mathcal{A}^{Port} \leq_{si} \mathcal{A}^2(\emptyset)$. Cela implique $\mathcal{A} \times^{\{!,?\}\times Port} \mathcal{A}^{Port} \leq_{si} \mathcal{A}^2(\{!,?\} \times Port')$. Soit la relation $Z \subseteq (Q^{\mathcal{A}} \times \Gamma) \times Q^2$ définie, pour tout $q', q'' \in Q'$, $I', I'' \in 2^{Li(At)}$ et $\gamma', \gamma'' \in \Gamma$, par :

$$((q', I'), \gamma')Z((q'', q_0^{L_{Port}}), \gamma'', I'') \text{ ssi } q'' = q',\ I'' = I' \text{ et } \gamma'' = \gamma'.$$

Montrons que Z est une relation de simulation.

- Il est clair que $((q'_0, I_0),\ \gamma_0)Z\ ((q'_0, q_0^{L_{Port}}),\ \gamma_0,\ I_0)$.
- Soient $((q'_1, I_1), \gamma_1)$ et $((q'_2, I_2), \gamma_2)$ des éléments de $Q^{\mathcal{A}} \times \Gamma$ tels que $((q'_1, I_1),\ \gamma_1)\ Z\ ((q'_1, q_0^{L_{Port}}),\ \gamma_1,\ I_1)$ et $((q'_1, I_1),\ \gamma_1) \rightarrow^a_{\mathcal{A}\times^{\{!,?\}\times Port}\mathcal{A}^{Port}} ((q'_2, I_2),\ \gamma_2)$. Montrons que $((q'_1,\ q_0^{L_{Port}}),\ \gamma_1,\ I_1) \rightarrow^a_{\mathcal{A}^2} ((q'_2, q_0^{L_{Port}}),\ \gamma_2,\ I_2)$.
  Deux cas sont possibles pour a : $a \in \Sigma$ et $a \in \{!,?\} \times Port$. Considérons le cas où $a \in \Sigma$. Par définition de $\times^{\{!,?\}\times Port}$, si $((q'_1, I_1), \gamma_1) \rightarrow^a_{\mathcal{A}\times^{\{!,?\}\times Port}\mathcal{A}^{Port}} ((q'_2, I_2), \gamma_2)$ alors $(q'_1, I_1) \rightarrow^a_{\mathcal{A}} (q'_2, I_2)$ et $\gamma_2 = \gamma_1$. Par construction de $\mathcal{A}$, $(q'_1, I_1) \rightarrow^a_{\mathcal{A}} (q'_2, I_2)$ implique qu'il existe $J, J' \in Li(At)$ tels que $(J,\ q'_1,\ a,\ q'_2,\ J') \in \delta'$, $J' \subseteq I_1$ et

$I_2 = (I_1 \setminus \neg J) \cup J'$. Ainsi, par construction de $\mathcal{A}^2$, $((q'_1, q_0^{L_{Port}}), \gamma_1, I_1) \rightarrow^a_{\mathcal{A}^2} ((q'_2, q_0^{L_{Port}}), \gamma_2, I_2)$.
Dans le cas où $a = ?\pi$, $\pi \in Port$ (le raisonnement est le même dans le cas où $a = !\pi$). Par définition de $\times^{\{!,?\}\times Port}$, si $((q'_1, I_1), \gamma_1) \rightarrow^{?\pi}_{\mathcal{A}\times\{!,?\}\times Port\mathcal{A}^{Port}} ((q'_2, I_2), \gamma_2)$ alors $(q'_1, I_1) \rightarrow^{?\pi}_{\mathcal{A}} (q'_2, I_2)$ et $\gamma_1 \rightarrow^{?\pi}_{\mathcal{A}^{Port}} \gamma_2$. Par construction de $\mathcal{A}$, si $(q'_1, I_1) \rightarrow^{?\pi}_{\mathcal{A}} (q'_2, I_2)$ alors il existe $J, J' \in Li(At)$ tels que $(J, q'_1, ?\pi, q'_2, J') \in \delta'$, $J' \subseteq I_1$ et $I_2 = (I_1 \setminus \neg J) \cup J'$. Par construction de $\mathcal{A}^{Port}$, si $\gamma_1 \rightarrow^{?\pi}_{\mathcal{A}^{Port}} \gamma_2$ alors $\gamma_1(\pi) > 0$, $\gamma_2(\pi) = \gamma_1(\pi) - 1$ et pour tout $\pi' \neq \pi$, $\gamma_1(\pi') = \gamma_2(\pi')$. Ainsi, par construction de $\mathcal{A}^2$, $((q'_1, q_0^{L_{Port}}), \gamma_1, I_1) \rightarrow^{?\pi}_{\mathcal{A}^2} ((q'_2, q_0^{L_{Port}}), \gamma_2, I_2)$.

□

Proposition 4.2.11.
$(q'_1, I_1) \rightarrow^{\{!,?\}\times Port'^\star}_{\mathcal{A}} (q'_2, I_2) \Longrightarrow$ *pour tout* $\gamma_1 \in \Gamma$, *il existe un* $\gamma_2 \in \Gamma$ *tel que* $\gamma_2(\pi) = \gamma_1(\pi)$, *avec* $\pi \in Port \setminus Port'$ *et :*

$$((q'_1, I_1), \gamma_1) \rightarrow^{\{!,?\}\times Port'^\star}_{\mathcal{A}\times\{!,?\}\times Port\mathcal{A}^{Port}} ((q'_2, I_2), \gamma_2).$$

Démonstration. Considérons deux paires d'états (q'_3, I_3) et (q'_4, I_4) dans la séquence $(q'_1, I_1) \rightarrow^{\{!,?\}\times Port'^\star}_{\mathcal{A}} (q'_2, I_2)$ telles que $(q'_3, I_3) \rightarrow^{\{?\pi'\}}_{\mathcal{A}} (q'_4, I_4)$ avec $\pi' \in Port'$. Le raisonnement est le même dans le cas $(q'_3, I_3) \rightarrow^{\{!\pi'\}}_{\mathcal{A}} (q'_4, I_4)$. Etant donnée une fonction $\gamma_1 \in \Gamma$, nous montrons que pour toutes fonction $\gamma_3 \in \Gamma$ telle que $\gamma_3(\pi) = \gamma_1(\pi)$, $\pi \in Port \setminus Port'$, il existe une fonction $\gamma_4 \in \Gamma$ telle que $\gamma_4(\pi) = \gamma_3(\pi)$, $\pi \in Port \setminus Port'$ et :

$$((q'_3, I_3), \gamma_3) \rightarrow^{\{!,?\}\times Port'^\star}_{\mathcal{A}\times\{!,?\}\times Port\mathcal{A}^{Port}} \circ \rightarrow^{\{?\pi'\}}_{\mathcal{A}\times\{!,?\}\times Port\mathcal{A}^{Port}} ((q'_4, I_4), \gamma_4).$$

Soit la fonction $\gamma_3 \in \gamma$ telle que $\gamma_3(\pi) = \gamma_1(\pi)$, $\pi \in Port \setminus Port'$. Si $\gamma_3(\pi') = 0$ alors l'action $?\pi'$ n'est pas exécutable à partir de $((q'_3, I_3), \gamma_3)$. Dans ce cas, il suffit de faire $!\pi'$ pour que $?\pi'$ soit exécutable. Plus précisément, soit la fonction $\gamma_5 \in \Gamma$ telle que $\gamma_5(\pi') = \gamma_3(\pi') + 1$ et pour tout $\pi'' \neq \pi'$, $\gamma_5(\pi'') = \gamma_3(\pi'')$. Par construction de $\mathcal{A}^{Port}$, $\gamma_3 \rightarrow^{!\pi'}_{\mathcal{A}^{Port}} \gamma_5$. De plus, $(q'_3, I_3) \rightarrow^{\{!\pi'\}}_{\mathcal{A}} (q'_3, I_3)$. Ainsi, $((q'_3, I_3), \gamma_3) \rightarrow^{\{!\pi'\}}_{\mathcal{A}\times\{!,?\}\times Port\mathcal{A}^{Port}} ((q'_3, I_3), \gamma_5)$. De plus, $\gamma_5(\pi') > 0$ et pour tout $\pi \in Port \setminus Port'$, $\gamma_5(\pi) = \gamma_3(\pi)$. Considérons la fonction $\gamma_4 \in \Gamma$ telle que $\gamma_4(\pi') = \gamma_5(\pi') - 1$ et pour tout $\pi'' \neq \pi'$, $\gamma_4(\pi'') = \gamma_5(\pi'')$. Par construction de $\mathcal{A}^{Port}$, $\gamma_5 \rightarrow^{?\pi'}_{\mathcal{A}^{Port}} \gamma_4$. De plus, $(q'_3, I_3) \rightarrow^{\{?\pi'\}}_{\mathcal{A}} (q'_4, I_4)$. Ainsi, $((q'_3, I_3), \gamma_5) \rightarrow^{\{?\pi'\}}_{\mathcal{A}\times\{!,?\}\times Port\mathcal{A}^{Port}} ((q'_4, I_4), \gamma_4)$. Par conséquent, $((q'_3, I_3), \gamma_3) \rightarrow^{\{!,?\}\times Port'^\star}_{\mathcal{A}\times\{!,?\}\times Port\mathcal{A}^{Port}} \circ \rightarrow^{\{?\pi'\}}_{\mathcal{A}\times\{!,?\}\times Port\mathcal{A}^{Port}} ((q'_4, I_4), \gamma_4)$ avec pour tout $\pi \in Port \setminus Port'$, $\gamma_4(\pi) = \gamma_3(\pi)$. Puisque, pour tout $\gamma_3 \in \Gamma$ telle que

$\gamma_3(\pi) = \gamma_1(\pi)$, $\pi \in Port \setminus Port'$, il existe une fonction $\gamma_4 \in \Gamma$ telle que $\gamma_4(\pi) = \gamma_3(\pi)$, $\pi \in Port \setminus Port'$ et :

$$((q'_3, I_3), \gamma_3) \rightarrow^{\{!,?\}\times Port'}_{\mathcal{A}\times\{!,?\}\times Port\,\mathcal{A}^{Port}}{}^{\star} \circ \rightarrow^{\{?\pi'\}}_{\mathcal{A}\times\{!,?\}\times Port\,\mathcal{A}^{Port}} ((q'_4, I_4), \gamma_4).$$

Alors, nous pouvons en déduire, par induction sur le chemin $(q'_1, I_1) \rightarrow^{\{!,?\}\times Port'^{\star}}_{\mathcal{A}} (q'_2, I_2)$, qu' il existe un $\gamma_2 \in \Gamma$ tel que $\gamma_2(\pi) = \gamma_1(\pi)$, avec $\pi \in Port \setminus Port'$ et :

$$((q'_1, I_1), \gamma_1) \rightarrow^{\{!,?\}\times Port'}_{\mathcal{A}\times\{!,?\}\times Port\,\mathcal{A}^{Port}}{}^{\star} ((q'_2, I_2), \gamma_2).$$

□

Proposition 4.2.12.
$\mathcal{A}^1 \leq_{si} \mathcal{A}\ (\{!,?\}\times Port') \Longrightarrow \mathcal{A}^1 \leq_{si} \mathcal{A}\times^{\{!,?\}\times Port}\mathcal{A}^{Port}(\{!,?\}\times Port')$.

Démonstration. Soit la relation $Z \subseteq Q^1 \times Q^{\mathcal{A}}$ telle que Z est une simulation modulo $\{!,?\} \times Port'$ et $(q_0, \gamma_0, I_0)Z(q'_0, I_0)$. Soit $Z' \subseteq Q^1 \times (Q^{\mathcal{A}} \times \Gamma)$ définie par $(q, \gamma, I)Z'((q', I'), \gamma')$ ssi $(q, \gamma, I)Z(q', I')$ et pour tout $\pi \in Port \setminus Port'$, $\gamma'(\pi) = \gamma(\pi)$. Par définition de Z', sachant que $(q_0, \gamma_0, I_0)Z(q'_0, I_0)$ on a $(q_0, \gamma_0, I_0)Z'((q'_0, I_0), \gamma_0)$. Il nous reste à vérifier que Z' est une simulation modulo $\{!,?\} \times Port'$. Soient $a \in \Sigma \cup (\{!,?\} \times (Port \setminus Port'))$, $(q_1, \gamma_1, I_1), (q_2, \gamma_2, I_2) \in Q^1$ et $((q'_1, I'_1), \gamma'_1) \in (Q^{\mathcal{A}} \times \Gamma)$ tels que $(q_1, \gamma_1, I_1)Z'((q'_1, I'_1), \gamma'_1)$ et :

$$(q_1, \gamma_1, I_1) \rightarrow^{\{!,?\}\times Port'^{\star}}_{\mathcal{A}^1} \circ \rightarrow^{\{a\}}_{\mathcal{A}^1} \circ \rightarrow^{\{!,?\}\times Port'^{\star}}_{\mathcal{A}^1} (q_2, \gamma_2, I_2).$$

Montrons qu'il existe $q'_2 \in Q'$, $I'_2 \in 2^{Li(At)}$ et $\gamma'_2 \in \Gamma$ tels que $(q_2, \gamma_2, I_2)Z(q'_2, I'_2)$, que pour tout $\pi \in Port \setminus Port'$, $\gamma'_2(\pi) = \gamma_2(\pi)$ et que :

$$((q'_1, I'_1), \gamma'_1) \rightarrow^{\{!,?\}\times Port'}_{\mathcal{A}\times\{!,?\}\times Port\,\mathcal{A}^{Port}}{}^{\star} \circ \rightarrow^{\{a\}}_{\mathcal{A}\times\{!,?\}\times Port\,\mathcal{A}^{Port}} \circ \rightarrow^{\{!,?\}\times Port'}_{\mathcal{A}\times\{!,?\}\times Port\,\mathcal{A}^{Port}}{}^{\star} ((q'_2, I'_2), \gamma'_2).$$

Par construction de Z', pour tout $\pi \in Port \setminus Port'$, $\gamma'_1(\pi) = \gamma_1(\pi)$ et $(q_1, \gamma_1, I_1)Z(q'_1, I'_1)$. Sachant que Z est une simulation modulo $\{!,?\} \times Port'$, il existe $q'_2 \in Q'$ et $I'_2 \in 2^{Li(At)}$ tels que $(q_2, \gamma_2, I_2)Z(q'_2, I'_2)$ et :

$$(q'_1, I'_1) \rightarrow^{\{!,?\}\times Port'^{\star}}_{\mathcal{A}} \circ \rightarrow^{\{a\}}_{\mathcal{A}} \circ \rightarrow^{\{!,?\}\times Port'^{\star}}_{\mathcal{A}} (q'_2, I'_2). \tag{4.6}$$

Considérons les deux paires d'états (q'_3, I'_3) et (q'_4, I'_4) dans la séquence 4.6 telles que $(q'_3, I_3) \rightarrow^{\{a\}}_{\mathcal{A}} (q'_4, I_4)$. D'après la proposition 4.2.11, puisque $(q'_1, I_1) \rightarrow^{\{!,?\}\times Port'^{\star}}_{\mathcal{A}} (q'_3, I_3)$ alors il existe un $\gamma'_3 \in \Gamma$ tel que : $\gamma'_3(\pi) = \gamma'_1(\pi)$, pour tout $\pi \in Port \setminus Port'$ et :

$$((q'_1, I'_1), \gamma'_1) \rightarrow^{\{!,?\}\times Port'}_{\mathcal{A}\times^{\{!,?\}\times Port}\mathcal{A}^{Port}}{}^{\star}((q'_3, I'_3), \gamma'_3).$$

Deux cas sont possibles pour a : $a \in \Sigma$ ou $a \in \{!, ?\} \times (Port \setminus Port')$. Considérons le cas où $a \in \Sigma$. Considérons la fonction $\gamma'_4 = \gamma'_3$. Par construction de $\mathcal{A} \times^{\{!,?\}\times Port} \mathcal{A}^{Port}$, puisque $(q'_3, I'_3) \rightarrow^{\{a\}}_{\mathcal{A}} (q'_4, I'_4)$ alors

$$((q'_3, I'_3), \gamma'_3) \rightarrow^{\{a\}}_{\mathcal{A}\times^{\{!,?\}\times Port}\mathcal{A}^{Port}} ((q'_4, I'_4), \gamma'_4).$$

avec pour tout $\pi \in Port \setminus Port'$, $\gamma'_4(\pi) = \gamma'_3(\pi)$.
Considérons le cas où $a = ?\pi'$, $\pi' \in Port \setminus Port'$ (le raisonnement est le même pour $a = !\pi'$). On sait que $(q_1, \gamma_1, I_1) \rightarrow^{\{!,?\}\times Port'^{\star}}_{\mathcal{A}^1} \circ \rightarrow^{\{?\pi'\}}_{\mathcal{A}^1} \circ \rightarrow^{\{!,?\}\times Port'^{\star}}_{\mathcal{A}^1} (q_2, \gamma_2, I_2)$. Il est clair que $\gamma_1(\pi') > 0$. De plus, $\gamma'_3(\pi) = \gamma'_1(\pi)$, pour tout $\pi \in Port \setminus Port'$. Il s'ensuit que $\gamma'_3(\pi') > 0$. Considérons la fonction $\gamma'_4 \in \Gamma$ telle que $\gamma'_4(\pi') = \gamma'_3(\pi') - 1$ et pour tout $\pi'' \neq \pi'$, $\gamma'_4(\pi'') = \gamma'_3(\pi'')$. Par construction de $\mathcal{A}^{Port}$, $\gamma'_3 \rightarrow^{\{?\pi'\}}_{\mathcal{A}^{Port}} \gamma'_4$. De plus, $(q'_3, I'_3) \rightarrow^{\{?\pi'\}}_{\mathcal{A}} (q'_4, I'_4)$. Par construction de $\mathcal{A} \times^{\{!,?\}\times Port} \mathcal{A}^{Port}$, $((q'_3, I'_3), \gamma'_3) \rightarrow^{\{?\pi'\}}_{\mathcal{A}\times^{\{!,?\}\times Port}\mathcal{A}^{Port}} ((q'_4, I'_4), \gamma'_4)$. D'après la proposition 4.2.11, puisque $(q'_4, I'_4) \rightarrow^{\{!,?\}\times Port'^{\star}}_{\mathcal{A}} (q'_2, I'_2)$ alors il existe un $\gamma'_2 \in \Gamma$ tel que : $\gamma'_2(\pi) = \gamma'_4(\pi)$, pour tout $\pi \in Port \setminus Port'$ et :

$$((q'_4, I'_4), \gamma'_4) \rightarrow^{\{!,?\}\times Port'}_{\mathcal{A}\times^{\{!,?\}\times Port}\mathcal{A}^{Port}}{}^{\star}((q'_2, I'_2), \gamma'_2).$$

Il reste à prouver que pour tout $\pi \in Port \setminus Port'$, $\gamma'_2(\pi) = \gamma_2(\pi)$. Dans le cas où $a \in \Sigma$, si $(q_1, \gamma_1, I_1) \rightarrow^{\{!,?\}\times Port'^{\star}}_{\mathcal{A}^1} \circ \rightarrow^{\{a\}}_{\mathcal{A}^1} \circ \rightarrow^{\{!,?\}\times Port'^{\star}}_{\mathcal{A}^1} (q_2, \gamma_2, I_2)$ alors $\gamma_2(\pi) = \gamma_1(\pi)$, pour tout $\pi \in Port \setminus Port'$. De la même façon, si

$$((q'_1, I'_1), \gamma'_1) \rightarrow^{\{!,?\}\times Port'}_{\mathcal{A}\times^{\{!,?\}\times Port}\mathcal{A}^{Port}}{}^{\star} \circ \rightarrow^{\{a\}}_{\mathcal{A}\times^{\{!,?\}\times Port}\mathcal{A}^{Port}} \circ \rightarrow^{\{!,?\}\times Port'}_{\mathcal{A}\times^{\{!,?\}\times Port}\mathcal{A}^{Port}}{}^{\star}((q'_2, I'_2), \gamma'_2)$$

alors $\gamma'_2(\pi) = \gamma'_1(\pi)$, pour tout $\pi \in Port \setminus Port'$. Sachant que $\gamma'_1(\pi) = \gamma_1(\pi)$, pour tout $\pi \in Port \setminus Port'$ on a $\gamma'_2(\pi) = \gamma_2(\pi)$, pour tout $\pi \in Port \setminus Port'$. Dans le cas $a = ?\pi'$, $\pi' \in Port \setminus Port'$ (le raisonnement est le même pour $a' = !\pi'$), $\gamma_2(\pi') = \gamma_1(\pi') - 1$ et pour tout $\pi'' \neq \pi'$ $\gamma_2(\pi'') = \gamma_1(\pi'')$. De la même façon, $\gamma'_2(\pi') = \gamma'_1(\pi') - 1$ et pour tout $\pi'' \neq \pi'$ $\gamma'_2(\pi'') = \gamma'_1(\pi'')$. Sachant que $\gamma'_1(\pi) = \gamma_1(\pi)$, pour tout $\pi \in Port \setminus Port'$ on a $\gamma'_2(\pi) = \gamma_2(\pi)$, pour tout $\pi \in Port \setminus Port'$. □

Proposition 4.2.13.

1. $Exec(\mathcal{A}c' \circledast Ac^{L_{Port}}, I_0) \leq_{si} \mathcal{A}(\{!, ?\} \times Port')$.
2. $Exec(\mathcal{A}c, I_0) \leq_{si} \mathcal{A}\ (\{!, ?\} \times Port') \Longrightarrow Exec(\mathcal{A}c, I_0) \leq_{si} Exec(\mathcal{A}c' \circledast Ac^{L_{Port}}, I_0)(\{!, ?\} \times Port')$.

Démonstration. Considérons le point 1. Montrons que $\mathcal{A}^2 \leq_{si} \mathcal{A}(\{!,?\} \times Port')$. Dans ce qui suit nous montrons que $\mathcal{A}^2 \leq_{si} \mathcal{A}(\emptyset)$. Cela impliquera $\mathcal{A}^2 \leq_{si} \mathcal{A}(\{!,?\} \times Port')$. Soit la relation $Z \subseteq Q^2 \times Q^{\mathcal{A}}$ définie par $((q', q_0^{L_{Port}}), \gamma, I) Z (q', I)$. Il est clair que $((q'_0, q_0^{L_{Port}}), \gamma_0, I_0) Z (q'_0, I_0)$. Soit $((q'_1, q_0^{L_{Port}}), \gamma_1, I_1) \in Q^2$. Montrons que pour toute action $a \in \Sigma \cup (\{!,?\} \times Port)$, s'il existe un état $((q'_2, q_0^{L_{Port}}), \gamma_2, I_2) \in Q^2$ tel que $((q'_1, q_0^{L_{Port}}), \gamma_1, I_1) \rightarrow^a_{\mathcal{A}^2} ((q'_2, q_0^{L_{Port}}), \gamma_2, I_2)$ alors $(q'_1, I_1) \rightarrow^a_{\mathcal{A}} (q'_2, I_2)$. Soit $((q'_2, q_0^{L_{Port}}), \gamma_2, I_2) \in Q^2$ tel que :

$$((q'_1, q_0^{L_{Port}}), \gamma_1, I_1) \rightarrow^a_{\mathcal{A}^2} ((q'_2, q_0^{L_{Port}}), \gamma_2, I_2).$$

Deux cas sont possibles pour $a : a \in \Sigma$ ou $a \in \{!,?\} \times Port$. Considérons le cas où $a \in \Sigma$. Par construction de $\mathcal{A}^2$, si $((q'_1, q_0^{L_{Port}}), \gamma_1, I_1) \rightarrow^a_{\mathcal{A}^2} ((q'_2, q_0^{L_{Port}}), \gamma_2, I_2)$ alors il existe $J, J' \in Li(At)$ tels que $J \subseteq I_1$, $I_2 = (I_1 \setminus \neg J') \cup J'$ et $(J, q_1, a, q_2, J') \in \delta'$. Par construction de $\mathcal{A}$ on a $(q'_1, I_1) \rightarrow^a_{\mathcal{A}} (q'_2, I_2)$.
Considérons le cas où $a = ?\pi$, $\pi \in Port$ (le raisonnement est le même dans le cas $a = !\pi$). Dans ce cas l'action $?\pi$ peut être exécutée par $\mathcal{A}c'$ ou par $\mathcal{A}c^{L_{Port}}$. Si l'action est exécutée par $\mathcal{A}c'$ alors le raisonnement est le même que celui utilisé dans le cas où $a \in \Sigma$. Supposons que $\mathcal{A}c^{L_{Port}}$ exécute l'action $?\pi$. Sachant que $(\emptyset, q_0^{L_{Port}}, ?\pi, q_0^{L_{Port}}, \emptyset)$ et $((q'_1, q_0^{L_{Port}}), \gamma_1, I_1) \rightarrow^{?\pi}_{\mathcal{A}^2} ((q'_2, q_0^{L_{Port}}), \gamma_2, I_2)$ on a $q'_2 = q'_1$ et $I_2 = I_1$. Par construction de $\mathcal{A}$ on sait que $(q'_1, I_1) \rightarrow^{?\pi}_{\mathcal{A}} (q'_1, I_1)$. Il en résulte que $(q'_1, I_1) \rightarrow^{?\pi}_{\mathcal{A}} (q'_2, I_2)$.

Considérons le point 2. Il suffit d'utiliser la proposition 4.2.10 ainsi que la proposition 4.2.12. □

Proposition 4.2.14. *Soient les ensembles finis Σ, $Port$, At, I_0 et un automate communicant conditionnel fini $\mathcal{A}c$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At. Il existe un réseau de Petri $\mathcal{N}$ sur $\Sigma \cup (\{!,?\} \times Port)$, de taille exponentielle par rapport à la taille de At et un marquage initial m_0 tels que :*

$$Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Exec(\mathcal{N}, m_0) \quad (\emptyset)$$

Démonstration. D'après la proposition 3.2.19, pour tout automate communicant conditionnel $\mathcal{A}c$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At, il existe un automate communicant conditionnel $\mathcal{A}c'$ de forme canonique tel que $Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Exec(\mathcal{A}c', I_0)(\emptyset)$. La taille de $\mathcal{A}c'$ est exponentielle par rapport à la taille de At. Nous considérons dans cette preuve que $\mathcal{A}c$ est sous la forme canonique. Il nous suffit donc de prouver qu'il existe un réseau de Petri $\mathcal{N}$ de taille polynomiale par rapport à $\mathcal{A}c$ tel que $Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Exec(\mathcal{N}, m_0)(\emptyset)$. Considérons l'ensemble $\overline{Q} = \{\overline{q} \mid q \in Q\}$. Le réseau de Petri ordinaire $\mathcal{N} = (P, T, F, l)$ sur $\Sigma \cup (\{!,?\} \times Port)$ est tel que :

- $P = Q \cup \overline{Q} \cup Port \cup Li(At)$.
- $T = \delta$.
- $F = \{(p,t) \mid p \in^\bullet t\} \cup \{ (t,p) \mid p \in t^\bullet\}$ où, pour chaque transition $t \in T$ telle que $t = (J, q, a, q', J')$ les ensembles ${}^\bullet t$ et $t^\bullet$ sont définis comme suit. Trois cas sont possibles pour a : $a \in \Sigma$, $a = ?\pi$, $\pi \in Port$ ou $a = !\pi$, $\pi \in Port$. Deux cas sont possibles pour q' : $q' = q$ ou $q' \neq q$. Dans le cas où $a \in \Sigma$ et $q' \neq q$, la transition t est telle que : ${}^\bullet t = \{q, \overline{q'}\} \cup J$ et $t^\bullet = \{q', \overline{q}\} \cup J'$, comme décrit dans la Figure 4.11. Dans le cas où $a = ?\pi$, $\pi \in Port$ et $q' \neq q$, la transition t est telle que : ${}^\bullet t = \{q, \overline{q'}, \pi\} \cup J$ et $t^\bullet = \{q', \overline{q}\} \cup J'$, comme décrit dans la Figure 4.12. Dans le cas où $a = !\pi$, $\pi \in Port$ et $q' \neq q$, la transition t est telle que : ${}^\bullet t = \{q, \overline{q'}\} \cup J$ et $t^\bullet = \{q', \overline{q}, \pi\} \cup J'$, comme décrit dans la Figure 4.13. Dans le cas où $a \in \Sigma$ et $q' = q$, la transition t est telle que : ${}^\bullet t = \{q\} \cup J$ et $t^\bullet = \{q\} \cup J'$, comme décrit dans la Figure 4.14. Dans le cas où $a = ?\pi$, $\pi \in Port$ et $q' = q$, la transition t est telle que : ${}^\bullet t = \{q, \pi\} \cup J$ et $t^\bullet = \{q\} \cup J'$, comme décrit dans la Figure 4.15. Dans le cas où $a = !\pi$, $\pi \in Port$ et $q' = q$, la transition t est telle que : ${}^\bullet t = \{q\} \cup J$ et $t^\bullet = \{q, \pi\} \cup J'$, comme décrit dans la Figure 4.16.
- $l : T \to \Sigma \cup (\{!, ?\} \times Port)$ telle que $l(t) = a$ si $t = (J, q, a, q', J')$.

Une place $q \in Q$ et la place $\overline{q}$ sont des places complémentaires. Également, une place $r \in Li(At)$ et la place $\neg r$ sont des places complémentaires.

Le marquage initial $m_0 : Q \cup \overline{Q} \cup Port \cup Li(At) \to \mathbb{N}$ est défini comme suit :

$$m_0(p) = \begin{cases} 1 & \text{si } p \in (I_0 \cup \{q_0\} \cup (\overline{Q} \setminus \{\overline{q_0}\})) \\ 0 & \text{sinon} \end{cases}$$

Il est clair que la taille de $\mathcal{N}$ est polynomiale par rapport à la taille de $\mathcal{A}c$. Soulignons que dans le cas où nous considérons uniquement les marquages accessibles à partir de m_0, $Exec(\mathcal{A}c, I_0)$ et $Exec(\mathcal{N}, m_0)$ sont isomorphes. Montrons que $Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Exec(\mathcal{N}, m_0)(\emptyset)$. Soient $Q \times \Gamma \times 2^{Li(At)}$ l'ensemble des états de $Exec(\mathcal{A}c, I_0)$, M l'ensemble des états de $Exec(\mathcal{N}, m_0)$, (les marquages de $\mathcal{N}$) et $Z \subseteq (Q \times \Gamma \times 2^{Li(At)}) \times M$ la relation définie par :

$$(q, \gamma, I) Z m \text{ ssi } m(p) = \begin{cases} 1 & \text{si } p \in (I \cup \{q\} \cup (\overline{Q} \setminus \{\overline{q}\})) \\ \gamma(p) & \text{si } p \in Port \\ 0 & \text{sinon} \end{cases}$$

Nous devons montrer que Z est une bisimulation et que $(q_0, \gamma_0, I_0) Z m_0$.

Considérons les états initiaux. L'état initial de $Exec(\mathcal{N}, m_0)$ est (q_0, γ_0, I_0) où pour tout $\pi \in Port$, $\gamma_0(\pi) = 0$. L'état initial de $Exec(\mathcal{N}, m_0)$ est le marquage m_0 avec :

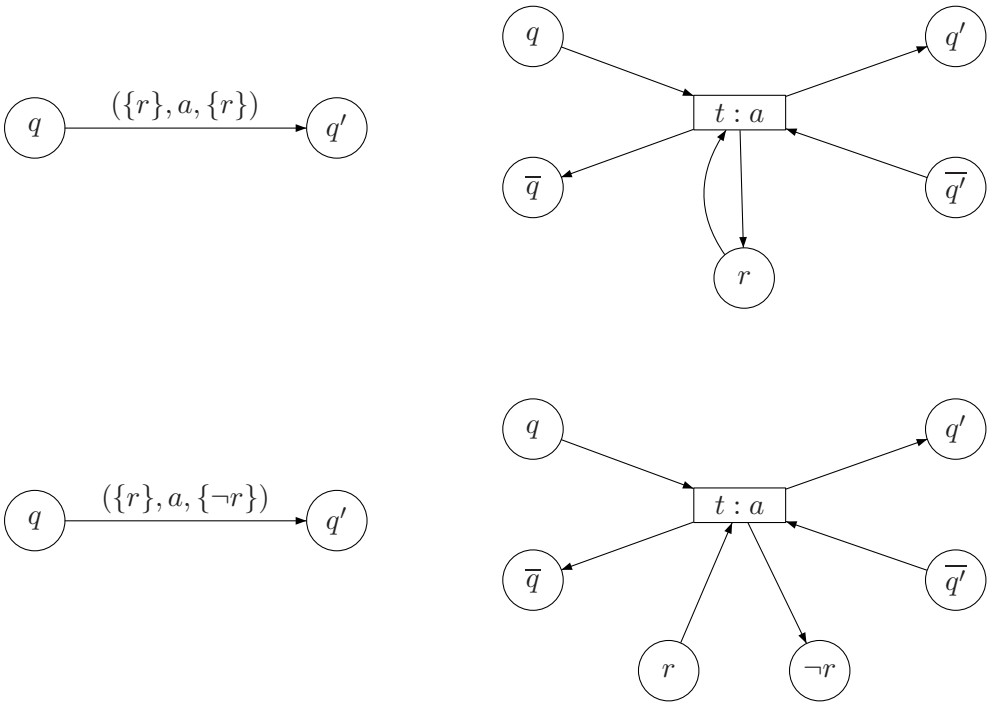

FIG. 4.11 – Construction de t dans le cas où $a \in \Sigma$. En haut à gauche, une transition $(J, q, a, q', J') \in \delta$ telle que $\{r\} = J$ et $\{r\} = J'$. En haut à droite, la transition $t \in T$ qui lui est associée. Toutes les places reliées à t sont également représentées. En bas de la figure, en considère le cas $\{r\} = J$ et $\{\neg r\} = J'$.

$$m_0(p) = \begin{cases} 1 & \text{si } p \in (I_0 \cup \{q_0\} \cup (\overline{Q} \setminus \{\overline{q_0}\})) \\ 0 & \text{sinon} \end{cases}$$

Ainsi, $(q_0, \gamma_0, I_0) Z m_0$.

Considérons un état $(q_1, \gamma_1, I_1) \in Q \times \Gamma \times 2^{Li(At)}$ et un état $m_1 \in M$ tels que $(q_1, \gamma_1, I_1) Z m_1$. Nous allons prouver les deux points suivants, pour tout $a \in \Sigma \cup (\{!, ?\} \times Port)$:

(1). S'il existe $(q_2, \gamma_2, I_2) \in Q \times \Gamma \times 2^{Li(At)}$ tel que $(q_1, \gamma_1, I_1) \rightarrow^a_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$ alors il existe $t \in T$ et $m_2 \in M$ tel que $l(t) = a$, $m_1[t\rangle m_2$ et $(q_2, \gamma_2, I_2) Z m_2$.

(2). S'il existe $t \in T$ et $m_2 \in M$ tel que $l(t) = a$, $m_1[t\rangle m_2$ alors il existe $(q_2, \gamma_2, I_2) \in Q \times \Gamma \times 2^{Li(At)}$ tel que $(q_1, \gamma_1, I_1) \rightarrow^a_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$

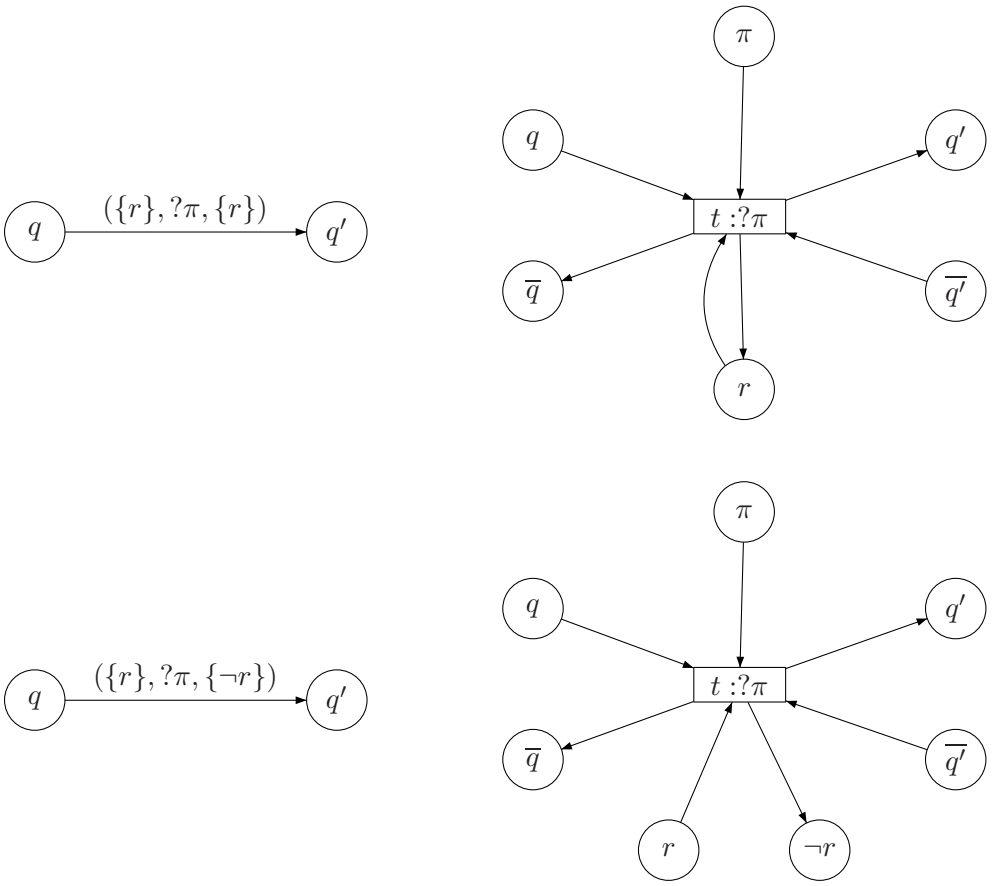

FIG. 4.12 – Construction de t dans le cas où $a = ?\pi$, $\pi \in Port$. En haut à gauche, une transition $(J, q, a, q', J') \in \delta$ telle que $\{r\} = J$ et $\{r\} = J'$. En haut à droite, la transition $t \in T$ qui lui est associée. Toutes les places reliées à t sont également représentées. En bas de la figure, en considère le cas $\{r\} = J$ et $\{\neg r\} = J'$.

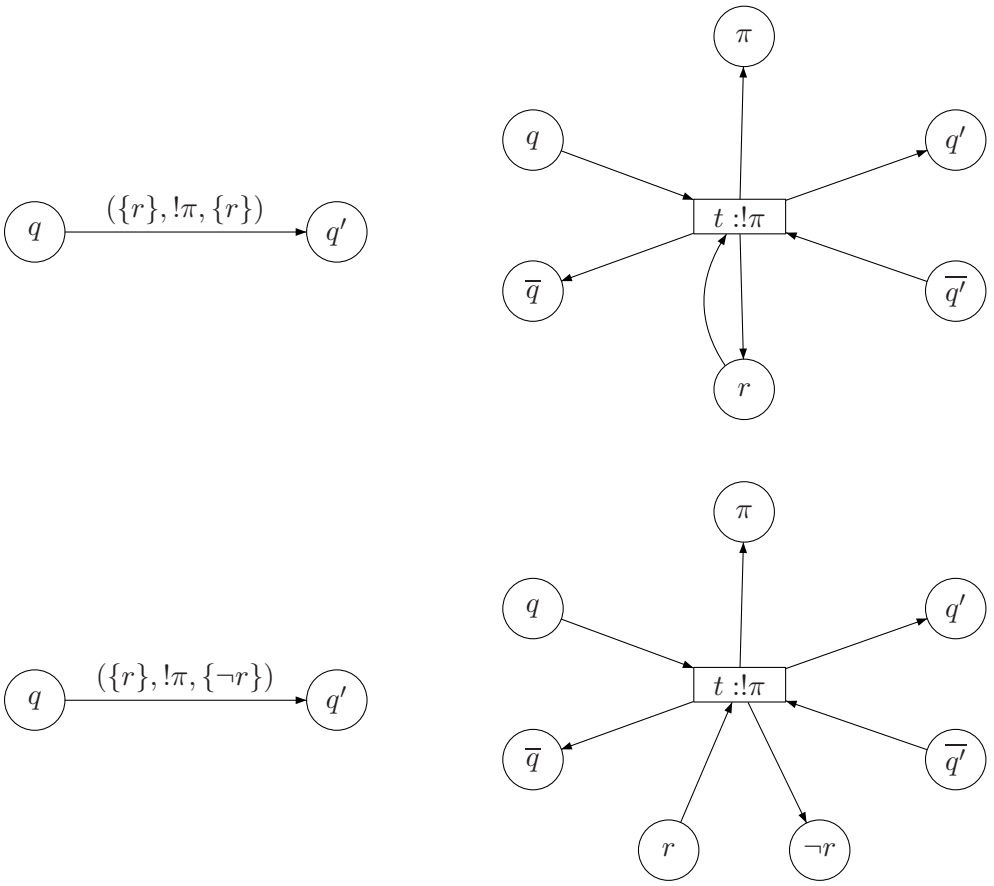

FIG. 4.13 – Construction de t dans le cas où $a = !\pi$, $\pi \in Port$. En haut à gauche, une transition $(J, q, a, q', J') \in \delta$ telle que $\{r\} = J$ et $\{r\} = J'$. En haut à droite, la transition $t \in T$ qui lui est associée. Toutes les places reliées à t sont également représentées. En bas de la figure, en considère le cas $\{r\} = J$ et $\{\neg r\} = J'$.

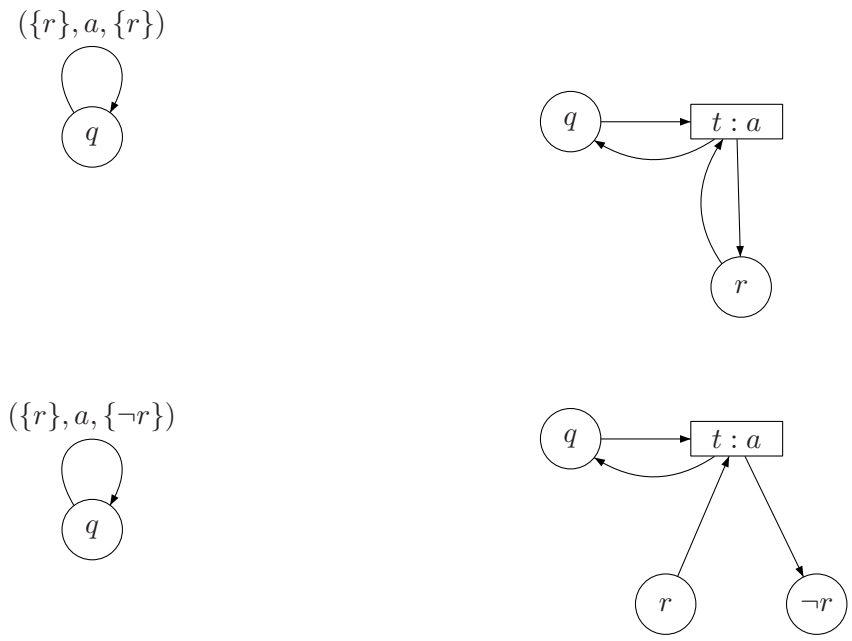

FIG. 4.14 – Construction de t dans le cas où $a \in \Sigma$. En haut à gauche, une transition $(J, q, a, q, J') \in \delta$ telle que $\{r\} = J$ et $\{r\} = J'$. En haut à droite, la transition $t \in T$ qui lui est associée. Toutes les places reliées à t sont également représentées. En bas de la figure, en considère le cas $\{r\} = J$ et $\{\neg r\} = J'$.

et $(q_2, \gamma_2, I_2) Z m_2$.

Considérons le point (1). Supposons qu'il existe $(q_2, \gamma_2, I_2) \in Q \times \Gamma \times 2^{Li(At)}$ tel que $(q_1, \gamma_1, I_1) \rightarrow^a_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$. Deux cas sont possibles pour $a : a \in \Sigma$ ou $a \in \{!, ?\} \times Port$. Considérons le cas où $a \in \Sigma$. Puisque $(q_1, \gamma_1, I_1) \rightarrow^a_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$ alors il existe deux ensembles consistants $J, J' \in Li(At)$ tel que $(J, q_1, a, q_2, J') \in \delta$, avec $J \subseteq I_1$, $I_2 = (I_1 \setminus \neg J') \cup J'$ et $\gamma_2 = \gamma_1$. Deux cas sont possibles pour $q_2 : q_2 \neq q_1$ ou $q_2 = q_1$. Considérons le cas où $q_2 \neq q_1$. Par construction de T, puisque $(J, q_1, a, q_2, J') \in \delta$ alors il existe une transition $t \in T$ telle que $l(t) = a$ et ${}^\bullet t = \{q_1, \overline{q_2}\} \cup J$ et $t^\bullet = \{q_2, \overline{q_1}\} \cup J'$. Sachant que $(q_1, \gamma_1, I_1) Z m_1$ on a

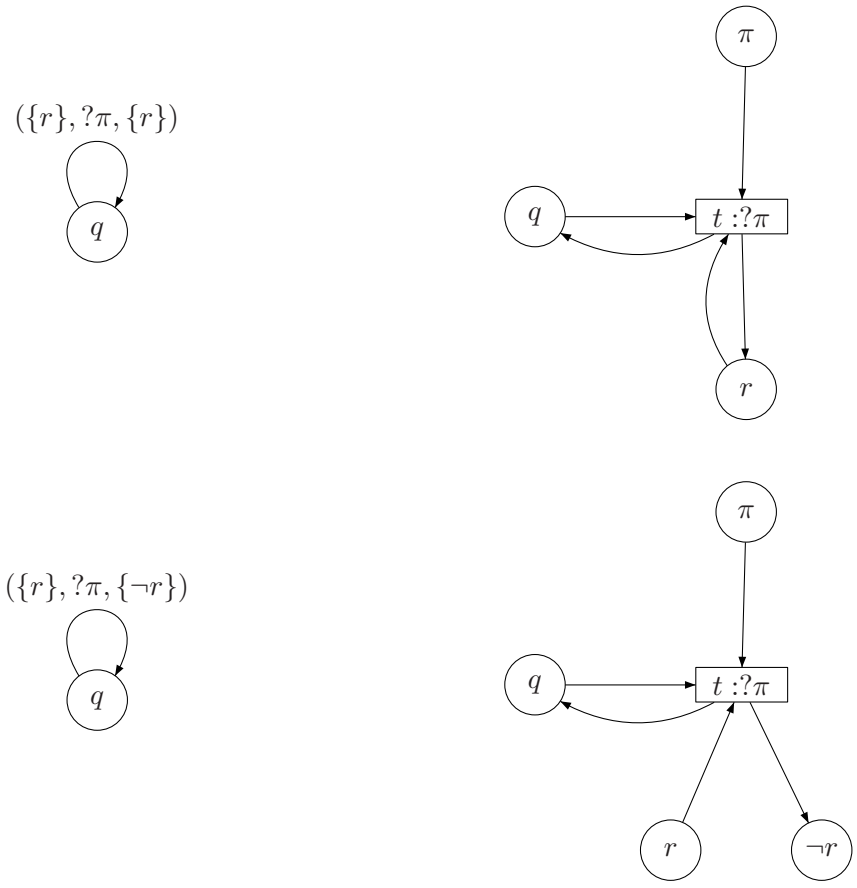

FIG. 4.15 – Construction de t dans le cas où $a = ?\pi$, $\pi \in Port$. En haut à gauche, une transition $(J, q, a, q, J') \in \delta$ telle que $\{r\} = J$ et $\{r\} = J'$. En haut à droite, la transition $t \in T$ qui lui est associée. Toutes les places reliées à t sont également représentées. En bas de la figure, en considère le cas $\{r\} = J$ et $\{\neg r\} = J'$.

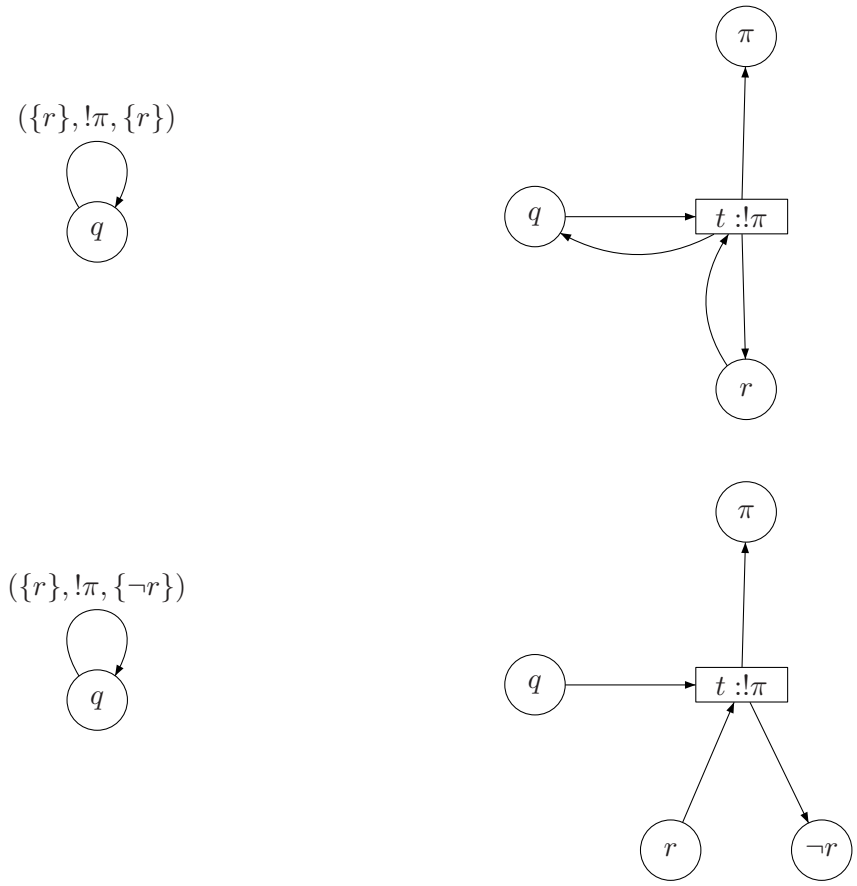

FIG. 4.16 – Construction de t dans le cas où $a = !\pi$, $\pi \in Port$. En haut à gauche, une transition $(J, q, a, q, J') \in \delta$ telle que $\{r\} = J$ et $\{r\} = J'$. En haut à droite, la transition $t \in T$ qui lui est associée. Toutes les places reliées à t sont également représentées. En bas de la figure, en considère le cas $\{r\} = J$ et $\{\neg r\} = J'$.

$$m_1(p) = \begin{cases} 1 & \text{si } p \in (I_1 \cup \{q_1\} \cup (\overline{Q} \setminus \{\overline{q_1}\})) \\ \gamma_1(p) & \text{si } p \in Port \\ 0 & \text{sinon} \end{cases}$$

Puisque $m_1(q_1) = 1$, $m_1(\overline{q_2}) = 1$ et pour tout $r \in J$, $m_1(r) = 1$ alors il existe $m_2 \in M$ telle que $m_1[t\rangle m_2$ et

$$m_2(p) = \begin{cases} m_1(p) + 1 & \text{si } p \notin {}^\bullet t \text{ et } p \in t^\bullet \\ m_1(p) - 1 & \text{si } p \in {}^\bullet t \text{ et } p \notin t^\bullet \\ m_1(p) & \text{sinon} \end{cases}$$

Par conséquent, sachant que J et J' sont consistants et que (J, q_1, a, q_2, J') est canonique alors le marquage m_2 est défini comme suit.

$$m_2(p) = \begin{cases} 1 & \text{si } p \in (I_2 \cup \{q_2\} \cup (\overline{Q} \setminus \{\overline{q_2}\})) \\ \gamma_1(p) & \text{si } p \in Port \\ 0 & \text{sinon} \end{cases}$$

Puisque, $\gamma_1 = \gamma_2$ alors par construction de Z, $(q_2, \gamma_2, I_2) Z m_2$.

Dans le cas où $q_2 = q_1$, le raisonnement est le même que dans le cas $q_2 \neq q_1$. L'unique différence est qu'il n'est pas nécessaire de vérifier que $m_1(\overline{q_2}) = 1$. De plus, $q_1 \in {}^\bullet t$ et $q_1 \in t^\bullet$, voir la Figure 4.14. Donc, $m_2(q_1) = m_1(q_1) = 1$. Aussi, $\overline{q_1} \notin {}^\bullet t$ et $\overline{q_1} \notin t^\bullet$. Par conséquent, $m_2(\overline{q_1}) = m_1(\overline{q_1}) = 0$.

Dans le cas où $a = ?\pi$, le raisonnement est le même. L'unique différence est le fait de vérifier que $m_1(\pi) > 0$, $m_2(\pi) = m_1(\pi) - 1$ et que pour tout $\pi' \neq \pi$, $m_2(\pi') = m_2(\pi)$. Par construction de $Exec(\mathcal{A}c, I_0)$, le fait que $(q_1, \gamma_1, I_1) \rightarrow^{?\pi}_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$ implique $\gamma_1(\pi) > 0$. Puisque $(q_1, \gamma_1, I_1) Z m_1$ alors $m_1(\pi) = \gamma_1(\pi)$. Sachant que $\gamma_1(\pi) > 0$ alors $m_1(\pi) > 0$. Par construction de m_2, puisque $\pi \in {}^\bullet t$ et $\pi \notin t^\bullet$, voir Figure 4.12, alors $m_2(\pi) = m_1(\pi) - 1$. De la même façon, puisque pour tout $\pi' \neq \pi$, $\pi' \notin {}^\bullet t$ et $\pi' \notin t^\bullet$ alors $m_2(\pi') = m_1(\pi')$. Dans le cas où $a = !\pi$, le raisonnement est le même. L'unique différence est le fait de vérifier que $m_2(\pi) = m_1(\pi) + 1$ et que pour tout $\pi' \neq \pi$, $m_2(\pi') = m_2(\pi)$. Par construction de m_2, puisque $\pi \in t^\bullet$ et $\pi \notin {}^\bullet t$, voir Figure 4.13, alors $m_2(\pi) = m_1(\pi) + 1$. De la même façon, puisque pour tout $\pi' \neq \pi$, $\pi' \notin {}^\bullet t$ et $\pi' \notin t^\bullet$ alors $m_2(\pi') = m_1(\pi')$.

Considérons le point (2). Supposons qu'il existe $t \in T$ et $m_2 \in M$ tel que $l(t) = a$ et $m_1[t\rangle m_2$. Par conséquent, pour tout $p \in {}^\bullet t$, $m_1(p) > 0$ et :

$$m_2(p) = \begin{cases} m_1(p) + 1 & \text{si } p \notin {}^\bullet t \text{ et } p \in t^\bullet \\ m_1(p) - 1 & \text{si } p \in {}^\bullet t \text{ et } p \notin t^\bullet \\ m_1(p) & \text{sinon} \end{cases}$$

Par construction de $\mathcal{N}$, puisque $t \in T$ et $l(t) = a$ alors il existe $q_1', q_2 \in Q$ et $J, J' \in Li(At)$ tels que $(J, q_1', a, q_2, J') \in \delta$ et $t = (J, q_1', a, q_2, J')$. Sachant que $q_1' \in {}^\bullet t$ et $m_1[t\rangle m_2$ alors $m_1(q_1') = 1$. Puisque $(q_1, \gamma_1, I_1) Z m_1$ alors q_1 est l'unique état dans Q tel que $m_1(q_1) = 1$. Donc, $q_1' = q_1$ et $(J, q_1, a, q_2, J') \in \delta$. Deux cas sont possibles pour a : $a \in \Sigma$ ou $a \in \{!, ?\} \times Port$. Considérons le cas

où $a \in \Sigma$. Deux cas sont possibles pour q_2 : $q_2 \neq q_1$ ou $q_2 = q_1$. Considérons le cas où $q_2 \neq q_1$. Dans ce cas, ${}^\bullet t = \{q_1, \overline{q_2}\} \cup J$ et $t^\bullet = \{q_2, \overline{q_1}\} \cup J'$. Le fait que $J \subseteq {}^\bullet t$ implique $m_1(r) = 1$, pour tout $r \in J$. Sachant que $(q_1, \gamma_1, I_1) Z m_1$ on a :

$$m_1(p) = \begin{cases} 1 & \text{si } p \in (I_1 \cup \{q_1\} \cup (\overline{Q} \setminus \{\overline{q_1}\})) \\ \gamma_1(p) & \text{si } p \in Port \\ 0 & \text{sinon} \end{cases}$$

Donc, $I_1 = \{r \mid r \in Li(At) \text{ et } m_1(r) = 1\}$. Ainsi $J \subseteq I_1$. Soit (q_2, γ_2, I_2) tel que $I_2 = (I_1 \setminus \neg J') \cup J'$ et $\gamma_2 = \gamma_1$. Par construction de $Exec(\mathcal{A}c, I_0)$, puisque $J \subseteq I_1$ alors $(q_1, \gamma_1, I_1) \rightarrow^a_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$. Sachant que J et J' sont consistants et que (J, q_1, a, q_2, J') est canonique alors le marquage m_2 est défini à partir de m_1 comme suit.

$$m_2(p) = \begin{cases} 1 & \text{si } p \in (I_2 \cup \{q_2\} \cup (\overline{Q} \setminus \{\overline{q_2}\})) \\ \gamma_1(p) & \text{si } p \in Port \\ 0 & \text{sinon} \end{cases}$$

Puisque, $\gamma_1 = \gamma_2$ alors par construction de Z, $(q_2, \gamma_2, I_2) Z m_2$.

Dans le cas où $q_2 = q_1$, le raisonnement est le même que dans le cas $q_2 \neq q_1$. L'unique différence est que $q_1 \in {}^\bullet t$ et $q_1 \in t^\bullet$, voir la Figure 4.14. Donc, $m_2(q_1) = m_1(q_1) = 1$. Aussi, $\overline{q_1} \notin {}^\bullet t$ et $\overline{q_1} \notin t^\bullet$. Par conséquent, $m_2(\overline{q_1}) = m_1(\overline{q_1}) = 0$.

Dans le cas où $a = ?\pi$, le raisonnement est le même. Mais nous devons en plus vérifier que $\gamma_1(\pi) > 0$. Par construction de t, $\pi \in {}^\bullet(t)$. Puisque, pour tout $p \in {}^\bullet t$, $m_1(p) > 0$ alors $m_1(\pi) > 0$. Sachant que $(q_1, \gamma_1, q_1) Z m_1$ alors $m_1(\pi) = \gamma_1(\pi)$. Ainsi, $\gamma_1(\pi) > 0$. Considérons (q_2, γ_2, I_2) tel que $I_2 = (I_1 \setminus \neg J') \cup J'$ et $\gamma_2(\pi) = \gamma_1(\pi) - 1$ et pour tout $\pi' \neq \pi$, $\gamma_2(\pi') = \gamma_2(\pi)$. Par construction de $Exec(\mathcal{A}c, I_0)$, puisque $J \subseteq I_1$ et que $\gamma_1(\pi) > 0$ alors $(q_1, \gamma_1, I_1) \rightarrow^{?\pi}_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$. Puisque $\pi \in {}^\bullet t$ et $\pi \notin t^\bullet$, voir Figure 4.12, alors $m_2(\pi) = m_1(\pi) - 1$. De la même façon, puisque pour tout $\pi' \neq \pi$, $\pi' \notin {}^\bullet t$ et $\pi' \notin t^\bullet$ alors $m_2(\pi') = m_1(\pi')$. Ainsi, $(q_2, \gamma_2, I_2) Z m_2$. Dans le cas où $a = !\pi$, le raisonnement est le même. Considérons (q_2, γ_2, I_2) tel que $I_2 = (I_1 \setminus \neg J') \cup J'$ et $\gamma_2(\pi) = \gamma_1(\pi) + 1$ et pour tout $\pi' \neq \pi$, $\gamma_2(\pi') = \gamma_2(\pi)$. Par construction de $Exec(\mathcal{A}c, I_0)$, puisque $J \subseteq I_1$ alors $(q_1, \gamma_1, I_1) \rightarrow^{!\pi}_{Exec(\mathcal{A}c, I_0)} (q_2, \gamma_2, I_2)$. Puisque $\pi \in t^\bullet$ et $\pi \notin {}^\bullet t$, voir Figure 4.13, alors $m_2(\pi) = m_1(\pi) + 1$. De la même façon, puisque pour tout $\pi' \neq \pi$, $\pi' \notin {}^\bullet t$ et $\pi' \notin t^\bullet$ alors $m_2(\pi') = m_1(\pi')$. Ainsi, $(q_2, \gamma_2, I_2) Z m_2$. □

Lemme 4.2.15. *Le problème $\mathcal{PEL}(\leq_{si})$ est décidable.*

Démonstration. Cette preuve se base sur la réduction au problème de l'équivalence entre un réseau de Petri et un automate fini. Considérons le problème de décision suivant :

Problème $\mathcal{ERA}(\leq_{si})$: problème de simulation entre un réseau de Petri et un automate

Instance : des ensembles finis d'actions Σ et $\Sigma' \subseteq \Sigma$, un réseau de Petri $\mathcal{N} = (P_{\mathcal{N}}, T_{\mathcal{N}}, F_{\mathcal{N}}, l_{\mathcal{N}})$ défini sur Σ, un marquage initial des places m_0 et un automate fini $\mathcal{A} = (q, q_0, \rightarrow)$ sur Σ.

Question : est-il vrai que $Exec(\mathcal{N}, m_0) \leq_{si} \mathcal{A}\ (\Sigma')$?

Sachant que $\mathcal{ERA}(\leq_{si})$ est décidable [Jan94], il suffit de réduire $\mathcal{PEL}(\leq_{si})$ à $\mathcal{ERA}(\leq_{si})$ afin de montrer que $\mathcal{PEL}(\leq_{si})$ est décidable. Etant donnés un ensemble fini d'actions Σ, des ensembles finis de ports $Port$ et $Port' \subseteq Port$, un ensemble fini de formules atomiques At, un ensemble maximal de littéraux I_0 sur At et des automates communicants conditionnels finis $\mathcal{A}c$ et $\mathcal{A}c'$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At. L'instance $\rho(\Sigma,\ Port,\ Port',\ At,\ I_0,\ \mathcal{A}c,\ \mathcal{A}c')$ de $\mathcal{ERA}(\leq_{si})$, que nous construisons est donnée par les ensembles finis d'actions $\Sigma_e = \Sigma \cup (\{!, ?\} \times Port)$ et $\Sigma' = \{!, ?\} \times Port'$, un marquage initial des places m_0, un réseau de Petri $\mathcal{N}$ défini sur Σ_e et un automate fini $\mathcal{A}$ défini sur Σ_e. Le réseau de Petri $\mathcal{N}$ et le marquage initial m_0 sont construits à partir de $\mathcal{A}c$ et I_0 selon la proposition 4.2.14. L'automate $\mathcal{A}$ est construit à partir de $\mathcal{A}c'$ selon la définition 4.2.7.

Il est évident que l'instance ρ est calculable par une machine de Turing en utilisant un espace fini. Montrons que :

$$Exec(\mathcal{N}, m_0) \leq_{si} \mathcal{A} \quad (\Sigma')$$
$$\Longleftrightarrow$$
$$Exec(\mathcal{A}c, I_0) \leq_{si} Exec(\mathcal{A}c' \circledast Ac^{L_{Port}}, I_0) \quad (\{!, ?\} \times Port')$$

($\Rightarrow$) Considérons l'implication de gauche à droite. Supposons que :

$$Exec(\mathcal{N}, m_0) \leq_{si} \mathcal{A} \quad (\Sigma') \tag{4.7}$$

D'après la proposition 4.2.14 :

$$Exec(\mathcal{A}c, I_0) \leq_{si} Exec(\mathcal{N}, m_0) \quad (\emptyset) \tag{4.8}$$

A partir de 4.7 et 4.8 :

$$Exec(\mathcal{A}c, I_0) \leq_{si} \mathcal{A} \quad (\Sigma') \tag{4.9}$$

D'après la proposition 4.2.13 et 4.9 :

$$Exec(\mathcal{A}c, I_0) \leq_{si} Exec(\mathcal{A}c' \circledast Ac^{L_{Port}}, I_0) \quad (\{!, ?\} \times Port') \tag{4.10}$$

($\Leftarrow$) Considérons l'implication de droite à gauche. Supposons que :

$$Exec(\mathcal{A}c, I_0) \leq_{si} Exec(\mathcal{A}c' \circledast Ac^{L_{Port}}, I_0) \quad (\{!, ?\} \times Port') \tag{4.11}$$

D'après la proposition 4.2.14 :

$$Exec(\mathcal{N}, m_0) \leq_{si} Exec(\mathcal{A}c, I_0) \quad (\emptyset) \tag{4.12}$$

A partir de 4.7 et 4.12 :

$$Exec(\mathcal{N}, m_0) \leq_{si} Exec(\mathcal{A}c' \circledast Ac^{L_{Port}}, I_0) \quad (\{!, ?\} \times Port') \tag{4.13}$$

D'après la proposition 4.2.13 :

$$Exec(\mathcal{A}c' \circledast Ac^{L_{Port}}, I_0) \leq_{si} \mathcal{A} \quad (\{!, ?\} \times Port') \tag{4.14}$$

A partir de 4.13 et 4.14 :

$$Exec(\mathcal{N}, m_0) \leq_{si} \mathcal{A} \quad (\{!, ?\} \times Port') \tag{4.15}$$

Cela conclut la preuve que $\mathcal{PEL}(\leq_{si})$ se réduit à $\mathcal{ERA}(\leq_{si})$ et que $\mathcal{PEL}(\leq_{si})$ est décidable. □

Nous pouvons maintenant énoncer le théorème suivant.

Théorème 4.2.16. *Le problème de la composition $\mathcal{PC}(\leq_{si})$ est décidable.*

Démonstration. Il suffit d'utiliser les lemmes 4.2.5 et 4.2.15 pour prouver que $\mathcal{PC}(\leq_{si})$ est décidable. □

Lemme 4.2.17. *Le problème $\mathcal{PEL}(\subseteq_{tr})$ est décidable.*

Démonstration. Il suffit d'utiliser la réduction du lemme 4.2.15. Sachant que le problème d'équivalence entre un réseau de Petri et un automate fini est décidable [Jan94], on obtient que $\mathcal{PEL}(\subseteq_{tr})$ est également décidable. □

Théorème 4.2.18. *Le problème de la composition $\mathcal{PC}(\subseteq_{tr})$ est décidable.*

Démonstration. Il suffit d'utiliser les lemmes 4.2.5 et 4.2.17 pour prouver que $\mathcal{PC}(\subseteq_{tr})$ est décidable. □

4.2.2 Résultats d'indécidabilité

Dans cette section, nous verrons que, contrairement au problème $\mathcal{PC}(\subseteq_{tr}$) et $\mathcal{PC}(\leq_{si})$, les problèmes $\mathcal{PC}(\equiv_{tr})$ et $\mathcal{PC}(\longleftrightarrow_{bi})$ sont indécidables. Soulignons qu'il n'est pas possible d'utiliser les réductions présentées dans la section 4.2.1, dans le cas où les relations d'équivalence de traces et de bisimulation sont considérées. Plus précisément, le problème $\mathcal{PC}(\equiv_{tr})$ (resp. $\mathcal{PC}(\longleftrightarrow_{bi})$) ne peut se réduire au problème $\mathcal{PEL}(\equiv_{tr})$ (resp. $\mathcal{PEL}(\longleftrightarrow_{bi})$).

Avant d'énoncer les résultats d'indécidabilité des problèmes $\mathcal{PC}(\equiv_{tr})$ et $\mathcal{PC}(\longleftrightarrow_{bi})$, nous allons considérer des problèmes de décision plus simples notés $\mathcal{SPC}(\equiv_{tr})$ et $\mathcal{SPC}(\longleftrightarrow_{bi})$. Dans un premier temps, nous montrons que le problème $\mathcal{SPC}(\equiv_{tr})$ (resp. $\mathcal{SPC}(\longleftrightarrow_{bi})$) se réduit au problème $\mathcal{PC}(\equiv_{tr})$ (resp. $\mathcal{PC}(\longleftrightarrow_{bi})$). Ensuite, dans un second temps, nous montrons que les problèmes $\mathcal{SPC}(\equiv_{tr})$ et $\mathcal{SPC}(\longleftrightarrow_{bi})$ sont indécidables. Par conséquent, nous obtenons l'indécidabilité de $\mathcal{PC}(\equiv_{tr})$ et $\mathcal{PC}(\longleftrightarrow_{bi})$. Ci-dessous, nous décrivons le problème $\mathcal{SPC}(\approx)$.

Problème $\mathcal{SPC}(\approx)$: problème simplifié de la composition

Instance : un ensemble fini d'actions Σ, des ensembles finis de ports $Port$ et $Port' \subseteq Port$, et des automates finis $\mathcal{A}$ et $\mathcal{B}$ sur $\Sigma \cup (\{!,?\} \times Port)$.

Question : existe-t-il un automate fini $\mathcal{M}$ sur $\{!,?\} \times Port$ tel que

$$Exec(\mathcal{A}) \approx Exec(\mathcal{B} \otimes \mathcal{M})(\{!,?\} \times Port') \,?$$

Pour décrire $\mathcal{SPC}(\subseteq_{tr})$, $\mathcal{SPC}(\equiv_{tr})$, $\mathcal{SPC}(\leq_{si})$ et $\mathcal{SPC}(\longleftrightarrow_{bi})$, il suffit de remplacer $\approx$ respectivement par $\subseteq_{tr}$, $\equiv_{tr}$, $\leq_{si}$ et $\longleftrightarrow_{bi}$.

Remarque 4.2.19. Le produit asynchrone entre $\mathcal{B}$ et $\mathcal{M}$ est défini, malgré le fait que $\mathcal{B}$ soit un automate sur $\Sigma \cup (\{!,?\} \times Port)$ et que $\mathcal{M}$ soit un automate sur $\{!,?\} \times Port$. Il suffit de considérer $\mathcal{M} = (Q^{\mathcal{M}}, q_0^{\mathcal{M}}, \rightarrow_{\mathcal{M}})$ comme un automate sur $\Sigma \cup (\{!,?\} \times Port)$ tel que pour toute action $a \in \Sigma$ $(q, a, q') \notin \rightarrow_{\mathcal{M}}$. Pour plus de précision, voir la remarque 3.1.9.

Lemme 4.2.20. *Le problème $\mathcal{SPC}(\approx)$ se réduit en espace logarithmique au problème $\mathcal{PC}(\approx)$, où $\approx \in \{\subseteq_{tr}, \equiv_{tr}, \leq_{si}, \longleftrightarrow_{bi}\}$.*

Démonstration. Considérons une instance du problème $\mathcal{SPC}(\approx)$. Soient Σ un ensemble fini d'actions, $Port$ et $Port' \subseteq Port$ des ensembles finis de ports et $\mathcal{A} = (Q^{\mathcal{A}}, q_0^{\mathcal{A}}, \rightarrow_{\mathcal{A}})$ et $\mathcal{B} = (Q^{\mathcal{B}}, q_0^{\mathcal{B}}, \rightarrow_{\mathcal{B}})$ des automates finis sur $\Sigma \cup (\{!,?\} \times Port)$. L'instance $\rho(\Sigma, Port, Port', A, B)$ de $\mathcal{PC}(\approx)$, que nous construisons est donnée par l'ensemble fini d'actions Σ_e, les ensembles finis de ports $Port_e$ et $Port'_e$, un ensemble fini de formules atomiques At, un ensemble maximal consistant de littéraux I_0 sur At, ainsi que les automates communicants conditionnels finis $\mathcal{A}c^{client}$, $\mathcal{A}c^{but}$, $\mathcal{A}c^1$, ..., $\mathcal{A}c^n$ sur Σ_e, $Port_e$ et At. Plus précisément, cette instance est construite comme suit :

- $\Sigma_e = \Sigma$,
- $Port_e = Port$,
- $Port'_e = Port'$,
- $At = \emptyset$,
- $I_0 = \emptyset$,

– $\mathcal{A}c^{client} = (Q^{client}, q_0^{client}, \delta^{client})$, tel que $\delta^{client} = \emptyset$,
– $\mathcal{A}c^{but} = Auto^{-1}(\mathcal{A})$,
– $n = 1$ et $\mathcal{A}c^1 = Auto^{-1}(\mathcal{B})$.

Nous allons montrer que :

1. ρ est calculable par une machine de Turing déterministe en utilisant un espace logarithmique,
2. il existe un automate $\mathcal{M}$ sur $\{!, ?\} \times Port$ tel que $Exec(\mathcal{A}) \approx Exec(\mathcal{B} \otimes \mathcal{M})(\{!, ?\} \times Port')$ ssi il existe un automate communicant conditionnel $\mathcal{A}c^{med}$ sur Σ_e, $Port_e$ et At, tel que $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, I_0) \approx Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \mathcal{A}c^{med}, I_0)(\{!, ?\} \times Port')$.

Concernant le premier point, il est évident que ρ est calculable par une machine de Turing déterministe qui travaille en espace logarithmique. Car il suffit à la machine de réécrire dans le ruban de sortie ce qu'elle lit dans son ruban d'entrée, sans utiliser les rubans de travail.

Concernant le second point, il est clair que $\delta^{client} = \emptyset$ implique que $\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}$ et $\mathcal{A}c^{but}$ (resp. $\mathcal{A}c^{client} \circledast \mathcal{A}c^1$ et $\mathcal{A}c^1$) sont isomorphes. Il suffit donc de prouver qu'il existe un automate $\mathcal{M}$ sur $\{!, ?\} \times Port$ tel que $Exec(\mathcal{A}) \approx Exec(\mathcal{B} \otimes \mathcal{M})(\{!, ?\} \times Port')$ ssi il existe un automate communicant conditionnel $\mathcal{A}c^{med}$ sur $\Sigma_e \cup (\{, ?\} \times Port_e)$ et At, tel que $Exec(\mathcal{A}c^{but}, I_0) \approx Exec(\mathcal{A}c^1 \circledast \mathcal{A}c^{med}, I_0)(\{!, ?\} \times Port')$.

Concernant l'implication de gauche à droite. Supposons qu'il existe un automate $\mathcal{M}$ sur $\{!, ?\} \times Port$ tel que :

$$Exec(\mathcal{A}) \approx Exec(\mathcal{B} \otimes \mathcal{M})(\{!, ?\} \times Port'). \tag{4.16}$$

Soit l'automate communicant conditionnel $\mathcal{A}c^{med} = Auto^{-1}(\mathcal{M})$ sur Σ_e, $Port_e$ et $At = \emptyset$. D'après la construction de ρ on a :

$$\mathcal{A}c^1 \circledast \mathcal{A}c^{med} = Auto^{-1}(\mathcal{B}) \circledast Auto^{-1}(\mathcal{M}).$$

D'après la proposition 4.2.2 on a :

$$Auto^{-1}(\mathcal{B}) \circledast Auto^{-1}(\mathcal{M}) = Auto^{-1}(\mathcal{B} \otimes \mathcal{M}).$$

Par conséquent :

$$\mathcal{A}c^1 \circledast \mathcal{A}c^{med} = Auto^{-1}(\mathcal{B} \otimes \mathcal{M}).$$

Par construction de ρ, $I_0 = \emptyset$. Ainsi :

$$Exec(\mathcal{A}c^1 \circledast \mathcal{A}c^{med}, I_0) = Exec(\mathcal{A}c^1 \circledast \mathcal{A}c^{med}, \emptyset) = Exec(Auto^{-1}(\mathcal{B} \otimes \mathcal{M}), \emptyset).$$

D'après la proposition 4.2.1 on a :

$$Exec(Auto^{-1}(\mathcal{B} \otimes \mathcal{M}), \emptyset) \text{ et } Exec(\mathcal{B} \otimes \mathcal{M}) \text{ sont isomorphes.}$$

On en déduit que :

$$Exec(\mathcal{A}c^1 \circledast \mathcal{A}c^{med}, I_0) \text{ et } Exec(\mathcal{B} \otimes \mathcal{M}) \text{ sont isomorphes.} \quad (4.17)$$

De la même façon, en utilisant la proposition 4.2.1 on a :

$$Exec(\mathcal{A}c^{but}, \emptyset) \text{ et } Exec(\mathcal{A}) \text{ sont isomorphes.} \quad (4.18)$$

En utilisant les équations 4.16, 4.17, 4.18 et le fait que l'isomorphisme est plus fort que la relation $\approx$ modulo un ensemble d'action, on a :

$$Exec(\mathcal{A}c^{but}, I_0) \approx Exec(\mathcal{A}c^1 \circledast \mathcal{A}c^{med}, I_0)(\{!, ?\} \times Port').$$

Concernant l'implication de droite à gauche. Supposons qu'il existe un automate communicant conditionnel $\mathcal{A}c^{med}$ sur $\Sigma_e \cup (\{!, ?\} \times Port_e)$ et $At = \emptyset$ tel que :

$$Exec(\mathcal{A}c^{but}, I_0) \approx Exec(\mathcal{A}c^1 \circledast \mathcal{A}c^{med}, I_0)(\{!, ?\} \times Port'). \quad (4.19)$$

Soit l'automate $\mathcal{M} = Auto(\mathcal{A}c^{med})$ sur $\Sigma \cup (\{!, ?\} \times Port)$. On sait que $Auto^{-1}(Auto(\mathcal{A}c^{med})) = \mathcal{A}c^{med}$, de la remarque 3.2.7. Donc $Auto^{-1}(\mathcal{M}) = \mathcal{A}c^{med}$. L'équation 4.19 peut être écrite comme suit :

$$Exec(Auto^{-1}(\mathcal{A}), \emptyset) \approx Exec(Auto^{-1}(\mathcal{B}) \circledast Auto^{-1}(\mathcal{M}), \emptyset)(\{!, ?\} \times Port'). \quad (4.20)$$

Comme pour l'implication de gauche à droite, en utilisant les proposition 4.2.1 et 4.2.2, on obtient que :

$$Exec(\mathcal{A}) \approx Exec(\mathcal{B} \circledast \mathcal{M})(\{!, ?\} \times Port'). \quad (4.21)$$

□

Définition 4.2.21 (Automate associé à un réseau de Petri). *Soit $\mathcal{N} = (P, T, F, l)$ un réseau de Petri ordinaire sur Σ et m_0 un marquage initial pour $\mathcal{N}$ avec $m_0(p) = 0$, pour tout $p \in P$. Considérons une transition $t \in T$ telle que ${}^\bullet t = \{p_1, \ldots, p_r\}$ et $t^\bullet = \{p'_1, \ldots, p'_s\}$. Soit $Q^t_{in} = \{q_1^{{}^\bullet t}, \ldots, q_{r-1}^{{}^\bullet t}\}$, $Q^t_{out} = \{q_1^{t^\bullet}, \ldots, q_{s-1}^{t^\bullet}\}$ et $AutoR(\mathcal{N}) = (Q, q_0, \rightarrow)$ un automate fini sur $\Sigma \cup (\{!, ?\} \times P)$ construit comme suit :*

- *$Q = \{q_0\} \cup \bigcup_{t \in T}(\{q_1^t, q_2^t\} \cup Q^{{}^\bullet t} \cup Q^{t^\bullet})$ et*
- *$\rightarrow \subseteq Q \times (\Sigma \cup (\{!, ?\} \times P)) \times Q$ est la relation de transition définie par : pour tout $t \in T$ tel que $l(t) = a$ on a :*
  - *$q_0 \rightarrow_{\mathcal{A}}^{?p_1} q_1^{{}^\bullet t}$,*

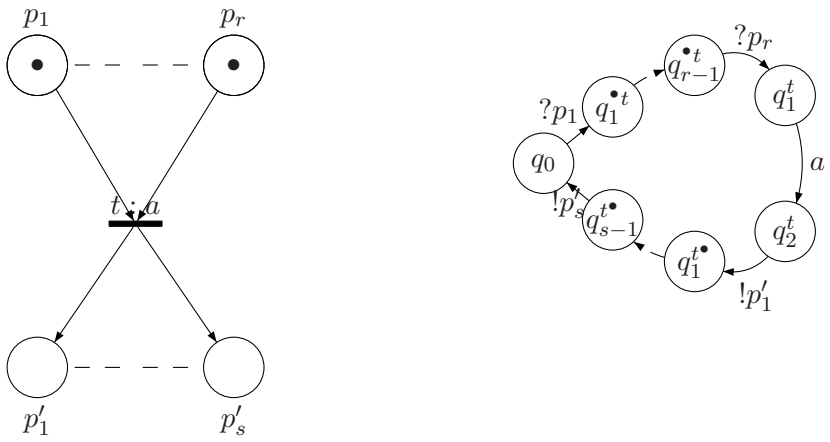

FIG. 4.17 – A gauche une transition t d'un réseau de Petri $\mathcal{N}$ et à droite sa transformation en transitions dans l' automate fini $AutoR(\mathcal{N})$.

- $q_{j-1}^{\bullet t} \rightarrow_{\mathcal{A}}^{?p_j} q_j^{\bullet t}$, $j \in \{2, \ldots, r-1\}$,
- $q_{r-1}^{\bullet t} \rightarrow_{\mathcal{A}}^{?p_r} q_1^t$,
- $q_1^t \rightarrow_{\mathcal{A}}^{a} q_2^t$,
- $q_2^t \rightarrow_{\mathcal{A}}^{!p'_1} q_1^{t\bullet}$,
- $q_{j-1}^{t\bullet} \rightarrow_{\mathcal{A}}^{!p'_j} q_j^{t\bullet}$, $j \in \{2, \ldots, s-1\}$ *et*
- $q_{s-1}^{t\bullet} \rightarrow_{\mathcal{A}}^{!p'_s} q_0$.

La Figure 4.17 montre que le tir de la transition a vide les places $p_1, \ldots, p_r$ qui sont en entrée de la transition et remplit les places $p'_1, \ldots, p'_s$ qui sont en sortie de cette transition. Cette transition est reproduite par l'automate $AutoR(\mathcal{N})$ par la séquence d'actions suivantes : lecture sur les ports $p_1, \ldots, p_r$, correspondant aux places en entrée de la transition, ce qui a pour effet de vider ces ports, puis la transition a, suivi de l'écriture sur les ports $p'_1, \ldots, p'_s$, correspondant aux places en sortie de la transition, ce qui a pour effet de remplir ces ports. Ainsi le contenu des ports reflète correctement le contenu des places du réseaux. L'exécution de l'automate ainsi obtenu est isomorphe modulo $(\{!, ?\} \times P)$ au graphe d'exécution du réseau de Petri.

Lemme 4.2.22. *Soit $\mathcal{N} = (P, T, F, l)$ un réseau de Petri ordinaire sur Σ et m_0 un marquage initial pour $\mathcal{N}$ avec $m_0(p) = 0$, pour tout $p \in P$.*

$Exec(AutoR(\mathcal{N})) \simeq_{iso} Exec(\mathcal{N}, m_0) \quad (\{!, ?\} \times P)$.

Démonstration. Pour que $Exec(AutoR(\mathcal{N}))$ soit isomorphe [5] à $Exec(\mathcal{N}, m_0)$ modulo $\{!, ?\} \times P$, il suffit que $Obs_{(\{!,?\}\times P)}(Exec(AutoR(\mathcal{N})))$ soit isomorphe à $Obs_{(\{!,?\}\times P)}(Exec(\mathcal{N}, m_0))$. Sachant que $Exec(\mathcal{N}, m_0)$ est défini sur Σ et que $\Sigma \cap (\{!, ?\} \times P) = \emptyset$ alors $Obs_{(\{!,?\}\times P)}(Exec(\mathcal{N}, m_0))$=$Exec(\mathcal{N}, m_0)$.

Nous utilisons dans cette preuve l'automate $\mathcal{A} = (Q, q_0, \rightarrow)$ pour représenter $AutoR(\mathcal{N})$ et l'automate $\mathcal{A}'$ pour représenter $Obs_{(\{!,?\}\times P)}(Exec(\mathcal{A}))$. Considérons $Q' = \{R(q, \gamma) \mid (q, \gamma) \in Q \times \Gamma\}$ l'ensemble des états de $\mathcal{A}'$. Rappelons que Γ est l'ensemble des fonctions $\gamma : P \rightarrow \mathbb{N}$. La construction de $\mathcal{A}$ est telle que, pour tout état $q \in Q$, $q_0 \rightarrow_{\mathcal{A}}^{\{!,?\}\times P^\star} q$ ou $q \rightarrow_{\mathcal{A}}^{\{!,?\}\times P^\star} q_0$. Ainsi, $Q' = \{R(q_0, \gamma) \mid \gamma \in \Gamma\}$. Soit la fonction $g : M \rightarrow Q'$ telle que $g(m) = R(q_0, m)$. Sachant que $m_0(p) = 0$, pour tout $p \in P$, Il est clair que $g(m_0) = R(q_0, m_0)$. De plus, g est une bijection. Montrons que pour tout marquage $m_1, m_2 \in M$ et pour toute action $a \in \Sigma$:

$$\begin{array}{c} m_1[a\rangle m_2 \\ \Longleftrightarrow \\ R(q_0, m_1) \rightarrow_{\mathcal{A}'}^{\{a\}} R(q_0, m_2). \end{array}$$

($\Rightarrow$) Considérons l'implication de gauche à droite. Si $m_1[a\rangle m_2$ alors il existe une transition t tels que $l(t) = a$, pour tout $p \in {}^\bullet t$, $m_1(p) > 0$ et :

$$m_2(p) = \begin{cases} m_1(p) + 1 & \text{si } p \notin^\bullet t \text{ et } p \in t^\bullet \\ m_1(p) - 1 & \text{si } p \in^\bullet t \text{ et } p \notin t^\bullet \\ m_1(p) & \text{sinon} \end{cases}$$

Ainsi, par construction de $\mathcal{A}$, si ${}^\bullet t = \{p_1, \ldots, p_r\}$ et $t^\bullet = \{p'_1, \ldots, p'_s\}$ alors :

- $q_0 \rightarrow_{\mathcal{A}}^{?p_1} q_1^{{}^\bullet t} \ldots q_{r-1}^{{}^\bullet t} \rightarrow_{\mathcal{A}}^{?p_r} q_1^t$,
- $q_1^t \rightarrow_{\mathcal{A}}^{a} q_2^t$ et
- $q_2^t \rightarrow_{\mathcal{A}}^{!p'_1} q_1^{t^\bullet} \ldots q_{s-1}^{t^\bullet} \rightarrow_{\mathcal{A}}^{!p'_s} q_0$.

Sachant que pour tout $p \in {}^\bullet t$, $m_1(p) > 0$ alors il existe $\gamma_1, \ldots, \gamma_r$, $\gamma'_1, \ldots, \gamma'_{s-1} \in \Gamma$ telles que :

- $(q_0, m_1) \rightarrow_{Exec(\mathcal{A})}^{\{?p_1\}} (q_1^{{}^\bullet t}, \gamma_1) \ldots (q_{r-1}^{{}^\bullet t}, \gamma_{r-1}) \rightarrow_{Exec(\mathcal{A})}^{\{?p_r\}} (q_1^t, \gamma_r)$,
- $(q_1^t, \gamma_r) \rightarrow_{Exec(\mathcal{A})}^{\{a\}} (q_2^t, \gamma_r)$ et
- $(q_2^t, \gamma_r) \rightarrow_{Exec(\mathcal{A})}^{\{!p'_1\}} (q_1^{t^\bullet}, \gamma'_1) \ldots (q_{s-1}^{t^\bullet}, \gamma'_{s-1}) \rightarrow_{Exec(\mathcal{A})}^{\{!p'_s\}} (q_0, m_2)$.

Par conséquent, d'après la construction de $\mathcal{A}'$, $R(q_0, m_1) \rightarrow_{\mathcal{A}'}^{\{a\}} R(q_0, m_2)$.
($\Leftarrow$) Considérons l'implication de droite à gauche. Supposons que :

$$R(q_0, m_1) \rightarrow_{\mathcal{A}'}^{\{a\}} R(q_0, m_2).$$

[5] Pour plus de détails concernant la définition de l'isomorphisme modulo un ensemble d'actions, voir la définition 3.1.24.

Par construction de $\mathcal{A}'$, il existe (q_3, γ_3), $(q_4, \gamma_3) \in Q \times \Gamma$ tels que :
- $(q_0, m_1) \rightarrow^{\{!,?\}\times P^\star}_{Exec(\mathcal{A})} (q_3, \gamma_3)$,
- $(q_3, \gamma_3) \rightarrow^{\{a\}}_{Exec(\mathcal{A})} (q_4, \gamma_3)$ et
- $(q_4, \gamma_3) \rightarrow^{\{!,?\}\times P^\star}_{Exec(\mathcal{A})} (q_0, m_2)$.

Par construction de $\mathcal{A}$, cela implique qu'il existe $\gamma_1, \ldots, \gamma_r$, $\gamma'_1, \ldots, \gamma'_{s-1} \in \Gamma$ et une transition $t \in T$ telles que $q_3 = q_1^t$, $q_4 = q_2^t$, ${}^\bullet t = \{p_1, \ldots, p_r\}$, $t^\bullet = \{p'_1, \ldots, p'_s\}$, $l(t) = a$ et :
- $(q_0, m_1) \rightarrow^{\{?p_1\}}_{Exec(\mathcal{A})} (q_1^{{}^\bullet t}, \gamma_1) \ldots (q_{r-1}^{{}^\bullet t}, \gamma_{r-1}) \rightarrow^{\{?p_r\}}_{Exec(\mathcal{A})} (q_1^t, \gamma_r)$,
- $(q_1^t, \gamma_r) \rightarrow^{\{a\}}_{Exec(\mathcal{A})} (q_2^t, \gamma_r)$ et
- $(q_2^t, \gamma_r) \rightarrow^{\{!p'_1\}}_{Exec(\mathcal{A})} (q_1^{t^\bullet}, \gamma'_1) \ldots (q_{s-1}^{t^\bullet}, \gamma'_{s-1}) \rightarrow^{\{!p'_s\}}_{Exec(\mathcal{A})} (q_0, m_2)$.

Il s'ensuit, par construction de $Exec(\mathcal{A})$, que $m_1(p) > 0$, pour tout $p \in {}^\bullet t$ et :

$$m_2(p) = \begin{cases} m_1(p) + 1 & \text{si } p \notin^\bullet t \text{ et } p \in t^\bullet \\ m_1(p) - 1 & \text{si } p \in^\bullet t \text{ et } p \notin t^\bullet \\ m_1(p) & \text{sinon} \end{cases}$$

Par conséquent, $m_1[a\rangle m_2$.

□

Lemme 4.2.23. *Le problème $\mathcal{SPC}(\equiv_{tr})$ est indécidable.*

Démonstration. Cette preuve se base sur la réduction du problème de l'équivalence de trace entre réseaux de Petri.

Considérons le problème de décision suivant :

Problème $\mathcal{RP}(\equiv_{tr})$: problème de l'équivalence de traces entre réseaux de Petri

Instance : soient Σ un ensemble fini d'actions, des réseaux de Petri $\mathcal{N} = (P^{\mathcal{N}}, T^{\mathcal{N}}, F^{\mathcal{N}}, l^{\mathcal{N}})$ et $\mathcal{O} = (P^{\mathcal{O}}, T^{\mathcal{O}}, F^{\mathcal{O}}, l^{\mathcal{O}})$ sur Σ et des marquages initiaux de places $m_0^{\mathcal{N}}$ et $m_0^{\mathcal{O}}$ respectivement pour $\mathcal{N}$ et $\mathcal{O}$.

Question : est-il vrai que $Exec(\mathcal{N}, m_0^{\mathcal{N}}) \equiv_{tr} Exec(\mathcal{O}, m_0^{\mathcal{O}})$ $(\emptyset)$?

Sachant que $\mathcal{RP}(\equiv_{tr})$ est indécidable [Jan95], il suffit de réduire $\mathcal{RP}(\equiv_{tr})$ à $\mathcal{SPC}(\equiv_{tr})$ afin de montrer que $\mathcal{SPC}(\equiv_{tr})$ est indécidable. Etant donnés un ensemble fini d'actions Σ et des réseaux de Petri $\mathcal{N} = (P_{\mathcal{N}}, T_{\mathcal{N}}, F_{\mathcal{N}}, l_{\mathcal{N}})$, $\mathcal{O} = (P_{\mathcal{O}}, T_{\mathcal{O}}, F_{\mathcal{O}}, l_{\mathcal{O}})$ définis sur Σ et des maquages initiaux $m_0^{\mathcal{N}}$ et $m_0^{\mathcal{O}}$ respectivement pour $\mathcal{N}$ et $\mathcal{O}$, nous devons déterminer si $Exec(\mathcal{N}, m_0^{\mathcal{N}}) \equiv_{tr} Exec(\mathcal{O}, m^{\mathcal{O}})$ $(\emptyset)$. Il est significatif de constater que le problème $\mathcal{RP}(\equiv_{tr})$ reste indécidable lorsque toutes les places des réseaux sont initialement vides, car $\mathcal{RP}(\equiv_{tr})$ peut être réduit à ce cas particulier. Dans le reste de cette

preuve, nous allons considérer que $m_0^{\mathcal{N}}(p) = 0$, $p \in P^{\mathcal{N}}$ et $m_0^{\mathcal{O}}(p) = 0$, $p \in P^{\mathcal{O}}$. Maintenant, nous pouvons décrire la réduction du problème $\mathcal{RP}(\equiv_{tr})$ au problème $\mathcal{SPC}(\equiv_{tr})$.

L'instance $\rho(\Sigma, \mathcal{N}, \mathcal{O}, m_0^{\mathcal{N}}, m_0^{\mathcal{O}})$ de $\mathcal{SPC}(\equiv_{tr})$ que nous construisons est donnée par l'ensemble fini d'actions Σ^e, les ensembles finis de ports $Port$ et $Port'$, et les automates finis $\mathcal{A} = (Q^{\mathcal{A}}, q_0^{\mathcal{A}}, \rightarrow_{\mathcal{A}})$ et $\mathcal{B} = (Q^{\mathcal{B}}, q_0^{\mathcal{B}}, \rightarrow_{\mathcal{B}})$ définis sur $(\{!, ?\} \times Port) \cup \Sigma^e$ tels que :

- $\Sigma^e = \Sigma \cup \{a_1^e, a_2^e, a_3^e, a_4^e\}$ où a_1^e, a_2^e, a_3^e et a_4^e sont de nouvelles actions,
- $Port = Port' = P_{\mathcal{N}} \cup P_{\mathcal{O}}$,
- $\mathcal{A}$ est l'automate fini de la Figure 4.18 et
- $\mathcal{B}$ est l'automate fini de la Figure 4.19.

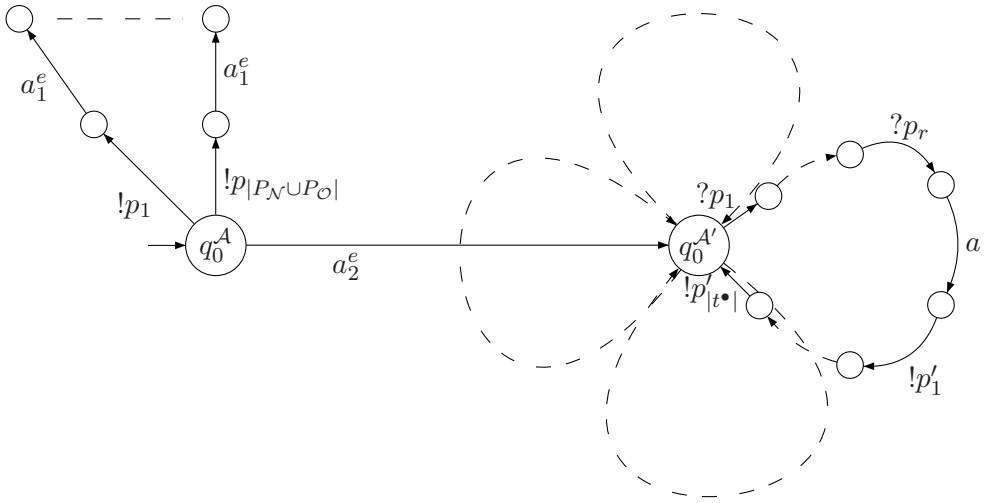

FIG. 4.18 – L'automate $\mathcal{A}$

Les parties en forme de fleur dans les automates $\mathcal{A}$ et $\mathcal{B}$ sont respectivement $AutoR(\mathcal{N})$ et $AutoR(\mathcal{O})$. Les séquences de transitions des automates $\mathcal{A}$ et $\mathcal{B}$ contenant les actions a_1^e, a_2^e, a_3^e ou a_4^e 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 est calculable par une machine de Turing déterministe qui termine toujours ses calculs.

Dans ce qui suit, nous montrons que :

$$Exec(\mathcal{N}, m_0^{\mathcal{N}}) \equiv_{tr} Exec(\mathcal{O}, m_0^{\mathcal{O}})\ (\emptyset) \text{ ssi il existe un automate } \mathcal{M} = (Q^{\mathcal{M}}, q_0^{\mathcal{M}}, \rightarrow_{\mathcal{M}},) \text{ défini sur } \{!, ?\} \times Port \text{ tel que } Exec(\mathcal{A}) \equiv_{tr} Exec(\mathcal{B} \otimes \mathcal{M})\ (\{!, ?\} \times Port).$$

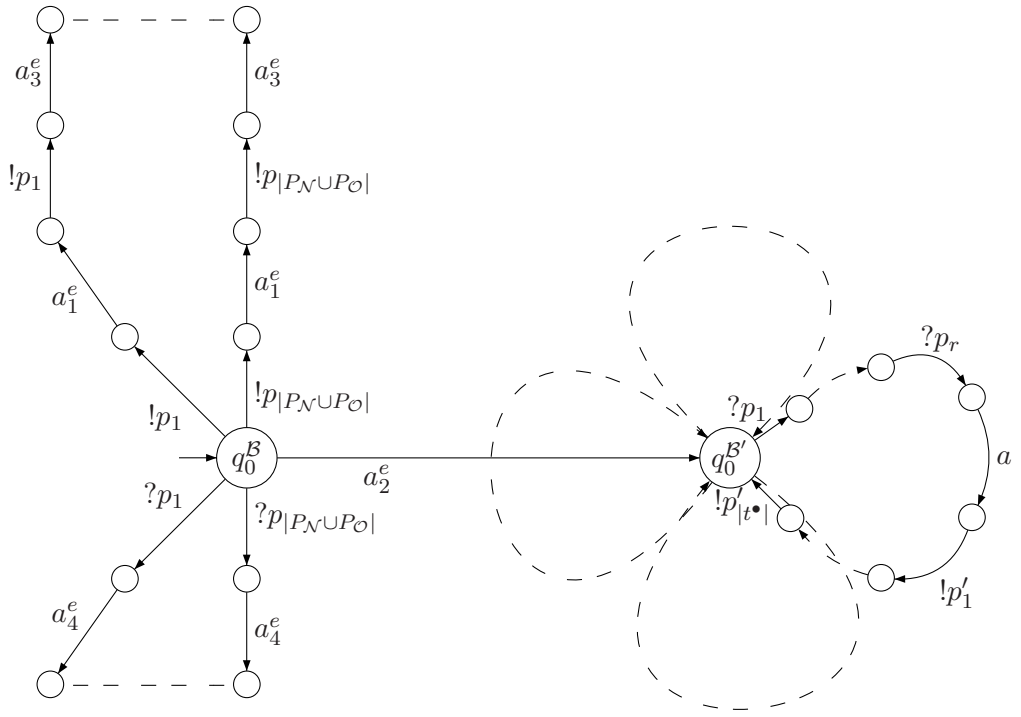

FIG. 4.19 – L'automate $\mathcal{B}$

Notons par $\mathcal{A}'$ l'automate $AutoR(\mathcal{N})$ et par $\mathcal{B}'$ l'automate $AutoR(\mathcal{O})$. D'après le lemme 4.2.22, on a $Exec(\mathcal{A}') \simeq_{iso} Exec(\mathcal{N}, m_0^\mathcal{N}) \quad (\{!,?\} \times P^\mathcal{N})$ et $Exec(\mathcal{B}') \simeq_{iso} Exec(\mathcal{O}, m_0^\mathcal{O}) \quad (\{!,?\} \times P^\mathcal{O})$.

Concernant l'implication de gauche à droite, supposons que :

$$Exec(\mathcal{N}, m_0^\mathcal{N}) \equiv_{tr} Exec(\mathcal{O}, m_0^\mathcal{O})(\emptyset).$$

Sachant que $Exec(\mathcal{N}, m_0^\mathcal{N}) \simeq_{iso} Exec(\mathcal{A}') \ (\{!,?\} \times Port)$ et que $Exec(\mathcal{O}, m_0^\mathcal{O}) \simeq_{iso} Exec(\mathcal{B}')) \ (\{!,?\} \times Port)$ alors :

$$Exec(\mathcal{A}') \equiv_{tr} Exec(\mathcal{B}') \ (\{!,?\} \times Port).$$

Il est facile de vérifier que, $Exec(\mathcal{A}) \equiv_{tr} Exec(\mathcal{B}) \ (\{!,?\} \times Port)$. Ainsi pour le médiateur $\mathcal{M} = (\{q_0^\mathcal{M}\}, q_0^\mathcal{M}, \emptyset)$ on a :

$$Exec(\mathcal{A}) \equiv_{tr} Exec(\mathcal{B} \otimes \mathcal{M}) \ (\{!,?\} \times Port).$$

Concernant l'implication de droite à gauche, supposons que :

$$Exec(\mathcal{N}, m_0^{\mathcal{N}}) \not\equiv_{tr} Exec(\mathcal{O}, m_0^{\mathcal{O}})(\emptyset).$$

Soit $\mathcal{M} = (Q^{\mathcal{M}}, q_0^{\mathcal{M}}, \rightarrow_{\mathcal{M}})$, un médiateur sur $\{!, ?\} \times Port$. Trois cas sont possibles :

(1) il n'existe aucune transition à partir de $q_0^{\mathcal{M}}$,
(2) il existe $\pi \in Port$ et $q^{\mathcal{M}} \in Q^{\mathcal{M}}$ tels que $(q_0^{\mathcal{M}}, !\pi, q^{\mathcal{M}}) \in \rightarrow_{\mathcal{M}}$ ou
(3) il existe $\pi \in Port$ et $q^{\mathcal{M}} \in Q^{\mathcal{M}}$ tels que $(q_0^{\mathcal{M}}, ?\pi, q^{\mathcal{M}}) \in \rightarrow_{\mathcal{M}}$.

Dans le cas (1), il est clair que $Exec(\mathcal{B} \otimes \mathcal{M})$ est isomorphe à $Exec(\mathcal{B})$. On sait que $Exec(\mathcal{N}, m_0^{\mathcal{N}}) \not\equiv_{tr} Exec(\mathcal{O}, m_0^{\mathcal{O}})(\emptyset)$ implique :

$$Exec(\mathcal{A}') \not\equiv_{tr} Exec(\mathcal{B}') \ (\{!, ?\} \times Port).$$

Ainsi, $Exec(\mathcal{A}) \not\equiv_{tr} Exec(\mathcal{B}) \ (\{!, ?\} \times Port)$ et donc :

$$Exec(\mathcal{A}) \not\equiv_{tr} Exec(\mathcal{B} \otimes \mathcal{M}) \ (\{!, ?\} \times Port).$$

Dans le cas (2), on aura une transition dans $Exec(\mathcal{B} \otimes \mathcal{M})$ étiquetée par a_4^e. Dans $Exec(\mathcal{A}')$ il n'existe pas de transition étiquetée par a_4^e. De ce fait :

$$Exec(\mathcal{A}) \not\equiv_{tr} Exec(\mathcal{B} \otimes \mathcal{M})(\{!, ?\} \times Port).$$

Dans le cas (3), on aura dans $Exec(\mathcal{B} \otimes \mathcal{M})$ une transition étiquetée par a_3^e. Dans $Exec(\mathcal{A})$ il n'existe pas de transition étiquetée par a_3^e. De ce fait :

$$Exec(\mathcal{A}) \not\equiv_{tr} Exec(\mathcal{B} \otimes \mathcal{M})(\{!, ?\} \times Port).$$

Par conséquent, il n'existe pas d'automate fini $\mathcal{M}$ défini sur $\{!, ?\} \times Port$ tel que $Exec(\mathcal{A}) \equiv_{tr} Exec(\mathcal{B} \otimes \mathcal{M}) \ (\{!, ?\} \times Port)$. Cela conclut la preuve. □

Théorème 4.2.24. *Le problème de la composition $\mathcal{PC}(\equiv_{tr})$ est indécidable.*

Démonstration. Il suffit d'utiliser les lemmes 4.2.20 et 4.2.23 pour prouver que $\mathcal{PC}(\equiv_{tr})$ est indécidable. □

Lemme 4.2.25. *Le problème $\mathcal{SPC}(\longleftrightarrow_{bi})$ est indécidable.*

Démonstration. Il suffit d'utiliser la réduction du lemme 4.2.23. Sachant que le problème de la bisimulation entre réseaux de Petri est indécidable [Jan95], on obtient que $\mathcal{SPC}(\longleftrightarrow_{bi})$ est également indécidable. □

Théorème 4.2.26. *Le problème de la composition $\mathcal{PC}(\longleftrightarrow_{bi})$ est indécidable.*

Démonstration. Il suffit d'utiliser les lemmes 4.2.20 et 4.2.25 pour prouver que $\mathcal{PC}(\longleftrightarrow_{bi})$ est indécidable. □

4.3 Conclusion

Dans ce chapitre, nous avons présenté notre modèle des services et nous l'avons comparé aux modèles existants. Nous avons également prouvé que les problèmes $\mathcal{PC}(\equiv_{tr})$ et $\mathcal{PC}(\subseteq_{tr})$ sont indécidables (voir le Tableau 4.1). Quant aux problèmes $\mathcal{PC}(\subseteq_{tr})$ et $\mathcal{PC}(\leq_{si})$ ils sont décidables, mais déterminer leur complexité reste un problème ouvert. Dans les chapitres suivants des restrictions seront considérées afin de simplifier les quatre problèmes de la composition et réduire leur complexité. Ces restrictions concernent par exemple : la taille des ports de communication, le type de communication considéré : aucune, synchrone ou asynchrone.

Problème	Décidable	Complexité
$\mathcal{PC}(\subseteq_{tr})$	*oui*	*?*
$\mathcal{PC}(\equiv_{tr})$	*non*	$\times$
$\mathcal{PC}(\leq_{si})$	*oui*	?
$\mathcal{PC}(\longleftrightarrow_{bi})$	*non*	$\times$

Tab. 4.1 – Tableau récapitulatif.

Chapitre 5

Composition de services dans un environnement sans communication

Dans le Chapitre 4, nous avons présenté notre modèle pour représenter les services Web. Nous y avons également défini le problème de la composition. Nous avons prouvé que le problème de la composition, lorsque les relations de bisimulation et d'équivalence de traces sont considérées, est indécidable. Cependant, nous avons vu que le problème devient décidable pour les relations de simulation et d'inclusion de traces. Dans ce chapitre, nous considérons des restrictions sur les services afin de réduire la complexité du problème de la composition. Plus précisément, nous allons considérer des services qui ne peuvent pas effectuer de communication [CGM06, BFDP08, BCF07]. Dans ce modèle, les actions de communication seront représentée par des actions d'un ensemble fini Σ. On ne distinguera plus les actions d'envoi de messages, de réception de messages et les actions internes. Ainsi, nous obtenons un modèle plus abstrait que celui du Chapitre 4. Ensuite, nous décrivons le problème de la composition pour ce modèle. Enfin, nous donnons des résultats concernant sa complexité.

5.1 Modèle des services

Dans ce chapitre, les services peuvent effectuer des actions sous certaines conditions. Une fois les actions effectuées, des effets doivent être pris en considération. Ces effets représentent des modifications sur les variables des services. Pour cela, nous utilisons un ensemble fini d'action Σ et un en-

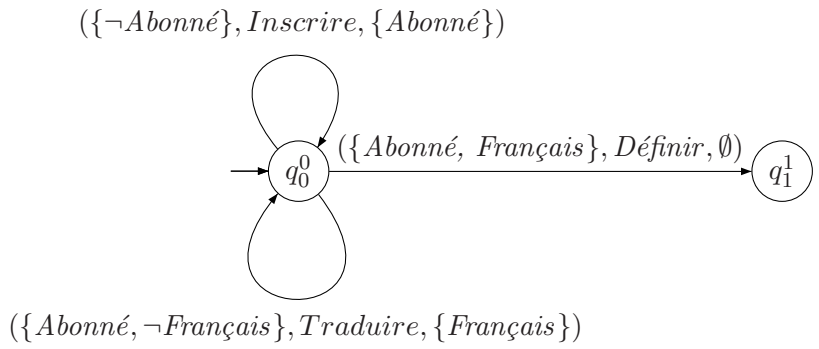

FIG. 5.1 – Service TradEtDef

semble fini de formules atomiques At. Comme les services ne peuvent pas communiquer, nous utiliserons le terme automate conditionnel (AC) au lieu d'automate communicant conditionnel (ACC).

Définition 5.1.1 (Service). Un *service* est un automate conditionnel $\mathcal{A}c = (Q, q_0, \delta)$ sur Σ et At tel que :
- Q est un ensemble fini d'états,
- q_0 est l'état initial,
- $\delta \subseteq 2^{Li(At)} \times Q \times \Sigma \times Q \times 2^{Li(At)}$ est une relation de transition.

Soulignons que tous les états sont finaux, pour les mêmes raisons qu'au chapitre précédent, à savoir rendre les preuves plus simples.

Exemple 5.1.2. *Considérons le service TradEtDef représenté dans la Figure 5.1. Ce service permet d'obtenir la définition d'un mot de la langue française. Dans le cas où on veut obtenir la définition d'un mot qui est dans une autre langue, le service peut le traduire et ensuite le définir. Pour utiliser le service il faut s'inscrire préalablement. Dans cet exemple,* $\Sigma = \{Inscrire, Traduire, Définir\}$ *et* $At = \{Abonné, Français\}$*. Pour exécuter l'action Définir, il faut être abonné au service et il faut que le mot à traduire appartienne à la langue française. Dans le cas où l'utilisateur du service n'est pas abonné, il faut qu'il effectue l'action* $Inscrire$*. Dans le cas où le mot n'appartient pas à la langue française, il faut effectuer l'action* $Traduire$*.*

Nous allons représenter l'aspect dynamique des automates communicants en leur associant des automates, comme nous l'avons fait pour les automates

communicants conditionnels (voir la définition 7.1.4). Nous considérons l'ensemble maximal consistant de littéraux $I_0 \subseteq Li(At)$, pour représenter l'initialisation des formules atomiques.

Définition 5.1.3 (Exécution des AC). Soit $\mathcal{A}c = (Q, q_0, \delta)$ un automate conditionnel sur Σ et At. Nous appelons *exécution* de $\mathcal{A}c$, l'automate $Exec(\mathcal{A}c, I_0) = (Q', q'_0, \rightarrow')$ défini sur Σ tel que :

- $Q' = \{(q, I) \mid q \in Q$ et $I \subseteq Li(At)$ est une partie maximale consistante$\}$,
- $q'_0 = (q_0, I_0)$ et
- $\rightarrow' \subseteq Q' \times \Sigma \times Q'$ est la relation de transition définie par $((q, I), a, (q', I')) \in \rightarrow'$ ssi il existe une transition $(J, q, a, q', J') \in \delta$ tel que $J \subseteq I$ et $I' = (I \setminus \neg J') \cup J'$ avec $\neg J' = \{p : \neg p \in J'\} \cup \{\neg p : p \in J'\}$.

5.2 Composition des services

Dans ce modèle simplifié de services, où les actions de communications n'existent pas, nous n'utilisons ni la notion de service médiateur, ni celle de service client. Le problème de la composition consistera à déterminer si un service but est équivalent à une communauté de services disponibles. Les équivalences qui nous intéressent sont : l'inclusion de traces, l'équivalence de traces, la simulation et la bisimulation.

Problème $\mathcal{PCSC}(\approx)$: problème de la composition dans un environnement sans communication.

Instance : les ensembles finis Σ, At et I_0 et des automates communicants conditionnels finis $\mathcal{A}c^{but}$, $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$ sur Σ et At.

Question : est il vrai que

$$Exec(\mathcal{A}c^{but}, I_0) \approx Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\emptyset), \text{ où } \approx \in \{\subseteq_{tr}, \equiv_{tr}, \leq_{si}, \longleftrightarrow_{bi}\} ?$$

Exemple 5.2.1. *Nous utilisons les mêmes ensembles At et I_0 que ceux de l'exemple 5.1.2. Quant à l'ensemble Σ en lui ajoute l'action $Transmettre$. Nous considérons comme service but le service représenté dans la Figure 5.1 et comme services de la communauté, les services $\mathcal{A}c^1$, $\mathcal{A}c^2$ et $\mathcal{A}c^3$ de la Figure 5.2. Le service $\mathcal{A}c^1$ permet à un utilisateur de s'inscrire à condition qu'il ne le soit pas déjà. Le service $\mathcal{A}c^2$ permet d'obtenir la traduction en français d'un mot dans une autre langue. Le service $\mathcal{A}c^3$ permet d'obtenir la*

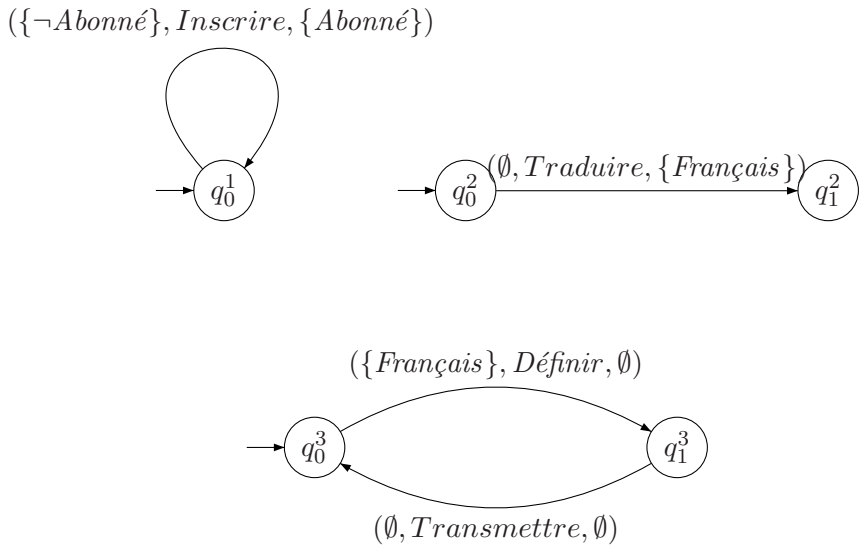

FIG. 5.2 – En haut les services $\mathcal{A}c^1$ et $\mathcal{A}c^2$ et en bas le service $\mathcal{A}c^3$.

définition des mots qui appartiennent à la langue française et de transmettre le résultat. Dans cet exemple, le service but est simulé par la communauté [1]. *Également, la trace du service but est incluse dans celle des services de la communauté. Cependant, le service but n'est ni bisimilaire ni équivalent pour la trace à la communauté. En effet, le service $\mathcal{A}c^3$ peut effectuer l'action $Transmettre$, ce qui n'est pas le cas du service but.*

5.3 Résultats de complexité

Dans cette section, nous donnons des résultats concernant la complexité du problème de la composition de services, lorsque l'environnement est sans communication. Nous montrons que les problèmes $\mathcal{PCSC}(\subseteq_{tr})$ et

[1] Formellement, la communauté est représentée par le produit asynchrone de $\mathcal{A}c^1$, $\mathcal{A}c^2$ et $\mathcal{A}c^3$.

$\mathcal{PCSC}(\equiv_{tr})$ sont EXPSPACE-complet. Nous montrons également que les problèmes $\mathcal{PCSC}(\leq_{si})$ et $\mathcal{PCSC}(\longleftrightarrow_{bi})$ sont EXPTIME-complet. Dans les travaux [BFDP08], les auteurs considèrent les services comme des automates finis dans un environement sans communication. Quant-à la relation choisie pour comparer le but et les services disponibles, ils utilisent la simulation. Ils obtiennent le même résultat que celui que nous présentons dans cette section.

5.3.1 Bornes inférieures de complexité

Nous analysons maintenant la difficulté intrinsèque qu'il y a à comparer des automates conditionnels ou des produits d'automates conditionnels. Dans cette section, nous caractérisons les bornes inférieures de complexité. Nous montrons que les problèmes $\mathcal{PCSC}(\subseteq_{tr})$, $\mathcal{PCSC}(\equiv_{tr})$, $\mathcal{PCSC}(\leq_{si})$ et $\mathcal{PCSC}(\longleftrightarrow_{bi})$ sont respectivement EXPSPACE-difficile, EXPSPACE-difficile, EXPTIME-difficile et EXPTIME-difficile. Pour cela, nous allons réduire le problème de l'équivalence de traces (resp. inclusion de traces) des réseaux de Petri saufs, qui est EXPSPACE-difficile [JM96], vers le problème $\mathcal{PCSC}(\equiv_{tr})$ (resp. $\mathcal{PCSC}(\subseteq_{tr})$) et nous allons réduire le problème de la bisimulation entre des réseaux de Petri saufs, qui est EXPTIME-difficile [JM96], vers le problème $\mathcal{PCSC}(\longleftrightarrow_{bi})$. En ce qui concerne le cas de la simulation, nous allons réduire le problème de la simulation entre un automate et un produit asynchrone d'automates, qui est EXPTIME-difficile [MW08], vers le problème $\mathcal{PCSC}(\leq_{si})$.

Théorème 5.3.1. *Le problème de la composition $\mathcal{PCSC}(\equiv_{tr})$ est EXPSPACE-difficile.*

Démonstration. Pour montrer que le problème $\mathcal{PCSC}(\equiv_{tr})$ est EXPSPACE-difficile, nous allons réduire le problème de la simulation entre deux réseaux de Petri saufs :

Instance : un ensemble fini d'actions Σ, les réseaux de Petri saufs $\mathcal{N}$ et $\mathcal{O}$ sur Σ et des marquages initiaux de places $m_0^{\mathcal{N}}$ et $m_0^{\mathcal{O}}$ respectivement pour $\mathcal{N}$ et $\mathcal{O}$.

Question : est-il vrai que $Exec(\mathcal{N}, m_0^{\mathcal{N}}) \equiv_{tr} Exec(\mathcal{O}, m_0^{\mathcal{O}})$?

qui est EXPSPACE-difficile [JM96], vers le problème $\mathcal{PCSC}(\equiv_{tr})$. Considérons une instance du problème ci-dessus. Soit Σ un ensemble fini d'actions, $\mathcal{N} = (P^{\mathcal{N}}, T^{\mathcal{N}}, F^{\mathcal{N}}, l^{\mathcal{N}})$, $\mathcal{O} = (P^{\mathcal{O}}, T^{\mathcal{O}}, F^{\mathcal{O}}, l^{\mathcal{O}})$ sur Σ et des marquages initiaux de places $m_0^{\mathcal{N}}$ et $m_0^{\mathcal{O}}$ respectivement pour $\mathcal{N}$ et $\mathcal{O}$. L'instance $\rho(\Sigma, \mathcal{N}, \mathcal{O}, m_0^{\mathcal{N}}, m_0^{\mathcal{O}})$ de $\mathcal{PCSC}(\equiv_{tr})$, que nous construisons est donnée par

le nombre entier $n = 1$, un ensemble fini d'actions $\Sigma' = \Sigma$, un ensemble fini de formules atomiques $At = P^{\mathcal{N}} \cup P^{\mathcal{O}}$, un ensemble maximal consistant $I_0 \in 2^{Li(At)}$ et des automates conditionnels à un seul état $\mathcal{A}c^{but}$ et $\mathcal{A}c^1$ sur Σ et At. Dans la suite, nous donnons le détail de la construction de I_0, $\mathcal{A}c^{but}$ et $\mathcal{A}c^1$.

Construction de l'ensemble I_0

Soit la fonction $h : M \to 2^{Li(At)}$ qui associe à chaque marquage $m \in M$ le sous-ensemble $h(m) = \{p \mid m(p) = 1\} \cup \{\neg p \mid m(p) = 0\}$ de $Li(At)$. L'ensemble maximal consistant I_0 est construit à partir des marquages initiaux $m_0^{\mathcal{N}}$ et $m_0^{\mathcal{O}}$ de la façon suivante : $I_0 = h(m_0^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}})$.

Construction de l'automate conditionnel $\mathcal{A}c^{but}$

La construction de $\mathcal{A}c^{but}$ se fait à partir du réseau de Petri sauf $\mathcal{N}$. L'automate $\mathcal{A}c^{but} = (Q^{but}, q_0^{but}, \delta^{but})$ est tel que :

- $Q^{but} = \{q_0^{but}\}$ et
- $\delta^{but} \subseteq 2^{Li(At)} \times Q^{but} \times \Sigma \times Q^{but} \times 2^{Li(At)}$ est la relation de transition définie par : $(J, q_0^{but}, a, q_0^{but}, J') \in \delta^{but}$ ssi il existe une transition $t \in T^{\mathcal{N}}$ telle que les trois conditions suivantes sont satisfaites :
  1. $l(t) = a$,
  2. $J = {}^\bullet t$ et
  3. $J' = t^\bullet \cup \neg({}^\bullet t \setminus t^\bullet)$.

Dans la Figure 5.3, nous représentons la transition de $\mathcal{A}c^{but}$ associée à une transition t dans $\mathcal{N}$.

Construction de l'automate conditionnel $\mathcal{A}c^1$

La construction de l'automate conditionnel $\mathcal{A}c^1$ se fait à partir du réseau de Petri sauf $\mathcal{O}$, de la même façon que la construction de l'automate conditionnel $\mathcal{A}c^{but}$ se fait à partir du réseau de Petri sauf $\mathcal{N}$.

Nous avons construit tous les éléments de l'instance du problème $\mathcal{PCSC}(\equiv_{tr})$, il nous reste à vérifier les deux points suivants :

(1) ρ est calculable par une machine de Turing déterministe en utilisant un espace logarithmique,

(2) les automates finis $Exec(\mathcal{N}, m_0^{\mathcal{N}})$ (resp. $Exec(\mathcal{O}, m_0^{\mathcal{O}})$) et $Exec(\mathcal{A}c^{but}, I_0)$ (resp. $Exec(\mathcal{A}c^1, I_0)$) sont isomorphes.

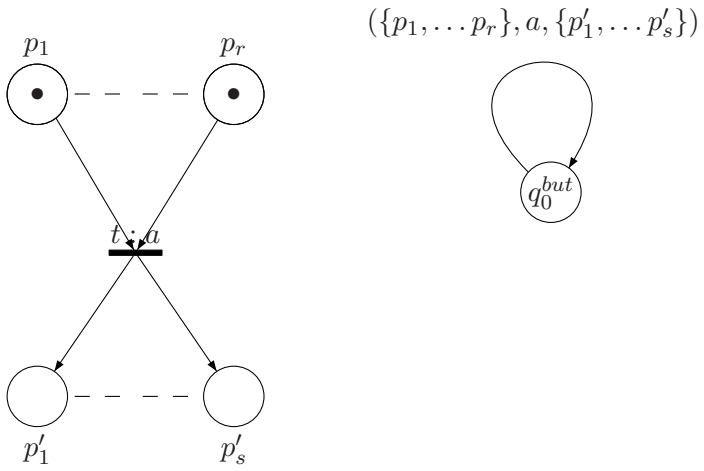

FIG. 5.3 – A gauche une transition t du réseau de Petri sauf $\mathcal{N}$ et à droite la transition qui lui est associée dans l'automate conditionnel $\mathcal{A}c^{but}$.

Vérification du point (1)

Une machine de Turing $\mathcal{M}$ calcule ρ de la façon suivante :

(a) La machine $\mathcal{M}$ écrit d'abord sur son ruban de sortie les ensembles Σ et At. Elle écrit ensuite l'ensemble $\{q_0^{but}\}$, contenant l'unique état de $\mathcal{A}c^{but}$ et l'ensemble $\{q_0^1\}$ contenant l'unique état de $\mathcal{A}c^1$.

(b) Ensuite, la machine va écrire la partie maximale consistante $I_0 = h(m_0^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}})$ de $Li(At)$. Pour ce faire, $\mathcal{M}$ lit sur son ruban d'entrée, pour chaque place $p \in P^{\mathcal{N}}$ (resp. $p \in P^{\mathcal{O}}$) la valeur de $m_0^{\mathcal{N}}(p)$ (resp. $m_0^{\mathcal{O}}(p)$) et ajoute p ou $\neg p$ à I_0 selon que cette valeur est 1 ou 0.

(c) Pour finir, la machine $\mathcal{M}$ va écrire, sur son ruban de sortie, la fonction de transition δ^{but} (resp. δ^1) de $\mathcal{A}c^{but}$ (resp. $\mathcal{A}c^1$). Pour ce faire, elle utilise deux compteurs i et j. Le compteur i va successivement prendre comme valeurs les transitions de $\mathcal{N}$ (resp. $\mathcal{O}$). Pour chaque valeur de i, $\mathcal{M}$ écrit toute les transitions $(J, q_0^{but}, a, q_0^{but}, J') \in \delta^{but}$ (resp. $(J, q_0^1, a, q_0^1, J') \in \delta^1$) où a est l'action correspondant à la transition i via la fonction l. Pour ce faire, $\mathcal{M}$ va donner successivement à j comme valeurs les places de $\mathcal{N}$ (resp. $\mathcal{O}$).

Toutes ces étapes peuvent bien sûr être réalisées de façon déterministe en utilisant un espace logarithmique.

Vérification du point (2)

Nous prouvons que :

$$Exec(\mathcal{N}, m_0^{\mathcal{N}}) \simeq_{iso} Exec(\mathcal{A}c^{but}, I_0).$$

Notons que la preuve est identique pour $Exec(\mathcal{O}, m_0^{\mathcal{O}}) \simeq_{iso} Exec(\mathcal{A}c^1, I_0)$. Nous procédons de la façon suivante. Les états de $Exec(\mathcal{N}, m_0^{\mathcal{N}})$ sont des marquages pour $\mathcal{N}$. Les états de $Exec(\mathcal{A}c^{but}, I_0)$ sont des couples de la forme (q_0^{but}, I) où q_0^{but} est l'unique état de $\mathcal{A}c^{but}$ et I est une partie maximale consistante de $Li(At)$.

Nous attirons votre attention sur le fait que, pour toute transition $(J, q_0^{but}, a, q_0^{but}, J')$ dans $\mathcal{A}c^{but}$ les ensembles J et J' sont inclus dans $Li(P^{\mathcal{N}})$. Par conséquent, les états accessibles dans $Exec(\mathcal{A}c^{but}, I_0)$, à partir de (q_0^{but}, I_0), sont de la forme $(q_0^{but}, (I^{\mathcal{N}} \cup h(m_0^{\mathcal{O}}))$ avec $I^{\mathcal{N}}$ un ensemble maximal consistant pour $Li(P^{\mathcal{N}})$.

Maintenant, considérons la fonction g qui associe à chaque marquage $m^{\mathcal{N}}$ accessible dans $Exec(\mathcal{N}, m_0^{\mathcal{N}})$ à partir de $m_0^{\mathcal{N}}$ le couple $(q_0^{but}, (h(m^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}}))$. La construction de la fonction h est telle que g est une bijection. De plus, $g(m_0^{\mathcal{N}}) = h(m_0^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}})$. Ainsi, $g(m_0^{\mathcal{N}}) = (q_0^{but}, I_0)$.

Dans le reste de la preuve, nous montrons que pour tout marquage $m_1^{\mathcal{N}}$, $m_2^{\mathcal{N}}$ pour $\mathcal{N}$ et pour toute action $a \in \Sigma$:

$$m_1^{\mathcal{N}}[a\rangle m_2^{\mathcal{N}} \iff g(m_1^{\mathcal{N}}) \rightarrow^{\{a\}}_{Exec(\mathcal{A}c^{but}, I_0)} g(m_2^{\mathcal{N}}).$$

($\Rightarrow$) Considérons l'implication de gauche à droite. Si

$$m_1^{\mathcal{N}}[a\rangle m_2^{\mathcal{N}}.$$

alors il existe une transition $t \in T^{\mathcal{N}}$ telle que :

- $l(t) = a$,
- pour tout $p \in {}^\bullet t$, $m_1^{\mathcal{N}}(p) = 1$ et
- $m_2^{\mathcal{N}}(p) = \begin{cases} m_1^{\mathcal{N}}(p) & \text{si } p \notin {}^\bullet t \cup t^\bullet \\ 1 & \text{si } p \in t^\bullet \\ 0 & \text{sinon} \end{cases}$

Par conséquent, dans l'automate $\mathcal{A}c^{but}$, $(J, q_0^{but}, a, q_0^{but}, J') \in \delta^{but}$ avec $J = {}^\bullet t$ et $J' = t^\bullet \cup \neg({}^\bullet t \setminus t^\bullet)$. Soit les ensembles maximaux consistants $I_1 = h(m_1^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}})$ et $I_2 = h(m_2^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}})$ et les états $g(m_1^{\mathcal{N}}) = (q_0, I_1)$ et $g(m_2^{\mathcal{N}}) = (q_0, I_2)$. Il est facile de vérifier que $I_2 = (I_1 \setminus \neg J') \cup J'$. On sait que $I_2 = h(m_2^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}})$. Soit $J'' = ({}^\bullet t \cup t^\bullet) \cup (\neg({}^\bullet t \cup t^\bullet))$. Intuitivement, J'' représente tout les littéraux qui peuvent être obtenus à partir de la transition t. Ainsi, $I_2 = (h(m_1^{\mathcal{N}}) \setminus J'') \cup t^\bullet \cup \neg({}^\bullet t \setminus t^\bullet) \cup h(m_0^{\mathcal{O}})$. Il s'en suit que, I_2=$(h(m_1^{\mathcal{N}}) \setminus J'') \cup h(m_0^{\mathcal{O}}) \cup J'$. Les ensembles $h(m_1^{\mathcal{N}})$ et $h(m_0^{\mathcal{O}})$

sont disjoints. Par conséquent, I_2=$((h(m_1^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}})) \setminus J'') \cup J'$. Sachant que l'ensemble $\neg J'$ est inclus dans J'' et que J' est maximal pour J'' [2], alors il est clair que, $I_2 = (I_1 \setminus \neg J') \cup J'$. De plus, sachant que pour tout $p \in {}^\bullet t$, $m_1^{\mathcal{N}}(p) = 1$ alors ${}^\bullet t \subseteq h(m_1^{\mathcal{N}})$. Il en résulte que ${}^\bullet t \subseteq I_1$. Par conséquent, $(q_0^{but}, I_1) \rightarrow^{\{a\}}_{Exec(\mathcal{A}c^{but}, I_0)} (q_0^{but}, I_2)$. Donc, $g(m_1^{\mathcal{N}}) \rightarrow^{\{a\}}_{Exec(\mathcal{A}c^{but}, I_0)} g(m_2^{\mathcal{N}})$.

($\Leftarrow$) Considérons l'implication de droite à gauche. Supposons que :

$$g(m_1^{\mathcal{N}}) \rightarrow^{\{a\}}_{Exec(\mathcal{A}c^{but}, I_0)} g(m_2^{\mathcal{N}}).$$

Par définition de g, on a :

$$(q_0^{but}, (h(m_1^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}}))) \rightarrow^{\{a\}}_{Exec(\mathcal{A}c^{but}, I_0)} (q_0^{but}, (h(m_2^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}}))).$$

Soit les ensembles maximaux consistants $I_1 = h(m_1^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}})$ et $I_2 = h(m_2^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}})$. La transition dans $Exec(\mathcal{A}c^{but}, I_0)$, implique qu' il existe une transition $(J, q_0^{but}, a, q_0^{but}, J') \in \delta^{but}$ telle que $J \subseteq h(m_1^{\mathcal{N}})$ et $I_2 = (I_1 \setminus \neg J') \cup J'$. D'après la construction de $\mathcal{A}c^{but}$, il existe une transition $t \in T^{\mathcal{N}}$ telle que :

- $l(t) = a$,
- $J = {}^\bullet t$ et
- $J' = t^\bullet \cup \neg({}^\bullet t \setminus t^\bullet)$.

Puisque $I_2 = (I_1 \setminus \neg J') \cup J'$ alors $I_2 = ((h(m_1^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}})) \setminus \neg J') \cup J'$. Sachant que $h(m_1^{\mathcal{N}})$ et $h(m_0^{\mathcal{O}})$ sont disjoints, il en résulte que $I_2 = (h(m_1^{\mathcal{N}}) \setminus \neg J') \cup J' \cup h(m_0^{\mathcal{O}})$. On sait que $I_2 = h(m_2^{\mathcal{N}}) \cup h(m_0^{\mathcal{O}})$. De ce fait, $h(m_2^{\mathcal{N}}) = (h(m_1^{\mathcal{N}}) \setminus \neg J') \cup J'$. D'une part, l'ensemble $(h(m_1^{\mathcal{N}}) \setminus \neg J')$ est composé d'un ensemble de littéraux inclus dans J. D'autre part, il contient un ensemble disjoint de $J'' = ({}^\bullet t \cup t^\bullet) \cup (\neg({}^\bullet t \cup t^\bullet))$. Par conséquent, $h(m_2^{\mathcal{N}}) = (h(m_1^{\mathcal{N}}) \setminus J'') \cup J'$. Ainsi, $h(m_2^{\mathcal{N}}) = (h(m_1^{\mathcal{N}}) \setminus J'') \cup t^\bullet \cup \neg({}^\bullet t \setminus t^\bullet)$. Par définition de h, on a :

$$m_2^{\mathcal{N}}(p) = \begin{cases} m_1^{\mathcal{N}}(p) & \text{si } p \notin {}^\bullet t \cup t^\bullet \\ 1 & \text{si } p \in t^\bullet \\ 0 & \text{sinon} \end{cases}$$

De plus, sachant que ${}^\bullet t \subseteq h(m_1^{\mathcal{N}})$, cela implique que pour tout $p \in {}^\bullet t$, $m_1^{\mathcal{N}}(p) = 1$. Par conséquent, $m_1^{\mathcal{N}}[a\rangle m_2^{\mathcal{N}}$. □

Théorème 5.3.2. *Le problème de la compostion $\mathcal{PCSC}(\subseteq_{tr})$ est EXPSPACE-difficile et le problème de la composition $\mathcal{PCSC}(\longleftrightarrow_{bi})$ est EXPTIME-difficile.*

[2] Pour plus de précision, voire la définition 3.2.3 d'un ensemble maximal pour un autre ensemble.

Démonstration. Il suffit d'utiliser la réduction du théorème 5.3.1 et de rappeler que le problème de la bisimulation entre réseaux de Petri saufs est EXPTIME-difficile [JM96] et que le problème de l'inclusion de traces entre réseaux de Petri saufs est EXPSPACE-difficiles [JM96]. □

Il est significatif de constater que dans la réduction utilisée dans la preuve du théorème 5.3.1, qui est la même que celle du théorème 5.3.2, on obtient un seul service disponible. Par conséquent, si nous considérons la restriction du problème de la composition qui consiste à considérer uniquement les instances avec $n = 1$, le problème reste EXPSPACE-difficile pour l'équivalence de traces et l'inclusion de traces et EXPTIME-difficile pour la bisimulation.

Théorème 5.3.3. *Le problème $\mathcal{PCSC}(\leq_{si})$ est EXPTIME-difficile.*

Démonstration. Considérons le problème de la simulation d'un automate fini par un produit d'automates finis :

Instance : des automates finis déterministes $\mathcal{A}$, $\mathcal{B}^1, \ldots, \mathcal{B}^n$ sur Σ.

Question : est il vrai que

$$\mathcal{A} \leq_{si} \mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n \,?$$

Ce problème est EXPTIME-difficile [MW08] et ses instances correspondent à celles du problème $\mathcal{PCSC}(\leq_{si})$ dans lesquelles $At = \emptyset$. Donc, le problème $\mathcal{PCSC}(\leq_{si})$ est EXPTIME-difficile. □

Soulignons que, dans le cas restrictif où le nombre de services disponibles n est fixé à 1, le problème de la composition pour la simulation reste EXPTIME-difficile. Comme dans la preuve du théoreme 5.3.3, la preuve est basée sur une réduction du problème de la simulation d'un automate fini par un produit d'automates finis au problème $\mathcal{PCSC}(\leq_{si})$ dans le cas où $n = 1$.

Théorème 5.3.4. *Dans le cas où $n = 1$, le problème $\mathcal{PCSC}(\leq_{si})$ est EXPTIME-difficile.*

Démonstration. Nous allons réduire le problème de la simulation d'un automate fini par un produit d'automates finis :

Instance : des automates finis déterministes $\mathcal{A}$, $\mathcal{B}^1, \ldots, \mathcal{B}^n$ sur Σ.

Question : est il vrai que

$$\mathcal{A} \leq_{si} \mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n \,?$$

qui est EXPTIME-difficile [MW08], au problème $\mathcal{PCSC}(\leq_{si})$ dans le cas où $n = 1$. Considérons une instance du problème ci-dessus. Soit $n \geq 2$, pour tout $i \in \{1, \ldots, n\}$, soit $\mathcal{B}^i = (Q^i, q_0^i, \rightarrow_{\mathcal{B}^i})$ et $\mathcal{A} = (Q, q_0, \rightarrow_{\mathcal{A}})$ des automates finis déterministes sur Σ. L'instance $\rho(n, \Sigma, \mathcal{A}, \mathcal{B}^1, \ldots, \mathcal{B}^n)$ de $\mathcal{PCSC}(\leq_{si})$, que nous construisons est donnée par un ensemble fini d'actions $\Sigma' = \Sigma$, un ensemble fini de formules atomiques At, un ensemble maximal consistant $I_0 \in 2^{Li(At)}$ et des automates conditionnels $\mathcal{A}c^{but}$ et $\mathcal{A}c^1$ sur Σ et At. Dans la suite, nous donnons le détail de la construction de At, I_0, $\mathcal{A}c^{but}$ et $\mathcal{A}c^1$.

Construction des ensembles At et I_0

Les formules atomiques vont nous permettre de coder les états des automates $\mathcal{B}^1, \ldots, \mathcal{B}^n$. Sans perte de généralité, nous supposons que :

$Card(Q^1) = \ldots = Card(Q^n)$, où $Card(.)$ représente la cardinalité.

Soit $t = \lceil log_2(Card(Q^1)) \rceil = \ldots = \lceil log_2(Card(Q^n)) \rceil$, où $\lceil . \rceil$ représente la partie entière supérieure. Pour tout $i \in \{1, \ldots, n\}$, nous considérons un ensemble $At^i = \{r_1^i, \ldots, r_t^i\}$ de t formules atomiques. Notons que les ensembles $At^1, \ldots, At^n$ sont deux à deux disjoints. L'ensemble At représente l'union des ensembles $At^1, \ldots, At^n$. Soit $f : Q^1 \cup \ldots \cup Q^n \rightarrow 2^{Li(At)}$ 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 $Li(At^i)$. Intuitivement, $f(q)$, représente le codage de q en binaire, sur t bits. L'ensemble I_0 est l'union du codage des états initiaux de $\mathcal{B}^1, \ldots, \mathcal{B}^n$. Formellement, $I_0 = f(q_0^1) \cup \ldots \cup f(q_0^n)$.

Pour une meilleur compréhension, considérons l'exemple suivant. Supposons que $n = 2$ et $\mid Q^1 \mid = \mid Q^2 \mid = 3$. Dans ce cas :

- $At = \{r_1^1, r_2^1, r_1^2, r_2^2\}$ et on peut choisir f en posant :
- $f(q_0^1) = \{\neg r_1^1, \neg r_2^1\}$, $f(q_1^1) = \{\neg r_1^1, r_2^1\}$, $f(q_2^1) = \{r_1^1, \neg r_2^1\}$,
- $f(q_0^2) = \{\neg r_1^2, \neg r_2^2\}$, $f(q_1^2) = \{\neg r_1^2, r_2^2\}$ et $f(q_2^2) = \{r_1^2, \neg r_2^2\}$.

Construction de l'automate conditionnel $\mathcal{A}c^{but}$

Concernant la construction de l'automate conditionnel $\mathcal{A}c^{but} = (Q^{but}, q_0^{but}, \delta^{but})$ sur Σ et At, il est construit à partir de l'automate $\mathcal{A}$ de la façon suivante :

- $Q^{but} = Q$,
- $q_0^{but} = q_0$ et
- $\delta^{but} \subseteq 2^{Li(At)} \times Q^{but} \times \Sigma \times Q^{but} \times 2^{Li(At)}$ est la relation de transition définie par :

$$(\emptyset, q_1, a, q_2, \emptyset) \text{ ssi } q_1 \rightarrow_{\mathcal{A}}^{a} q_2.$$

FIG. 5.4 – À gauche, une transition de l'automate $\mathcal{A}$. À droite, la transition qui lui est associée dans l'automate conditionnel $\mathcal{A}c^{but}$.

Dans la Figure 5.4, nous représentons une transition de $\mathcal{A}c^{but}$ obtenue à partir d'une transition de $\mathcal{A}$.

Construction de l'automate conditionnel $\mathcal{A}c^1$

L'automate conditionnel $\mathcal{A}c^1$ est construit à partir des automates $\mathcal{B}^1, \ldots, \mathcal{B}^n$ de la façon suivante. Soit $\mathcal{B}' = (Q^{\mathcal{B}'}, q_0^{\mathcal{B}'}, \rightarrow_{\mathcal{B}'})$ le produit asynchrone des $\mathcal{B}^i$, $i \in \{1, \ldots, n\}$. Considérons l'automate conditionnel $\mathcal{A}c^1 = (Q', q_0', \delta')$ sur Σ et At tel que :

- $Q' = \{q_0'\}$ et
- $\delta' \subseteq 2^{Li(At)} \times Q' \times \Sigma \times Q' \times 2^{Li(At)}$ est la relation de transition définie par : $(J, q_0', a, q_0', J') \in \delta'$ ssi il existe $i \in \{1, \ldots, n\}$ et des états $q_1^i, q_2^i \in Q^i$ tels que :
  - $q_1^i \rightarrow_{\mathcal{B}^i}^{\{a\}} q_2^i$,
  - $J = f(q_1^i)$ et
  - $J' = f(q_2^i)$.

Dans la Figure 5.5, pour un $i \in \{1, \ldots, n\}$, nous représentons une transition de $\mathcal{A}c^1$ obtenue à partir d'une transition de $\mathcal{B}^i$.

Dans ce qui suit, nous vérifions les trois points suivants :

(1) La réduction ρ est calculable par une machine de Turing déterministe en utilisant un espace logarithmique.
(2) Les automates finis $\mathcal{A}$ et $Exec(\mathcal{A}c^{but}, I_0)$ sont isomorphes.
(3) Les automates finis $\mathcal{B}'$ et $Exec(\mathcal{A}c^1, I_0)$ sont isomorphes.

Vérification du point (1)

L'argument est semblable à celui que nous avons développé dans la preuve du théorème 5.3.1. Une machine de Turing $\mathcal{M}$ calcule ρ de la façon

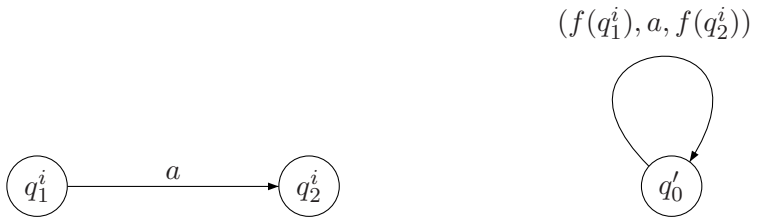

FIG. 5.5 – À gauche, une transition de l'automate $\mathcal{B}^i$, pour un $i \in \{1, \ldots, n\}$. À droite, la transition qui lui est associée dans l'automate conditionnel $\mathcal{A}c^1$.

suivante :

(a) La machine $\mathcal{M}$ écrit d'abord sur son ruban de sortie l'ensembles Σ et l'ensemble $\{q_0^1\}$ contenant l'unique état de $\mathcal{A}c^1$. Elle écrit ensuite l'ensemble Q^{but} et l'état $\{q_0^{but}\}$ qu'elle lit à partir de son ruban d'entrée. Afin d'écrire δ^{but}, pour chaque transition $(q_1 \rightarrow_{\mathcal{A}} q_2)$ que la machine lit, sur son ruban d'entrée, elle écrit $(\emptyset, q_1, a, q_2, \emptyset)$ sur son ruban de sortie.

(b) Maintenant, la machine va écrire les ensembles $At^1, \ldots At^n$ qui constituent l'ensemble At. Pour cela, elle va utiliser un compteur t pour calculer $\lceil log_2(\mid Q^n \mid) \rceil$. Ensuite elle écrit un a un les ensembles $At^i = \{r_1^i, \ldots, r_t^i\}$, pour tout $i \in \{1, \ldots, n\}$. Pour écrire $I_0 = f(q_0^1) \cup \ldots \cup f(q_0^n)$ la machine va écrire pour chaque q_0^i les littéraux $\neg r_1^i, \ldots, \neg r_t^i$.

(c) Pour finir, la machine $\mathcal{M}$ va écrire, sur son ruban de sortie, la fonction de transition δ^1 de $\mathcal{A}c^1$. Pour ce faire, elle utilise deux compteurs i et j. Le compteur i va représenter un automate $\mathcal{B}^i$ et le compteur j une transition. Pour chaque valeur de i, si $\mathcal{M}$ lit la transition j définie par $q_1^i \rightarrow_{\mathcal{B}^i}^{\{a\}} q_2^i$ alors elle écrit la transitions $(f(q_1^i), q_0^1, a, q_0^1, f(q_2^i)) \in \delta^1$.

Toutes ces étapes peuvent bien sûr être faites de façon déterministe en utilisant un espace logarithmique.

Vérification du point (2)

L'automate conditionnel $\mathcal{A}c^{but}$ est la simple réécriture de l'automate fini déterministe $\mathcal{A}$ sous le format d'automate conditionnel. Dans l'automate fini $Exec(\mathcal{A}c^{but}, I_0)$, les états accessibles à partir de l'état initial (q_0, I_0) sont

tous de la forme (q, I_0), $q \in Q$. Si nous considérons la fonction g qui associe à chaque état accessible (q, I_0)dans $Exec(\mathcal{A}c^{but}, I_0)$ l'état q dans $\mathcal{A}$ alors il est claire que g est un isomorphisme.

Vérification du point (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 $Exec(\mathcal{A}c^1, I_0)$ sont des couples de la forme (q'_0, I) où q'_0 est l'unique état de $\mathcal{A}c^1$ 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 unique (q'_0, I) tel que $I = f(q^1) \cup \ldots \cup f(q^n)$. Il est clair que g est une bijection et que $g(q_0^1, \ldots q_0^n) = (q'_0, I_0)$.

Il reste à prouver que pour tout n-uplet d'états $(q_1^1, \ldots, q_1^n)$, $(q_2^1, \ldots, q_2^n)$ de $\mathcal{B}'$ et pour toute action $a \in \Sigma$:

$$(q_1^1, \ldots, q_1^n) \rightarrow^a_{\mathcal{B}'} (q_2^1, \ldots, q_2^n) \Longleftrightarrow g(q_1^1, \ldots, q_1^n) \rightarrow^a_{Exec(\mathcal{A}c^{but}, I_0)} g(q_2^1, \ldots, q_2^n).$$

($\Rightarrow$) Considérons l'implication de gauche à droite. Supposons que :

$$(q_1^1, \ldots, q_1^n) \rightarrow^{\{a\}}_{\mathcal{B}'} (q_2^1, \ldots, q_2^n).$$

Par définition du produit asynchrone, cela implique qu'il existe un $i \in \{1, \ldots, n\}$ tel que :

- $q_1^i \rightarrow^{\{a\}}_{\mathcal{B}^i} q_2^i$ et
- pour tout $j \neq i$, $q_2^j = q_1^j$.

Ainsi, par construction de $\mathcal{A}c^1$, $(f(q_1^i), q'_0, a, q'_0, f(q_2^i)) \in \delta'$. Considérons l'ensemble maximal consistant $I_2 = f(q_1^1) \cup \ldots \cup f(q_2^i) \cup \ldots \cup f(q_1^n)$ et l'état $g(q_2^1, \ldots, q_2^n) = (q'_0, I_2)$. Soit $I_1 = f(q_1^1) \cup \ldots \cup f(q_1^n)$. Puisque $g(q_1^1, \ldots, q_1^n) = (q'_0, I_1)$ alors $f(q_1^i) \subseteq I_1$. Par construction de $Exec(\mathcal{A}c^1, I_0)$, puisque $f(q_1^i) \subseteq I_1$ et $I_2 = (I_1 \setminus \neg f(q_2^i)) \cup f(q_2^i)$ alors :

$$(q'_0, I_1) \rightarrow^{\{a\}}_{Exec(\mathcal{A}c^1, I_0)} (q'_0, I_2).$$

Par conséquent, $g(q_1^1, \ldots, q_1^n) \rightarrow^{\{a\}}_{Exec(\mathcal{A}c^1, I_0)} g(q_2^1, \ldots, q_2^n)$.

($\Leftarrow$) Considérons l'implication de droite à gauche. Supposons que :

$$g(q_1^1, \ldots, q_1^n) \rightarrow^{\{a\}}_{Exec(\mathcal{A}c^1, I_0)} g(q_2^1, \ldots, q_2^n). \tag{5.1}$$

Soit $I_1 = f(q_1^1) \cup \ldots \cup f(q_1^n)$ et $I_2 = f(q_2^1) \cup \ldots \cup f(q_2^n)$. Par construction de g, $g(q_1^1, \ldots, q_1^n) = (q'_0, I_1)$ et $g(q_2^1, \ldots, q_2^n) = (q'_0, I_2)$. Par construction de $Exec(\mathcal{A}c^1, I_0)$, l'équation 5.1 implique qu'il existe des ensembles consistants $J, J' \in 2^{Li(At)}$ tels que :

- $J \subseteq I_1$,
- $I_2 = (I_1 \setminus \neg J') \cup J'$ et
- $(J, q'_0, a, q'_0, J') \in \delta'$.

D'après la construction de $\mathcal{A}c^1$, il existe un $i \in \{1, \ldots, n\}$ et des états $q_1^i, q_2^i \in Q^i$ tels que

- $q_1^i \rightarrow_{\mathcal{B}^i}^{\{a\}} q_2^i$,
- $J = f(q_1^i)$ et
- $J' = f(q_2^i)$.

Montrons que $(q_2^1, \ldots, q_2^i, \ldots, q_2^n) = (q_1^1, \ldots, q_2^i, \ldots, q_1^n)$. Sachant que $I_2 = (I_1 \setminus \neg f(q_i^2)) \cup f(q_2^i)$ et que $I_1 = f(q_1^1) \cup \ldots f(q_1^i) \ldots \cup f(q_1^n)$ alors $I_2 = f(q_1^1) \cup \ldots f(q_2^i) \ldots \cup f(q_1^n)$. Vu que g est une bijection alors $(q_2^1, \ldots, q_2^i, \ldots, q_2^n) = (q_1^1, \ldots, q_2^i, \ldots, q_1^n)$. A partir de la définition d'un produit asynchrone on a, si $q_1^i \rightarrow_{\mathcal{B}^i}^{\{a\}} q_2^i$ alors :

$$(q_1^1, \ldots, q_1^i, \ldots q_1^n) \rightarrow_{\mathcal{B}'}^{\{a\}} (q_1^1, \ldots, q_2^i, \ldots, q_1^n).$$

□

5.3.2 Bornes supérieures de complexité

Dans cette section, nous caractérisons les bornes supérieures de complexité pour les problèmes $\mathcal{PCSC}(\subseteq_{tr})$, $\mathcal{PCSC}(\equiv_{tr})$, $\mathcal{PCSC}(\leq_{si})$ et $\mathcal{PCSC}(\longleftrightarrow_{bi})$. Pour cela, nous allons proposer des algorithmes basés sur des procédures connues, qui permettent de résoudre des problèmes d'équivalences entre automates finis. Nous allons montrer que les problèmes $\mathcal{PCSC}(\subseteq_{tr})$ et $\mathcal{PCSC}(\equiv_{tr})$ sont dans EXPSPACE. Nous allons également montrer que $\mathcal{PCSC}(\leq_{si})$ et $\mathcal{PCSC}(\longleftrightarrow_{bi})$ sont dans EXPTIME. Rappelons quelques résultats. Les problèmes de l'inclusion de traces et de l'équivalence de traces, entre deux automates finis, sont dans PSPACE [GJ90]. Les problèmes de la simulation et de la bisimulation, entre deux automates finis, sont dans PTIME [HS96]. Nous remarquons une augmentation exponentielle de la complexité des problèmes, lorsqu'on considère des automates conditionnels. Cela est du d'une part, à la taille exponentielle des exécutions associées aux automates conditionnels, par rapport au nombre de formules atomiques. D'autre part, la taille du produit d'automates conditionnels est exponentiel par rapport au nombre d'automates considérés.

Théorème 5.3.5. *$\mathcal{PCSC}(\longleftrightarrow_{bi})$ est dans EXPTIME.*

Démonstration. Considérons l'algorithme suivant : Un temps exponentiel, par rapport à n et à la taille de At, est nécessaire pour que l'algorithme

Algorithm 1 Algorithme pour la résolution de $\mathcal{PCSC}(\longleftrightarrow_{bi})$

1: **Resoudre-**$\mathcal{PCSC}(\longleftrightarrow_{bi})(\Sigma, At, I_0, n, \mathcal{A}c^{but}, \mathcal{A}c^1, \ldots, \mathcal{A}c^n)$
2: $\mathcal{A} \leftarrow Exec(\mathcal{A}c^{but}, I_0)$
3: $\mathcal{A}c' \leftarrow \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n$
4: $\mathcal{A}' \leftarrow Exec(\mathcal{A}c', I_0)$
5: **if** $\mathcal{A} \longleftrightarrow_{bi} \mathcal{A}'$ **then retourner** Vrai
6: **else retourner** Faux
7: **end if**
8: **end procedure**

retourne une réponse. Le détail du temps nécessaire pour les quatre étapes de la procédure est donné comme suit :

- Dans la ligne 2, un temps exponentiel par rapport à la taille de At est suffisant pour construire $\mathcal{A}$ à partir de $\mathcal{A}c^{but}$ et de I_0.
- Dans la ligne 3, un temps exponentiel par rapport à n et à la taille de At est suffisant pour construire $\mathcal{A}c'$ à partir de $\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n$.
- Dans la ligne 4, un temps exponentiel par rapport à la taille de At est suffisant pour construire $\mathcal{A}'$ à partir de $\mathcal{A}c'$ et de I_0.
- Dans la ligne 5, un temps polynomial par rapport à la taille de $\mathcal{A}$ et $\mathcal{A}'$ est suffisant pour déterminer si $\mathcal{A}$ et $\mathcal{A}'$ sont bisimilaires [HS96].

Notre algorithme est correct et complet, sachant que dans [HS96], la procédure proposée retourne "vrai" ssi $\mathcal{A}$ et $\mathcal{A}'$. Par conséquent $\mathcal{PCSC}(\longleftrightarrow_{bi}$) est dans EXPTIME. □

Un argument semblable à l'argument précédent montrerait que :

Théorème 5.3.6. *$\mathcal{PCSC}(\leq_{si})$ est dans EXPTIME, $\mathcal{PCSC}(\subseteq_{tr})$ et $\mathcal{PCSC}(\equiv_{tr}))$ sont dans EXPSPACE.*

Démonstration. Il suffit de rappeler que le problème de la simulation entre automates finis est dans P [HS96], que le problème de l'inclusion de traces entre automates finis est dans PSPACE [GJ90] et que le problème de l'équivalence de traces entre automates finis est dans PSPACE [GJ90]. □

Soulignons que dans le cas où le nombre de services disponibles n est fixé à 1, le problème de la composition pour l'inclusion de traces et l'équivalence de traces reste EXPSPACE-complet. De même, le problème reste EXPTIME-complet, pour la simulation et la bisimulation. Cela est du au coût des étapes 1. et 3. de l'algorithme précédemment présenté.

Problème	Complexité
$\mathcal{PCSC}(\subseteq_{tr})$	*EXPSPACE-complet*
$\mathcal{PCSC}(\equiv_{tr})$	*EXPSPACE-complet*
$\mathcal{PCSC}(\leq_{si})$	*EXPTIME-complet*
$\mathcal{PCSC}(\longleftrightarrow_{bi})$	*EXPTIME-complet*

TAB. 5.1 – Tableau récapitulatif, dans le cas où l'environnement est sans communication.

5.4 Conclusion

Dans ce chapitre, nous avons considéré un modèle simplifié des services, dans un environnement sans communications. Ce modèle peut être vu comme une abstraction du modèle présenté dans le Chapitre 4. Dans le cas où les services ne peuvent pas communiquer, nous avons montré que le problème de la composition pour la bisimulation et la simulation est EXPTIME-complet. Concernant l'inclusion de traces et l'équivalence de traces, nous avons montré que le problème de la composition est EXPSPASE-complet. De plus, nous avons montré que ces résultats restent valables dans le cas où la communauté contient un unique service disponible. Le Tableau 5.1 récapitule tout ces résultats.

Chapitre 6

Composition de services dans un environnement avec communication asynchrone

Dans ce chapitre, nous nous intéressons au problème de la composition de services, dans le cas où la communication entre ces derniers est asynchrone [BCF08b, BCF08a]. Par communication asynchrone nous voulons dire que l'envoi ou la réception de messages s'effectue, par les services, de façon autonome. En d'autres termes, lorsqu'un service envoie un message il n'a pas besoin de se synchroniser avec les autres services disponibles. Dans un premier temps, nous présentons le cadre formel que nous utilisons pour représenter les services. Ensuite, nous décrivons le problème de la composition que nous considérons. Nous donnerons enfin l'ensemble de nos résultats concernant ce problème.

6.1 Modèle des services

Le modèle que nous considérons, pour la représentation des services, est identique à celui que nous avons décrit dans le Chapitre 4. L'unique différence est que nous considérons le cas particulier où la taille des ports est bornée par un entier $k \in \mathbb{N}$. Sans perte de généralité, nous considérons que tout les ports sont bornés par le même entier k. Nous rappelons que dans ce modèle, les services partagent un ensemble fini d'actions noté Σ, un ensemble fini de formules atomiques noté At et un ensemble fini de ports noté $Port$. Les actions dans Σ représentent les actions internes qu'un service peut effectuer. Pour tout $\pi \in port$, les actions de la formes $!\pi$ représentent

l'envoi d'un message dans le port π. Réciproquement, les actions de la forme $?\pi$ représentent la réception d'un message dans le port π. Toutes les actions effectuées par un service, que ce soit des actions internes ou des actions de communications, peuvent être conditionnées par un sous-ensemble[1] de $Li(At)$. Les actions peuvent engendrer des effets, également représentés par un sous ensemble de $Li(At)$. Nous rappelons ci-dessous la définition d'un service et de son exécution.

Définition 6.1.1 (**Service**). Un *service* est un automate communicant conditionnel (ACC) $\mathcal{A}c = (Q, q_0, F, \delta)$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At tel que :
- Q est un ensemble fini d'états,
- q_0 est l'état initial,
- F est l'ensemble des états finaux,
- $\delta \subseteq 2^{Li(At)} \times Q \times (\Sigma \cup (\{!, ?\} \times Port)) \times Q \times 2^{Li(At)}$ est une relation de transition.

Dans notre modèle, nous utiliserons un service but, un service client, des services disponibles et un service médiateur. Le service but représente le service qui pourra réaliser les actions souhaitées par un client. Le service client est un service qui ne peut effectuer que des actions de communications. De plus, les messages envoyés par le client sont reçus par le service but. Réciproquement, les messages envoyés par le service but sont reçus par le client. Pour définir le service client, nous avons besoin de deux états : à partir du premier état il ne peut que transmettre des messages et à partir du second il ne peut que recevoir des messages. Les services disponibles sont les services répertoriés dans l'annuaire UDDI (Universel Description, Discovery, and Integration) des services et qui sont mis à la disposition d'autres services. Enfin, le service médiateur est un service qui communique avec les services disponibles et qui peut leur servir d'intermédiaire. Par exemple, si deux services disponibles ne communiquent pas sur les mêmes ports alors le services médiateurs se chargera de récupérer les messages de l'un et de les envoyer à l'autre. Ainsi, les services particuliers qui ne peuvent effectuer que de la communication sont le service client et le service médiateur.

Définition 6.1.2 (**Service client**). Un *service client* est un automate communicant conditionnel $\mathcal{A}c^{client} = (Q^{client}, q_0^{client}, F^{client}, \delta^{client})$ sur $\{!, ?\} \times Port$ et At, tel que $Q^{client} = \{q_0^{client}, q_1^{client}\}$ est une paire d'états, $F^{client} = Q^{client}$ et pour tout port $\pi \in Port$ et pour tout ensemble $J, J' \subseteq Li(At)$ on a :

[1] L'ensemble des littéraux qu'on peut construire à partir de At est noté $Li(At)$, voire 3.2.1.

- $(J, q_1^{client}, !\pi, q_i^{client}, J') \notin \delta^{client}$ où $q_i^{client} \in Q^{client}$, $i \in \{0,1\}$,
- $(J, q_0^{client}, ?\pi, q_i^{client}, J') \notin \delta^{client}$ où $q_i^{client} \in Q^{client}$, $i \in \{0,1\}$ et

Définition 6.1.3 (Service médiateur). Un *service médiateur* est un automate communicant conditionnel $\mathcal{A}c^{med} = (Q^{med}, q_0^{med}, F^{med}, \delta^{med})$ sur $\{!,?\} \times Port$ et At tel que $\delta^{med} \subseteq \{\emptyset\} \times Q^{med} \times (\{!,?\} \times Port) \times Q^{med} \times \{\emptyset\}$.

En ayant les ports initialement vides et à partir d'un ensemble initial $I_0 \subseteq Li(At)$ maximal consistant, l'exécution d'un automate communicant conditionnel nous permet de représenter l'évolution du contenu des ports et celle de l'ensemble I_0. Pour représenter le contenu des ports, nous utilisons les fonctions $\gamma : Port \to \{1, \ldots, k\}$ avec $k \in \mathbb{N}$. L'ensemble de toutes ces fonctions est noté Γ.

Définition 6.1.4 (Exécution des ACC). Soit un entier $k \in \mathbb{N}$ et $\mathcal{A}c = (Q, q_0, F, \delta)$ un automate communicant conditionnel sur $\Sigma \cup (\{!,?\} \times Port)$ et At. Nous appelons *exécution* de $\mathcal{A}c$, l'automate $Exec(\mathcal{A}c, I_0) = (Q', q_0', F', \to')$ défini sur $\Sigma \cup (\{!,?\} \times Port)$ tel que :

- $Q' = \{(q, \gamma, I) \mid q \in Q, \gamma \in \Gamma$ et $I \subseteq Li(At)$ est une partie maximale consistante$\}$,
- $q_0' = (q_0, \gamma_0, I_0)$ où $\forall \pi \in Port, \gamma_0(\pi) = 0$,
- $F' = \{(q, \gamma, I) \mid q \in F\}$ et
- $\to' \subseteq Q' \times \Sigma \cup (\{!,?\} \times Port) \times Q'$ est la relation de transition définie par $((q, \gamma, I), a, (q', \gamma', I')) \in \to'$ ssi il existe une transition $(J, q, a, q', J') \in \delta$ telle que $J \subseteq I$, $I' = (I \setminus \neg J') \cup J'$ où $\neg J' = \{p : \neg p \in J'\} \cup \{\neg p : p \in J'\}$ et
  - $a \in \Sigma$ et $\gamma = \gamma'$,
  - $a = ?\pi$, $\gamma(\pi) > 0$, $\gamma'(\pi) = \gamma(\pi) - 1$ et $\gamma'(\pi') = \gamma(\pi')$ pour $\pi' \neq \pi$ et
  - $a = !\pi$, $\gamma(\pi) < k$, $\gamma'(\pi) = \gamma(\pi) + 1$ et $\gamma'(\pi') = \gamma(\pi')$ pour $\pi' \neq \pi$.

6.2 Composition des services

Le problème de la composition consiste, étant donné un service client, un service but et des service disponibles, à déterminer l'existence d'un service médiateur tel que : le comportement du système composé du service client et du service but soit équivalent au comportement du système composé du client, du médiateur et des services disponibles. L'équivalence doit être satisfaite lorsque d'une part la communication entre les services disponibles n'est pas observable et d'autre part la communication entre les services et le médiateur n'est également pas observable. La communication non observable se fera à travers l'ensemble de ports $Port' \subseteq Port$. Plus précisement, notre

problème de la composition est le problème de décision défini de la façon suivante :

Problème $\mathcal{PBC}(\approx)$ **:** problème borné de la composition

Instance : un entier k, les ensembles finis Σ, $Port$ et $Port' \subseteq Port$, At et $I_0 \subseteq Li(At)$, un automate communicant conditionnel $\mathcal{A}c^{client}$ sur $\{!,?\} \times Port$ et At et des automates communicants conditionnels finis $\mathcal{A}c^{but}$, $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At.

Question : existe-t-il un service médiateur $\mathcal{A}c^{med}$ sur $\{!,?\} \times Port$ et At, tel que

$$Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, I_0)$$
$$\approx$$
$$Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast \mathcal{A}c^{med}, I_0)(\{!,?\} \times Port') \, ?$$

où $\approx \in \{\subseteq_{tr}, \equiv_{tr}, \leq_{si}, \longleftrightarrow_{bi}\}$.

Les relations auxquels nous nous intéressons sont l'inclusion de traces, l'équivalence de traces, la simulation et la bisimulation.

6.3 Résultats de complexité : bornes inférieures

Dans cette section et la section suivante, nous présentons des résultats concernant les quatre problèmes de la composition $\mathcal{PBC}(\subseteq_{tr})$, $\mathcal{PBC}(\equiv_{tr})$, $\mathcal{PBC}(\leq_{si})$ et $\mathcal{PBC}(\longleftrightarrow_{bi})$. Nous considérerons que les ports peuvent contenir au plus un message à la fois (c'est-à-dire $k = 1$). Pour le cas général avec $k \geq 1$, le raisonnement sera identique. Les réductions utilisées pour le calcul de la borne inférieure dans le cas où $k = 1$ seront les mêmes que pour le cas où $k \geq 1$. Les procédures utilisées pour le cas où $k = 1$ seront les mêmes que celles utilisées pour le cas où $k \geq 1$. L'espace ainsi que le temps nécessaire pour résoudre le cas où $k = 1$ et $k \geq 1$ seront du même ordre de grandeur par rapport à la taille de l'entrée du problème. Nous donnerons plus de précision, lorsque nous décrirons les différentes procédures de résolutions des problèmes.

Dans ce qui suit, nous montrons que les problèmes $\mathcal{PBC}(\subseteq_{tr})$ et $\mathcal{PBC}(\equiv_{tr}$) sont EXPSPACE-difficiles et que les problèmes $\mathcal{PBC}(\leq_{si})$ et $\mathcal{PBC}(\longleftrightarrow_{bi})$ sont EXPTIME-difficiles. Pour prouver que le problème $\mathcal{PBC}(\subseteq_{tr})$ est EXPSPACE-difficile, nous utilisons une réduction, en espace logarithmique, du problème de l'universalité des langages réguliers avec carré, connu pour être EXPSPACE-difficile [MS72]. Pour montrer que $\mathcal{PBC}(\leq_{si})$ est EXPTIME-difficile, nous utilisons une réduction, en espace logarithmique, du problème qui consiste à déterminer si un automate fini est similaire

au produit asynchrone de n automates finis, connu pour être EXPTIME-difficile [MW08]. En ce qui concerne le problème $\mathcal{PBC}(\equiv_{tr})$ (resp. $\mathcal{PBC}(\longleftrightarrow_{bi}$)), nous utilisons une réduction, en espace logarithmique, du problème de l'inclusion de traces (resp. bisimulation) entre deux réseaux de Petri 1-saufs, connu pour être EXPSPACE-difficile [JM96] (resp. EXPTIME-difficile [JM96]).

6.3.1 Inclusion de traces

Dans le Chapitre 4, plus précisément dans le lemme 4.2.4, nous avons prouvé que le problème de l'équivalence avec le médiateur large $\mathcal{PEL}(\subseteq_{tr})$ se réduit au problème de la composition $\mathcal{PC}(\subseteq_{tr})$, en utilisant un espace logarithmique. L'unique différence entre le problème $\mathcal{PC}(\subseteq_{tr})$ et $\mathcal{PBC}(\subseteq_{tr})$ est que les ports peuvent contenir au plus un message. Ainsi, la réduction proposée dans la preuve du lemme 4.2.4 est applicable pour réduire le problème $\mathcal{PEL}(\subseteq_{tr})$ au problème $\mathcal{PBC}(\subseteq_{tr})$. Par conséquent, pour prouver que $\mathcal{PBC}(\subseteq_{tr})$ est EXPSPACE-difficile, il suffit de prouver que $\mathcal{PEL}(\subseteq_{tr})$ est EXPSPACE-difficile. Pour cela, nous utilisons une réduction du problème de l'universalité des expression régulière avec carré, qui est EXPSPACE-difficile [MS72], au problème $\mathcal{PEL}(\subseteq_{tr})$. Nous montrons, également, que cette réduction est faite en espace logarithmique. Nous rappelons ci-dessous, la définition d'une expression régulière avec carré et la définition du problème de l'universalité des expression régulière avec carré.

Définition 6.3.1 (**Expression régulière avec carré**). Soit Σ un alphabet fini et Σ^* l'ensemble des mots de longueur finie sur Σ. Nous notons le mot vide (de longueur 0) par ϵ. *Une expression régulière avec carré* α définie sur Σ est construite comme suit :

$$\alpha := \epsilon \mid a \mid \gamma \circ \sigma \mid \gamma \cup \sigma \mid \gamma^+ \mid \gamma^2, \text{ où } a \in \Sigma.$$

Chaque expression α définit un langage rationnel sur Σ. Nous notons par $L(\alpha)$ le langage défini par α. De plus, le nombre d'occurrences dans α des opération $\epsilon, \circ, \cup, ^+, ^2$ sera noté $Op(\alpha)$. Considérons le problème de décision suivant :

Problème UER **:** problème de l'universalité des expressions régulières avec carré

Instance : un alphabet fini Σ et une expression régulière avec carré α définie sur Σ.

Question : est-il vrai que $L(\alpha)=\Sigma^*$?

Notre objectif est de réduire ce problème au problème de l'équivalence avec le médiateur large $\mathcal{PEL}(\subseteq_{tr})$. Rappelons d'abord la définition d'un médiateur large pour un ensemble de ports ainsi que la description du problème $\mathcal{PEL}(\subseteq_{tr})$.

Définition 6.3.2 (**Service médiateur large**). Le *service médiateur large* pour $Port$ est un automate communicant conditionnel $\mathcal{A}c^{L_{Port}} = (Q^{L_{Port}}, q_0^{L_{Port}}, F^{L_{Port}}, \delta^{L_{Port}})$ sur $\{!,?\} \times Port$ et At tel que :
- $Q^{L_{Port}} = \{q_0^{L_{Port}}\}$,
- $F^{L_{Port}} = \{q_0^{L_{Port}}\}$ et
- $\delta^{L_{Port}} = \{(\emptyset, q_0^{L_{Port}}, a, q_0^{L_{Port}}, \emptyset) \mid a \in \{!,?\} \times Port\}$.

Nous pouvons considérer comme exemple le médiateur large de la Figure 6.1. Dans cette exemple, nous considérons que $Port = \{\pi_1, \pi_2\}$.

$$(\emptyset, !\pi_1, \emptyset), (\emptyset, ?\pi_1, \emptyset), (\emptyset, !\pi_2, \emptyset), (\emptyset, ?\pi_2, \emptyset)$$

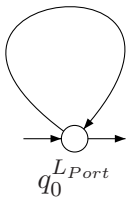

FIG. 6.1 – Automate fini $\mathcal{A}c^{L_{port}}$, dans le cas où $Port = \{\pi_1, \pi_2\}$.

Problème $\mathcal{PEL}(\subseteq_{tr})$ **:** problème d'équivalence avec le médiateur large

Instance : un ensemble fini d'actions Σ, des ensembles finis de ports $Port$ et $Port' \subseteq Port$, un ensemble fini de formules atomiques At, un ensemble maximal consistant de littéraux I_0 sur At et des automates communicants conditionnels finis $\mathcal{A}c$ et $\mathcal{A}c'$ sur $\Sigma \cup (\{!,?\} \times Port)$, et At.

Question : est-ce-que

$$Exec(\mathcal{A}c, I_0) \subseteq_{tr} Exec(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port}}, I_0)(\{!,?\} \times Port') \,?$$

Soulignons qu'un port $\pi \in Port$, peut contenir au plus un message.

Théorème 6.3.3. *Le problème de la compostion $\mathcal{PBC}(\subseteq_{tr})$ est EXPSPACE-difficile.*

Démonstration. Dans ce qui suit, nous prouvons que le problème UER se réduit en espace logarithmique au problème $\mathcal{PEL}(\subseteq_{tr})$. Soit une instance de UER donnée par Σ et α. L'instance $\rho(\Sigma, \alpha)$ de $\mathcal{PEL}(\subseteq_{tr})$ que nous construisons est donnée par Σ', $Port_\alpha$, $Port'_\alpha$, At_α, I_0^α, $\mathcal{A}c$ et $\mathcal{A}c'_\alpha$. Plus précisément, fixons d'abord $\Sigma' = \Sigma$. La construction de $Port_\alpha$, At_α et $\mathcal{A}c'_\alpha$ est faite par induction sur α (le détail de ces constructions est donné dans la suite de la preuve). Définissons maintenant $Port'_\alpha = Port_\alpha$. Quant à l'ensemble I_0^α, il est construit à partir de At_α de la façon suivante : $I_0^\alpha = \{\neg p \mid p \in At_\alpha\}$. Concernant la construction de $\mathcal{A}c$, elle se fait à partir de Σ de la façon suivante.

1) Construction de $\mathcal{A}c$

L'automate communicant conditionnel $\mathcal{A}c$ est construit de façon à ne dépendre que de Σ. L'idée de cette construction est d'avoir, pour toute valeur possible de I_0^α, le langage reconnu par l'exécution de $\mathcal{A}c$ égal à Σ^*. Pour cela, nous construisons $\mathcal{A}c = (Q, q_0, F, \delta)$ tel que :

- $Q = \{q_0\}$,
- $F = \{q_0\}$ et
- $\delta = \{(\emptyset, q_0, a, q_0, \emptyset) \mid a \in \Sigma\}$.

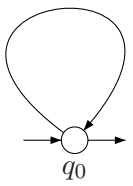

FIG. 6.2 – Automate communicant conditionnel $\mathcal{A}c$, lorsque $\Sigma = \{a_1, a_2, \ldots, a_m\}$.

En supposant que $\Sigma = \{a_1, a_2, \ldots, a_m\}$, l'automate communicant conditionnel $\mathcal{A}c$ est celui de la Figure 6.2. Pour tout At_α et $I_0^\alpha \subseteq At_\alpha$, l'automate $Exec(\mathcal{A}c, I_0^\alpha)$ est celui de la Figure 6.3. Cet automate contient un unique état accessible à partir de l'état initial. Cela est dû au fait que les préconditions et les effets des transitions de $\mathcal{A}c$ sont des ensembles vides. De plus, le fait que $\mathcal{A}c$ ne contient aucune action dans $Port_\alpha$ implique que :

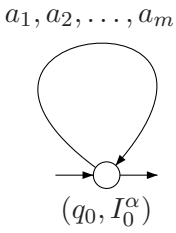

FIG. 6.3 – Automate fini $Exec(\mathcal{A}c, I_0^\alpha)$ pour l'automate communicant conditionnel $\mathcal{A}c$ de la Figure 6.2 et un I_0^α arbitraire.

$$Tr_{(\{!,?\}\times Port_\alpha)}(Exec(\mathcal{A}c, I_0^\alpha)) = Tr_\emptyset(Exec(\mathcal{A}c, I_0^\alpha)) = \Sigma^*.$$

2) Construction de $Port_\alpha$, At_α et $\mathcal{A}c'_\alpha$

La construction de $Port_\alpha$, At_α et $\mathcal{A}c'_\alpha$ se fait par induction sur α. La construction de $\mathcal{A}c'_\alpha$ est telle que :

$$Tr_{(\{!,?\}\times Port_\alpha)}(Exec(\mathcal{A}c'_\alpha \circledast \mathcal{A}c^{L_{Port_\alpha}}, I_0^\alpha)) = Tr_{(\{!,?\}\times Port_\alpha)}(Exec(\mathcal{A}c'_\alpha, I_0^\alpha))$$

et $Tr_{(\{!,?\}\times Port_\alpha)}(Exec(\mathcal{A}c'_\alpha, I_0^\alpha)) = L(\alpha)$.

Rappelons que les actions du médiateur sont non observables, car elle appartiennent à $\{!,?\}\times Port_\alpha$. Cependant, ces actions peuvent rendre possible l'exécution d'une action observable $a \in \Sigma$ exécutée par $\mathcal{A}c'_\alpha$. Par exemple, considérons l'automate $\mathcal{A}c'_\alpha$ de la Figure 6.4. Dans ce cas :

- $Tr_{(\{!,?\}\times Port_\alpha)}(Exec(\mathcal{A}c'_\alpha, I_0^\alpha)) = \emptyset$ et
- $Tr_{(\{!,?\}\times Port_\alpha)}(Exec(\mathcal{A}c'_\alpha \circledast \mathcal{A}c^{L_{Port_\alpha}}, I_0^\alpha)) = \{a\}$.

Ainsi, l'action non observable $!\pi$, exécutée par le médiateur large, a permis l'exécution de l'action observable a par $\mathcal{A}c'_\alpha$. Pour éviter des cas semblables à celui-ci, nous construisons $\mathcal{A}c'_\alpha$ tel que toutes ses actions de communications sont exécutables. Dans ce qui suit, nous notons par t_α le cardinal de l'ensemble $Port_\alpha$ et par s_α le cardinal de l'ensemble At_α.

Remarque 6.3.4. *Afin de ne pas encombrer les Figures 6.7, 6.8 et 6.9, les transitions étiquetées par $(\emptyset, \theta\pi, \emptyset)$ seront étiquetée par $\theta\pi$, où $\theta \in \{!,?\}$ et $\pi \in Port_\alpha$.*

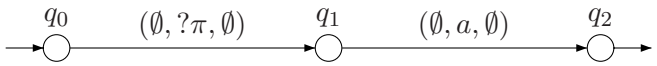

FIG. 6.4 – ACC $\mathcal{A}c'_\alpha$.

Base de l'induction : Cas où $\alpha = a$. Dans ce cas, on pose $Port_a = \emptyset$, $At_a = \emptyset$ et $\mathcal{A}c'_a = (Q^a, q_0^a, F^a, \delta^a)$ est l'automate communicant conditionnel de la Figure 6.5. Dans $\mathcal{A}c'_a$, l'ensemble des états finaux est un singleton $F^a = \{q_1^a\}$. Il est clair que :

$$Tr_{(\{!,?\}\times Port_a)}(Exec(\mathcal{A}c'_a \circledast \mathcal{A}c^{L_{Port_a}}, \emptyset)) = Tr_{(\{!,?\}\times Port_a)}(Exec(\mathcal{A}c'_a, \emptyset)).$$

La raison est que $\mathcal{A}c'_a$ ne contient aucune action de communication. De plus :

$$Tr_{(\{!,?\}\times Port_a)}(Exec(\mathcal{A}c'_a \circledast \mathcal{A}c^{L_{Port_a}}, \emptyset)) = L(a) = \{a\}.$$

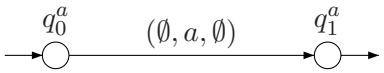

FIG. 6.5 – ACC $\mathcal{A}c'_\alpha$, dans le cas où $\alpha = a$.

Hypothèse de l'induction : considérons deux expressions régulières avec carré γ et σ telles qu'il existe des ensembles $Port_\gamma$, $Port_\sigma$, At_γ, At_σ et des ACC $\mathcal{A}c'_\gamma = (Q^\gamma, q_0^\gamma, F^\gamma, \delta^\gamma)$ sur $\Sigma \cup (\{!,?\} \times Port_\gamma)$ et $\mathcal{A}c'_\sigma = (Q^\sigma, q_0^\sigma, F^\sigma, \delta^\sigma)$ sur $\Sigma \cup (\{!,?\} \times Port_\sigma)$ tels que :

- $t_\gamma = Op(\gamma)$ et $t_\sigma = Op(\sigma)$,
- $s_\gamma = 2 \times Op(\gamma)$ et $s_\sigma = 2 \times Op(\sigma)$,
- $F^\gamma = \{q_1^\gamma\}$ et $F^\sigma = \{q_1^\sigma\}$,
- $Tr_{(\{!,?\}\times Port_\gamma)}(Exec(\mathcal{A}c'_\gamma \circledast \mathcal{A}c^{L_{Port_\gamma}}, I_0^\gamma)) = L(\gamma)$ et
- $Tr_{(\{!,?\}\times Port_\sigma)}(Exec(\mathcal{A}c'_\sigma \circledast \mathcal{A}c^{L_{Port_\sigma}}, I_0^\sigma)) = L(\sigma)$

Pas de l'induction : Cas où $\alpha = \epsilon$. Dans ce cas, on pose $Port_\epsilon = \{\pi_1\}$, $At_\epsilon = \{p_1, p_2\}$ et $\mathcal{A}c'_\epsilon = (Q^\epsilon, q_0^\epsilon, F^\epsilon, \delta^\epsilon)$ sur $\Sigma \cup (\{!,?\} \times Port_\epsilon)$

est l'automate communicant conditionnel de la Figure 6.6. Il est clair que :

$$Tr_{(\{!,?\}\times Port_\epsilon)}(Exec(\mathcal{A}c'_\epsilon \circledast \mathcal{A}c^{L_{Port_\epsilon}}, I_0^\epsilon)) = Tr_{(\{!,?\}\times Port_\epsilon)}(Exec(\mathcal{A}c'_\epsilon, I_0^\epsilon)).$$

La raison est que les transitions étiquetées par $!\pi_1$ et $?\pi_1$ sont exécutables dans $Exec(\mathcal{A}c'_\epsilon, I_0^\epsilon)$. De plus :
- $t_\epsilon = Op(\epsilon)$ avec $Op(\epsilon) = 1$,
- $s_\epsilon = 2 \times Op(\epsilon)$,
- $F^\epsilon = \{q_1^\epsilon\}$,
- $Tr_{(\{!,?\}\times Port_\epsilon)}(Exec(\mathcal{A}c'_\epsilon \circledast \mathcal{A}c^{L_{Port_\epsilon}}, I_0^\epsilon)) = L(\epsilon)=\{\epsilon\}$.

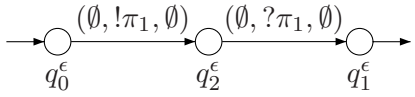

FIG. 6.6 – ACC $\mathcal{A}c'_\alpha$, dans le cas où $\alpha = \epsilon$.

Cas où $\alpha = \gamma \circ \sigma$. Dans ce cas, on pose $Port_{\gamma\circ\sigma} = Port_\gamma \cup Port_\sigma \cup \{\pi_{t_\gamma+t_\sigma+1}\}$, $At_{\gamma\circ\sigma} = At_\gamma \cup At_\sigma \cup \{p_{s_\gamma+s_\sigma+1}, p_{s_\gamma+s_\sigma+2}\}$ et $\mathcal{A}c'_{\gamma\circ\sigma}$ est l'automate communicant conditionnel de la Figure 6.7. Il est clair que :

$$Tr_{(\{!,?\}\times Port_{\gamma\circ\sigma})}(Exec(\mathcal{A}c'_{\gamma\circ\sigma} \circledast \mathcal{A}c^{L_{Port_{\gamma\circ\sigma}}}, I_0^{\gamma\circ\sigma})) = Tr_{(\{!,?\}\times Port_{\gamma\circ\sigma})}(Exec(\mathcal{A}c'_{\gamma\circ\sigma}, I_0^{\gamma\circ\sigma})).$$

La raison est que les transitions étiquetées par $!\pi_{t_\gamma+t_\sigma+1}$ et $?\pi_{t_\gamma+t_\sigma+1}$ sont exécutables dans $Exec(\mathcal{A}c'_{\gamma\circ\sigma}, I_0^{\gamma\circ\sigma})$. De plus :
- $t_{\gamma\circ\sigma} = t_\gamma + t_\sigma + 1$, ainsi $t_{\gamma\circ\sigma} = Op(\gamma \circ \sigma)$,
- $s_{\gamma\circ\sigma} = s_\gamma + s_\sigma + 2$, ainsi $s_{\gamma\circ\sigma} = 2 \times Op(\gamma \circ \sigma)$,
- $F^{\gamma\circ\sigma} = \{q_1^{\gamma\circ\sigma}\}$,
- $Tr_{(\{!,?\}\times Port_{\gamma\circ\sigma})}(Exec(\mathcal{A}c'_{\gamma\circ\sigma} \circledast \mathcal{A}c^{L_{Port_{\gamma\circ\sigma}}}, I_0^{\gamma\circ\sigma})) = L(\gamma \circ \sigma)=\{ww' \mid w \in L(\gamma)$ et $w' \in L(\sigma)\}$.

Cas où $\alpha = \gamma \cup \sigma$. Dans ce cas, on pose $Port_{\gamma\cup\sigma} = Port_\gamma \cup Port_\sigma \cup \{\pi_{t_\gamma+t_\sigma+1}\}$, $At_{\gamma\cup\sigma} = At_\gamma \cup At_\sigma \cup \{p_{s_\gamma+s_\sigma+1}, p_{s_\gamma+s_\sigma+2}\}$ et $\mathcal{A}c'_{\gamma\cup\sigma}$ est l'automate communicant conditionnel de la Figure 6.8. Les transitions étiquetées par $!\pi_{t_\gamma+t_\sigma+1}$ et $?\pi_{t_\gamma+t_\sigma+1}$ sont exécutables dans $Exec(\mathcal{A}c'_{\gamma\cup\sigma}, I_0^{\gamma\cup\sigma})$. Par conséquent :

$$Tr_{(\{!,?\}\times Port_{\gamma\cup\sigma})}(Exec(\mathcal{A}c'_{\gamma\cup\sigma} \circledast \mathcal{A}c^{L_{Port_{\gamma\cup\sigma}}}, I_0^{\gamma\cup\sigma})) = Tr_{(\{!,?\}\times Port_{\gamma\cup\sigma})}(Exec(\mathcal{A}c'_{\gamma\cup\sigma}, I_0^{\gamma\cup\sigma})).$$

De plus :
- $t_{\gamma\cup\sigma} = t_\gamma + t_\sigma + 1$, ainsi $t_{\gamma\cup\sigma} = Op(\gamma \circ \sigma)$,

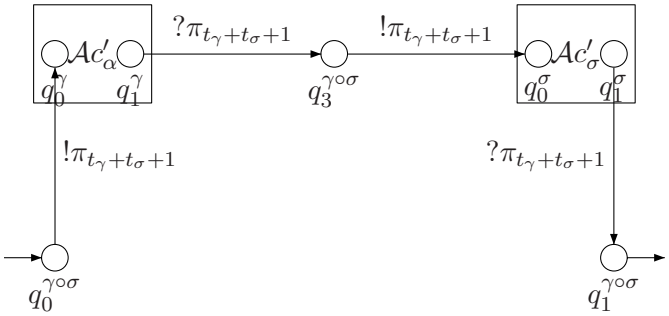

FIG. 6.7 – ACC $\mathcal{A}c'_\alpha$, dans le cas où $\alpha = \gamma \circ \sigma$.

- $s_{\gamma\cup\sigma} = s_\gamma + s_\sigma + 2$, ainsi $s_{\gamma\cup\sigma} = 2 \times Op(\gamma \cup \sigma)$,
- $F^{\gamma\cup\sigma} = \{q_1^{\gamma\cup\sigma}\}$,
- $Tr_{(\{!,?\}\times Port_{\gamma\cup\sigma})}(Exec(\mathcal{A}c'_{\gamma\cup\sigma} \circledast \mathcal{A}c^{L_{Port_{\gamma\cup\sigma}}}, I_0^{\gamma\cup\sigma})) = L(\gamma \cup \sigma) = \{w \mid w \in L(\gamma)$ ou $w \in L(\sigma)\}$.

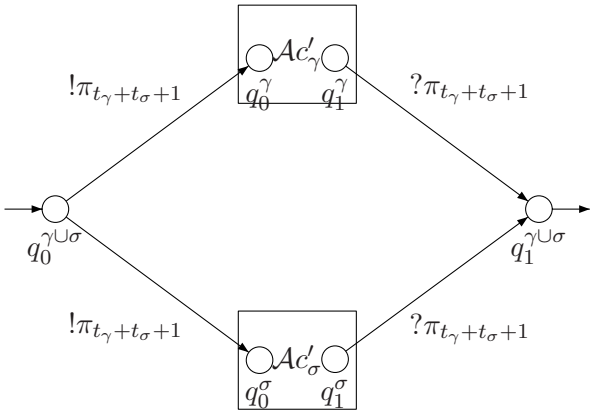

FIG. 6.8 – ACC $\mathcal{A}c'_\alpha$, dans le cas où $\alpha = \gamma \cup \sigma$.

Cas où $\alpha = \gamma^+$. Dans ce cas, on pose $Port_{\gamma^+} = Port_\gamma \cup \{\pi_{t_\gamma+1}\}$, $At_{\gamma^+} = At_\gamma \cup \{p_{s_\gamma+1}, p_{s_\gamma+2}\}$ et $\mathcal{A}c'_{\gamma^+}$ est l'automate communicant conditionnel de la Figure 6.9. Les transitions étiquetées par $!\pi_{t_\gamma+1}$ et $?\pi_{t_\gamma+1}$ sont exécutables dans $Exec(\mathcal{A}c'_{\gamma^+}, I_0^{\gamma^+})$. Par conséquent :

$$Tr_{(\{!,?\}\times Port_{\gamma^+})}(Exec(\mathcal{A}c'_{\gamma^+} \circledast \mathcal{A}c^{L_{Port_{\gamma^+}}}, I_0^{\gamma^+})) =$$
$$Tr_{(\{!,?\}\times Port_{\gamma^+})}(Exec(\mathcal{A}c'_{\gamma^+}, I_0^{\gamma^+})).$$

De plus :
- $t_{\gamma^+} = t_\gamma + 1$, ainsi $t_{\gamma^+} = Op(\gamma^+)$,
- $s_{\gamma^+} = s_\gamma + 2$, ainsi $s_{\gamma^+} = 2 \times Op(\gamma^+)$,
- $F^{\gamma^+} = \{q_1^{\gamma^+}\}$,
- $Tr_{(\{!,?\}\times Port_{\gamma^+})}(Exec(\mathcal{A}c'_{\gamma^+} \circledast \mathcal{A}c^{L_{Port_{\gamma^+}}}, I_0^{\gamma^+})) = L(\gamma^+)=\{w^n \mid n \geq 0 \text{ et } w \in L(\gamma)\}$.

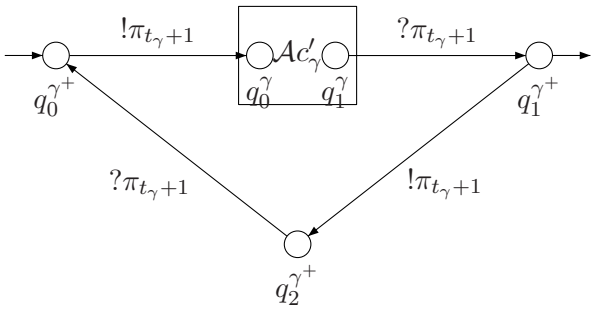

FIG. 6.9 – ACC $\mathcal{A}c'_\alpha$, dans le cas où $\alpha = \gamma^+$.

Cas où $\alpha = \gamma^2$. Dans ce cas, on pose $Port_{\gamma^2} = Port_\gamma \cup \{\pi_{t_\gamma+1}\}$, $At_{\gamma^2} = At_\gamma \cup \{p_{s_\gamma+1}, p_{s_\gamma+2}\}$ et $\mathcal{A}c'_{\gamma^2}$ est l'automate communicant conditionnel de la Figure 6.10. L'exécution de $\mathcal{A}c'_{\gamma^2}$ à partir de $I_0^{\gamma^2}$ est représentée dans la Figure 6.11 L'unique condition pour que les transitions étiquetées par $!\pi_{t_\gamma+1}$ et $?\pi_{t_\gamma+1}$ soient exécutables dans $Exec(\mathcal{A}c'_{\gamma^2}, I_0^{\gamma^2})$, est que les conditions sur les littéraux soient satisfaites. Sachant que le médiateur large ne peut pas modifier la valeur des littéraux alors :

$$Tr_{(\{!,?\}\times Port_{\gamma^2})}(Exec(\mathcal{A}c'_{\gamma^2} \circledast \mathcal{A}c^{L_{Port_{\gamma^2}}}, I_0^{\gamma^2})) = Tr_{(\{!,?\}\times Port_{\gamma^2})}(Exec(\mathcal{A}c'_{\gamma^2}, I_0^{\gamma^2})).$$

De plus :
- $t_{\gamma^2} = t_\gamma + 1$, ainsi $t_{\gamma^2} = Op(\gamma^2)$,
- $s_{\gamma^2} = s_\gamma + 2$, ainsi $s_{\gamma^2} = 2 \times Op(\gamma^2)$,
- $F^{\gamma^2} = \{q_1^{\gamma^2}\}$,
- $Tr_{(\{!,?\}\times Port_{\gamma^2})}(Exec(\mathcal{A}c'_{\gamma^2} \circledast \mathcal{A}c^{L_{Port_{\gamma^2}}}, I_0^{\gamma^2})) = L(\gamma^2)=\{w^2 \mid w \in L(\gamma)\}$.

Nous avons montré que $\mathcal{A}c$ et $\mathcal{A}c'_\alpha$ sont tels que $Tr_{(\{!,?\}\times Port_\alpha)}(Exec(\mathcal{A}c'_\alpha \circledast \mathcal{A}c^{L_{Port_\alpha}}, I_0^\alpha)) = L(\alpha)$ et $Tr_{(\{!,?\}\times Port_\alpha)}(Exec(\mathcal{A}c, I_0^\alpha)) = \Sigma^\star$. Sachant que $L(\alpha) \subseteq \Sigma^\star$ alors il est évident que :

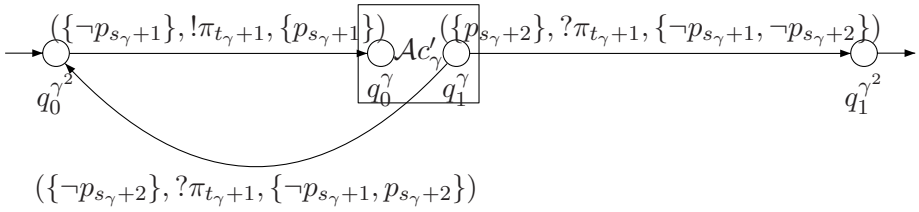

FIG. 6.10 – ACC $\mathcal{A}c'_\alpha$, dans le cas où $\alpha = \gamma^2$.

$L(\alpha) = \Sigma^\star$ ssi $Exec(\mathcal{A}c, I_0^\alpha) \subseteq_{tr} Exec(\mathcal{A}c'_\alpha \circledast \mathcal{A}c^{L_{Port_\alpha}}, I_0^\alpha)(\{!, ?\} \times Port')$.

Il nous reste à montrer que la réduction ρ est calculable par une machine de Turing déterministe, en utilisant un espace logarithmique. Considérons la machine de Turing $\mathcal{M}$ qui calcule ρ de la façon suivante :

(a) La machine $\mathcal{M}$ écrit d'abord sur son ruban de sortie l'ensemble $\Sigma' = \Sigma$. Elle écrit ensuite l'ensemble $\{q_0\}$, contenant l'unique état de $\mathcal{A}c$. Afin d'écrire les transitions de $\mathcal{A}c$ la machine utilise un ruban de travail contenant un compteur pour les éléments de Σ.

(b) Ensuite, la machine va écrire les ensembles $Port_\alpha$, At_α et I_0^α, la machine utilise deux compteurs. Le premier compteur a comme valeur maximale le le codage en base 2 de $s_\alpha = 2 \times Op(\alpha)$. Quant au second compteur, il a comme valeur maximale le le codage en base 2 de $t_\alpha = Op(\alpha)$.

(c) Pour finir, la machine va écrire les éléments de $\mathcal{A}c'_\alpha$. La fonction de transition de $\mathcal{A}c'_\alpha$ est l'élément qui utilise le plus d'espace dans les rubans de travail. Afin d'écrire la fonction de transition δ^α, la machine $\mathcal{M}$ utilise sept rubans de travail. A savoir, un premier ruban sera dédié à un compteur pour les états [2] de $\mathcal{A}c'_\alpha$. Un deuxième ruban sera dédié à un compteur pour les ports. Un troisième ruban sera dédié à un compteur pour les formules atomiques. Un quatrième ruban permettra

[2] Il est facile de vérifier que la valeur maximale que peut atteindre ce compteur est bornée par un polynôme en fonction du nombre d'opérations dans α. Bien entendu cette valeur maximale sera codée en base 2.

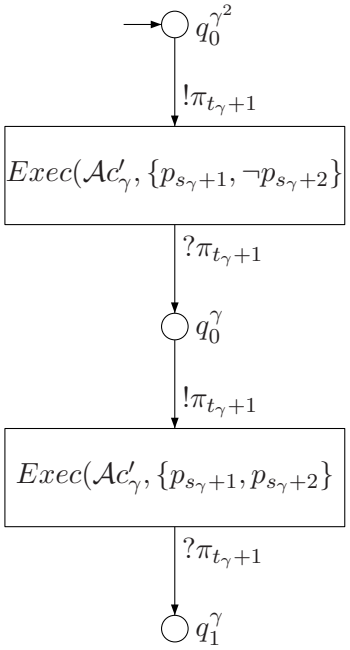

FIG. 6.11 – L'exécution de $\mathcal{A}c'_{\gamma^2}$ à partir de $I_0^{\gamma^2}$.

de mémoriser l'état initial et l'état final de la dernière opération lue dans le ruban d'entrée. Cela permettra d'écrire les transitions entre les états générés par la dernière opération et les états qui sont générés par l'opération en cours de lecture (dans le ruban d'entrée). De la même façon, un cinquième ruban permettra de mémoriser les deux ports générés par l'opération en cours de lecture (sur le ruban d'entrée). Un sixième ruban permettra de mémoriser les deux formules atomiques générés par l'opération en cours de lecture (sur le ruban d'entrée). Un septième ruban permettra de garder en mémoire tous les états générés par l'opération courante.

Toutes ces étapes peuvent être réalisées de façon déterministe en utilisant un espace logarithmique. □

6.3.2 Simulation

Pour montrer que le problème $\mathcal{PBC}(\leq_{si})$ est EXPTIME-difficile, nous utilisons des résultats connus sur le problème de la simulation entre un automate fini est un produit asynchrone d'automates finis. Formellement, le problème est défini comme suit :

Problème : problème de la simulation d'un automate par un produit asynchrone d'automates

Instance : un ensemble fini d'actions Σ, des automates finis déterministes $\mathcal{A}, \mathcal{B}^1, \ldots, \mathcal{B}^n$ sur Σ.

Question : est-ce-que $\mathcal{A} \leq_{si} \mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n(\emptyset)$?

Il a été prouvé dans [MW08] que ce problème est EXPTIME-difficile. Le théorème suivant se base sur ce résultat.

Théorème 6.3.5. *Le problème de la compostion $\mathcal{PBC}(\leq_{si})$ est EXPTIME-difficile.*

Démonstration. Rappelons que les automates finis sont un cas particulier d'automates communicants conditionnels[3]. Soit $\mathcal{A}, \mathcal{B}^1, \ldots, \mathcal{B}^n$ une instance du problème de la simulation d'un automate par un produit asynchrone d'automates sur l'alphabet Σ. Nous, considérons l'instance suivante du problème $\mathcal{PBC}(\leq_{si})$:

- un ensemble fini d'actions $\Sigma' = \Sigma$,
- un ensemble de Port $Port = \emptyset$,
- un ensemble de formules atomiques $At = \emptyset$,
- un automate communicant conditionnel Ac^{client} tel que $\delta^{client} = \emptyset$,
- un automate communicant conditionnel $\mathcal{A}c^{but} = Auto^{-1}(\mathcal{A}^1)$ et
- les automates $\mathcal{A}c^1 = Auto^{-1}(B^1), \ldots, \mathcal{A}c^n = Auto^{-1}(\mathcal{B}^n)$.

Il est clair que cette réduction est calculable en espace logarithmique. De plus,

$$\mathcal{A} \leq_{si} B^1 \otimes \ldots \otimes B^n$$
ssi
il existe un service médiateur $\mathcal{A}c^{med}$ sur $\emptyset$ et At tel que
$$Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, \emptyset)$$
$$\leq_{si}$$
$$Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast \mathcal{A}c^{med}, \emptyset).$$

[3]Pour plus de détail voir la remarque 3.2.7.

En effet, l'implication de gauche à droite est triviale. Supposons que $\mathcal{A} \leq_{si} B^1 \otimes \ldots \otimes B^n$. Dans ce cas, pour tous les services médiateurs $\mathcal{A}c^{med}$ sur $\emptyset$ et At, on a $Exec(\mathcal{A}c^{but}, \emptyset) \leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast \mathcal{A}c^{med}, \emptyset)$. Sachant que $\delta^{client} = \emptyset$, alors $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, \emptyset) \leq_{si} Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast \mathcal{A}c^{med}, \emptyset)$. Concernant l'implication de droite à gauche, s'il existe un ACC $\mathcal{A}c^{med}$ sur $\emptyset$ et At tel que $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, \emptyset) \leq_{si} Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast \mathcal{A}c^{med}, \emptyset)$ alors $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, \emptyset) \leq_{si} Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, \emptyset)$. Sachant que $\delta^{client} = \emptyset$ alors $Exec(\mathcal{A}c^{but}, \emptyset) \leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, \emptyset)$. Par conséquent, $\mathcal{A} \leq_{si} B^1 \otimes \ldots \otimes B^n$. □

6.3.3 Bisimulation et équivalence de traces

Dans le Chapitre 4, nous avons représenté les services par des automates communicants conditionnels qui communiquent à travers des ports non bornés. Dans ce cas, afin de prouver l'indécidabilité du problème de la composition pour la bisimulation (resp. l'équivalence de traces), nous avons utilisé des résultats d'indécidabilité concernant le problème de la bisimulation (resp. équivalence de traces) entre réseaux de Petri. Cependant, dans [JM96] les auteurs ont montré que le problème de la bisimulation (resp. équivalence de traces) entre réseaux de Petri devient décidable, lorsque les réseaux 1-saufs sont considérés. Plus précisément, ils ont montré que le problème devient EXPTIME-difficile pour la bisimulation et EXPSPACE-difficile pour l'équivalence de traces. En se basant sur ces résultats et en utilisant la réduction proposée dans le Chapitre 4, nous montrons que lorsque les ACC communiquent à travers des ports qui contiennent au plus un message, le problème de la composition est EXPTIME-difficile pour la bisimulation et EXPSPACE-difficile pour l'équivalence de traces.

Théorème 6.3.6. *Le problème de la compostion $\mathcal{PBC}(\longleftrightarrow_{bi})$ est EXPTIME-difficile.*

Démonstration. Afin de prouver que le problème de la bisimulation entre les réseaux de Petri se réduit au problème de la composition, dans le cas ou les ports sont non bornés, nous avons utilisé les résultats des lemmes 4.2.20, 4.2.22 et 4.2.25. De la même façon, pour prouver que le problème de la bisimulation entre réseaux de Petri 1-saufs se réduit au problème $\mathcal{PBC}(\longleftrightarrow_{bi})$, nous utilisons ces trois lemmes. Cependant, nous considérons maintenant des réseaux de Petri 1-saufs. Aussi, nous considérons des ports contenant au plus 1 message au lieu de ports non

bornés. Sachant que le problème de la bisimulation entre réseaux de Petri 1-saufs est EXPTIME-difficile, cette réduction nous permet de déduire que le problème $\mathcal{PBC}(\longleftrightarrow_{bi})$ est EXPTIME-difficile. Plus clairement, le lemme 4.2.20 concerne la réduction du problème simplifié de la composition vers le problème général de la composition. Il est trivial de constater que ce lemme reste vrai, dans le cas particulier des ACC qui communiquent à travers des ports bornés. Ensuite, le lemme 4.2.22 concerne l'isomorphisme entre $Exec(AutoR(\mathcal{N}))$ et $Exec(\mathcal{N}, m_0)$ modulo la communication[4]. Dans le cas où $\mathcal{N}$ est un réseau de Petri 1-sauf et où $\mathcal{A}c$ est un ACC qui communique à travers des ports contenant au plus un message, ce résultat reste vrai. Rappelons que dans le cas où les ports contiennent au plus un message, pour un ACC $\mathcal{A}c$ et un ensemble I_0, on ne peut exécuter l'action $!\pi$, dans $Exec(\mathcal{A}c, I_0)$ si π contient un message. Cette restriction est simulée dans les réseaux de Petri 1-saufs par le fait que le tir d'une transition t ne peut pas être effectué si une des places dans $t^\bullet$ contient déjà un jeton [Mur89]. Quant au lemme 4.2.25, il utilise les résultats des lemmes précédents pour montrer que le problème de la bisimulation entre les réseaux de Petri se réduit au problème simplifié de la composition. Cette preuve reste vrai lorsque des réseaux de Petri 1-saufs et des ports bornés sont considérés. □

Théorème 6.3.7. *Le problème de la compostion $\mathcal{PBC}(\equiv_{tr})$ est EXPSPACE-difficile.*

Démonstration. Il suffit d'utiliser l'approche proposée dans la preuve du théorème 6.3.6. Sachant que le problème de l'équivalence de traces entre deux réseaux de Petri 1-saufs est EXPSPACE-difficile [JM96], on obtient que $\mathcal{PBC}(\equiv_{tr})$ est également EXPSPACE-difficile. □

6.4 Bornes supérieures de complexité : simulation et inclusion de traces

Dans ce qui suit, nous montrons que le problème de la composition, lorsque les ports contiennent au plus un message, peut être résolue en temps exponentiel pour la simulation, en espace exponentiel pour l'inclusion de traces et en temps doublement exponentiel pour la bisimulation. Malheureusement, nous ne savons pas si, pour l'équivalence de traces, le problème de la composition est décidable. En d'autre terme, nous n'avons pas de procédure pour le résoudre.

[4]Pour plus de détail concernant le ACC $AutoR(\mathcal{N})$, voir la définition 4.2.21.

En ce qui concerne les problèmes $\mathcal{PBC}(\leq_{si})$ et $\mathcal{PBC}(\subseteq_{tr})$, afin de les résoudre, nous utilisons des approches basées sur l'utilisation du médiateur large et des procédures existantes pour la résolution des problèmes de simulation [HS96] et d'inclusion de traces [GJ90] entre automates finis.

Ainsi que nous le verrons dans la Section 6.5, cette approche ne peut pas être adaptée pour la résolution du problème $\mathcal{PBC}(\longleftrightarrow_{bi})$, nous proposons une méthode basée sur la synthèse de contrôleur pour le résoudre. Plus précisément, nous utilisons, pour la résolution du problème de la synthèse de contrôleur, une méthode basée sur la satisfaction d'une formule du μ-calcul et une méthode basée sur la filtration. La méthode que nous proposons pour réduire le problème $\mathcal{PBC}(\longleftrightarrow_{bi})$ vers le problème de la synthèse de contrôleurs est également applicable pour $\mathcal{PBC}(\equiv_{tr})$. Cependant, le problème obtenu est un problème de synthèse de contrôleurs avec masques [Tsi89]. Ce dernier est connu pour être PSPACE-difficile. Néanmoins, nous ignorons si ce problème est décidable.

6.4.1 Procédure basée sur l'utilisation du médiateur large pour la résolution de $\mathcal{PBC}(\leq_{si})$ et $\mathcal{PBC}(\subseteq_{tr})$

Dans le Chapitre 4, dans le lemme 4.2.5, nous avons réduit le problème $\mathcal{PC}(\sqsubseteq)$ au problème $\mathcal{PEL}(\sqsubseteq)$, avec $\sqsubseteq \in \{\leq_{si}, \subseteq_{tr}\}$. Pour prouver cette réduction nous avons montré :

il existe un service médiateur $\mathcal{A}c^{med}$ sur $\{!,?\} \times Port$ et At tel que $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, I_0) \sqsubseteq Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast \mathcal{A}c^{med}, I_0)(\{!,?\} \times Port')$ **ssi** $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, I_0) \sqsubseteq Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast Ac^{L_{Port}}, I_0)\ (\{!,?\} \times Port')$.

Ce résultat nous permet d'établir que l'algorithme ci-dessous est correct et complet. Cet algorithme a, comme entrées, les actions Σ, un ensemble de ports $Port$, un ensemble non observables de ports $Port' \subseteq Port$, un service client $\mathcal{A}c^{client}$, un service but $\mathcal{A}c^{but}$ et des services disponibles $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$. La procédure intitulée Résoudre-$\mathcal{PC}(\leq_{si})$ retourne "Vrai" **ssi** il existe un service médiateur tel que le système composé du service client et du service but est simulé par le système composé du service client, des services de la communauté et du médiateur. Dans la ligne 2, la fonction $LargeMed$ retourne le médiateur large associé à l'ensemble $Port$. Cette opération peut être effectuée en un temps **linéaire** par rapport à la taille de $Port$. Dans la ligne 3, l'algorithme calcule un automate fini $\mathcal{A}$ à partir de $\mathcal{A}c^{client}, \mathcal{A}c^{but}$ et I_0. Cet automate est d'une taille **exponentielle** par rapport à la taille

Algorithm 2 Algorithme pour la résolution de $\mathcal{PBC}(\leq_{si})$

1: **Resoudre-**$\mathcal{PC}(\leq_{si})(\Sigma, Port, Port', At, I_0, \mathcal{A}c^{client}, \mathcal{A}c^{but}, \mathcal{A}c1, \ldots, \mathcal{A}c^n)$
2: $Ac^{L_{Port}} \leftarrow LargeMed(Port)$
3: $\mathcal{A} \leftarrow Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, I_0)$
4: $\mathcal{A}c \leftarrow \mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast \mathcal{A}c^{L_{Port}}$
5: $\mathcal{A}' \leftarrow Exec(\mathcal{A}c, I_0)$
6: **if** $\mathcal{A} \leq_{si} \mathcal{A}' \quad (\{!, ?\} \times Port')$ **then retourner** Vrai
7: **else retourner** Faux
8: **end if**
9: **end procedure**

de I_0 et de $Port$. Dans la ligne 4, l'opération qui calcule $\mathcal{A}c$ peut être effectuée en un temps **exponentiel** par rapport au nombre n des services disponibles. Comme pour la taille de $\mathcal{A}$, la taille de $\mathcal{A}'$, construit dans la ligne 5, est **exponentielle** par rapport à la taille de I_0 et de $Port$. Dans la ligne 6, le temps nécessaire pour déterminer si $\mathcal{A} \leq_{si} \mathcal{A}' \quad (\{!, ?\} \times Port')$ est **polynomial** relativement à la taille des automates finis $\mathcal{A}$ et $\mathcal{A}'$ [HS96]. Par conséquent, afin de retourner une solution, l'algorithme prend un temps **exponentiel** par rapport à la taille de son entrée. Ainsi, nous obtenons le résultat suivant.

Théorème 6.4.1. *Le problème de la compostion $\mathcal{PBC}(\leq_{si})$ est dans EXPTIME.*

La méthode utilisée pour résoudre $\mathcal{PBC}(\leq_{si})$ peut être adaptée pour résoudre $\mathcal{PBC}(\subseteq_{tr})$. Plus précisément, un algorithme pour résoudre $\mathcal{PBC}(\subseteq_{tr}$) est obtenu en modifiant uniquement la ligne 6, de l'algorithme 2. Pour

Algorithm 3 Algorithme pour résoudre $\mathcal{PBC}(\subseteq_{tr})$

1: **Resoudre-**$\mathcal{PC}(\subseteq_{tr})(\Sigma, Port, Port', At, I_0, \mathcal{A}c^{client}, \mathcal{A}c^{but}, \mathcal{A}c1, \ldots, \mathcal{A}c^n)$
2: $Ac^{L_{Port}} \leftarrow LargeMed(Port)$
3: $\mathcal{A} \leftarrow Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, I_0)$
4: $\mathcal{A}c \leftarrow \mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast \mathcal{A}c^{L_{Port}}$
5: $\mathcal{A}' \leftarrow Exec(\mathcal{A}c, I_0)$
6: **if** $\mathcal{A} \subseteq_{tr} \mathcal{A}' \quad (\{!, ?\} \times Port')$ **then retourner** Vrai
7: **else retourner** Faux
8: **end if**
9: **end procedure**

retourner une solution, l'algorithme 3 utilise un espace **exponentiel** par rapport à la taille de I_0 et de $Port$ et au nombre n de services disponibles. Les raisons sont les suivantes. (1) L'opération de la ligne 6 nécessite un espace **polynomial** [GJ90]. (2) La taille de $\mathcal{A}$ et de $\mathcal{A}'$ est **exponentielle** par rapport à la taille de I_0 de $Port$ et au nombre n. Par conséquent,

Théorème 6.4.2. *Le problème de la compostion $\mathcal{PBC}(\subseteq_{tr})$ est dans EXPSPACE.*

6.5 Bornes supérieures de complexité : Bisimulation

6.5.1 Discussion

Sachant que le problème de la bisimulation entre automates finis est dans PTIME, on peut se poser la question suivante : est-il possible de modifier la ligne 6 de l'algorithme 2, afin de résoudre le problème $\mathcal{PBC}(\longleftrightarrow_{bi})$ en temps exponentiel ? La réponse est : l'algorithme obtenu sera correct mais pas complet. En d'autre termes, si l'algorithme retourne "Faux" cela n'implique pas que la réponse au problème $\mathcal{PBC}(\longleftrightarrow_{bi})$ est "Faux". C'est le cas de l'exemple suivant.

Exemple 6.5.1. *Considérons un ensemble d'actions $\Sigma = \{a, b\}$, un ensemble de ports $Port = Port' = \{\pi_1\}$, un ensemble vide de formule atomique, un ACC $\mathcal{A}c^{client}$ tel que $\delta^{client} = \emptyset$, le ACC $\mathcal{A}c^{but}$ représenté dans la Figure 6.12 et un unique service disponible $\mathcal{A}c^1$, également représenté dans la Figure 6.12. Il est facile de vérifier que $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, \emptyset)$ n'est pas bisimilaire à $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \mathcal{A}c^{L_{Port}}, \emptyset)$ modulo $\{!, ?\} \times Port'$. Par conséquent, l'algorithme retournera la valeur "Faux". Cependant, si nous considérons le service médiateur tel que $\delta^{med} = \emptyset$ alors $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, \emptyset)$ est bisimilaire à $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \mathcal{A}c^{med}, \emptyset)$ modulo $\{!, ?\} \times Port'$. En effet, $\mathcal{A}c^{med}$ ne transmet pas de messages sur le port π_1. Par conséquent, l'action b ne peut pas être exécutée dans $Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \mathcal{A}c^{med}, \emptyset)$.*

Pour proposer un algorithme correct et complet, permettant la résolution du problème $\mathcal{PBC}(\longleftrightarrow_{bi})$, nous utilisons une approche basée sur la synthèse de contrôleurs. Ainsi, dans ce qui suit nous proposons une réduction du problème de la composition vers le problème de la synthèse de contrôleur.

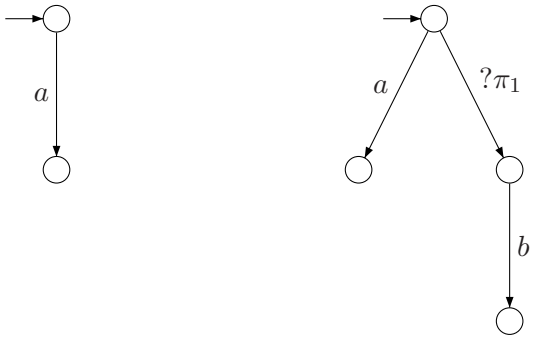

FIG. 6.12 – De gauche à droite, le ACC $\mathcal{A}c^{but}$ et le ACC Ac^1.

Ensuite, afin de résoudre le problème de la synthèse de contrôleurs obtenu, nous proposons deux procédures. D'une part, nous proposons une procédure basée sur la satisfaction d'une formule du μ-calcul. D'autre part, nous proposons une procédure basée sur la filtration.

6.5.2 Synthèse de contrôleurs

Contrôler un système signifie lui adjoindre un système nommé contrôleur qui peut lui interdire certaines actions, afin que le produit synchrone des deux systèmes[5] satisfasse certaines propriétés [RW89].

L'idée de l'approche que nous proposons est de contrôler le système composé du service client, des services disponibles et du médiateur large afin d'obtenir un système qui soit bisimilaire au système composé du service client et du service but. Sachant que le service client et les services disponibles ne sont pas contrôlables, l'objectif est de contrôler le médiateur large afin d'avoir un nouveau service médiateur pour lequel les deux systèmes sont bisimilaires. Pour cela, il est nécessaire, d'une part de distinguer les actions contrôlables de celles qui ne le sont pas et d'autre part, les actions observables de celles qui ne le sont pas.

Rappelons quelques concepts de la théorie de la synthèse de contrôleurs. Pour définir les contraintes de contrôlabilité (notés C_{ctr}) et d'observabilité

[5]Le produit synchrone représente le comportement du système initial lorsqu'il est contrôlé.

(noté C_{obs}) que le contrôleur doit satisfaire il est nécessaire de partitionner l'ensemble des actions. Soit Σ un ensemble fini d'actions. Cet ensemble est partitionné en un ensemble d'actions contrôlables noté Σ_{ct} et un ensemble d'actions non-contrôlables noté Σ_{nct}. De plus, Σ est partitionné en un ensemble d'actions observables noté Σ_{ob} et un ensemble d'actions non-observables noté Σ_{nob}.

Définition 6.5.2 (Contraintes de contrôlabilité et d'observabilité). Soit $\mathcal{C}$ un automate fini sur Σ. Pour cet automate, les *contraintes d'observabilité* (notées C_{obs}) et les *contraintes de contrôlabilité* (notées C_{ctr}), paramétrées par les partitions $\Sigma_{ct} \uplus \Sigma_{nct}$ et $\Sigma_{ob} \uplus \Sigma_{nob}$, sont définies comme suit :

C_{ctr} : pour tout état q dans $\mathcal{C}$ et pour toute action non-contrôlable $a \in \Sigma_{nct}$, il existe une transition à partir de q étiquetée par a.

: C_{obs}pour tout les états q dans $\mathcal{C}$ et pour toute action non-observable $a \in \Sigma_{nob}$, s'il existe une transition à partie de q étiquetée par a alors cette transition boucle sur q.

Le contrôleur satisfait C_{ctr} ssi il réagit à toute action non-contrôlable et il satisfait C_{obs} ssi il ne peut pas détecter l'occurrence des actions non-observables.

Le problème de la synthèse de contrôleur qui nous intéresse utilise une fonction appelée masque [Tsi89]. Cette fonction est un renommage des actions dans Δ par des actions dans l'ensemble $\Sigma \cup \{\epsilon\}$ qui lui est disjoint. Considérons le problème suivant :

Problème $\mathcal{CS}$: problème de la synthèse de contrôleurs

Instance : des ensembles finis d'actions Σ et Δ, un automate fini $\mathcal{S}$ sur Σ et un automate fini $\mathcal{G}$ sur Δ, une fonction $Masque : \Delta \to \Sigma \cup \{\epsilon\}$ et des contraintes C_{obs} et C_{ctr}.

Question : existe-il un automate $\mathcal{C}$ sur Δ qui satisfait C_{obs} et C_{ctr} et tel que : $\mathcal{S} \longleftrightarrow_{bi} Masque(\mathcal{G} \times \mathcal{C})\ (\{\epsilon\})$?

Le problème du contrôle tel qu'il est présenté ici à été étudié par [ZKJ06, Tsi89]. Plus précisément, dans [ZKJ06] les auteurs considèrent que $\Delta = \Sigma$ et dans [Tsi89] ils considèrent l'équivalence de traces au lieu de la bisimulation[6]. Pour être résolu ce problème peut être réduit à un problème de contrôle avec une formule du μ-calcul à satisfaire. Il peut également être résolu en utilisant une méthode basée sur la filtration.

[6] Considérer l'équivalence de trace revient à considérer la bisimulation avec des automates déterministes.

Dans ce qui suit, nous donnons quelques définitions et résultats préliminaires. C'est résultats sont utilisées pour prouver que les algorithmes que nous proposons sont corrects et complets.

6.5.3 Définitions et résultats préliminaires

Rappelons la description du problème $\mathcal{PBC}(\longleftrightarrow_{bi})$:

Instance : les ensembles finis Σ, $Port$ et $Port' \subseteq Port$, At et $I_0 \subseteq Li(At)$, un automate communicant conditionnel $\mathcal{A}c^{client}$ sur $\{!,?\} \times Port$ et At et des automates communicants conditionnels finis $\mathcal{A}c^{but}$, $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At.

Question : existe-t-il un service médiateur $\mathcal{A}c^{med}$ sur $\{!,?\} \times Port$ et At, tel que

$$Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, I_0) \longleftrightarrow_{bi} Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast \mathcal{A}c^{med}, I_0)(\{!,?\} \times Port') \,?$$

Une description simplifiée de ce problème serait la suivante :

Instance : les ensembles finis Σ, $Port$ et $Port' \subseteq Port$, At et $I_0 \subseteq Li(At)$, des automates conditionnels finis $\mathcal{A}c$ et $\mathcal{A}c'$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At.

Question : existe-t-il un service médiateur $\mathcal{A}c^{med}$ sur $\{!,?\} \times Port$ et At, tel que

$$Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Exec(\mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0)(\{!,?\} \times Port') \,?$$

Il est clair que si nous arrivons à résoudre ce problème alors nous saurons résoudre le problème $\mathcal{PBC}(\longleftrightarrow_{bi})$. En effet, il suffit juste de remplacer $\mathcal{A}c$ par $\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}$ et $\mathcal{A}c'$ par $\mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n$. Bien-entendu, la taille de $\mathcal{A}c'$ est exponentielle par rapport à n et à la taille de At. Notre objectif est de réduire le problème ci-dessus au problème $\mathcal{CS}$.

Intuitivement, l'idée de la réduction est de considérer $\mathcal{A}c$ comme étant la spécification et $\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port}}$ comme étant le système à contrôler. Plus précisément, les actions contrôlables et observables sont celles effectuées par le médiateur large. Afin de distinguer les actions de communication exécutées par le médiateur large de celles exécutées par $\mathcal{A}c'$ nous associons à tout ensemble $Port$ un ensemble de copies $Port^\circ$. Ce dernier représente l'ensemble de ports que le médiateur large utilise. L'ensemble $Port^\circ$ est constitué des

éléments renommés de $Port$. Formellement, $Port^\circ = \{\pi^\circ \mid \pi \in Port\}$. Par conséquent, le médiateur large sera $\mathcal{A}c^{L_{Port^\circ}}$, Σ_{ct}=$\{!,?\} \times Port^\circ$ et Σ_{ob}=$\{!,?\} \times Port^\circ$.

Remarque 6.5.3. *Sachant que $\Sigma_{ct} = \Sigma_{ob}$, alors au lieu de dire qu'un automate satisfait C_{obs} et C_{ctr} nous dirons que cette automate boucle sur Σ_{nct}.*

Dans la réduction que nous proposons, nous utilisons les fonctions Del° et $Exec^\circ$.

Sachant que dans $\mathcal{A}c$ la communication s'effectue à travers les ports dans $Port$ et que dans $\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port}}$ la communication s'effectue à travers les ports dans $Port \cup Port^\circ$ alors, avant de vérifier la bisimulation entre $\mathcal{A}c$ et $\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port}}$ une fois contrôlé, il faut supprimer le renommage des ports. Ainsi, pour un automate $\mathcal{A}$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^\circ))$, on associe un automate $Del^\circ(\mathcal{A})$. Dans ce dernier, on ne peut plus distinguer les communications effectuées par le médiateur large de celles effectuées par les autres services. Formellement, la fonction Del° est définie comme suit :

Définition 6.5.4 (**Fonction** Del°)**.** Soit $\mathcal{A} = (Q, q_0, \rightarrow)$ un automate sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^\circ))$. $Del^\circ(\mathcal{A}) = (Q', q'_0, \rightarrow')$ est un automate sur $\Sigma \cup (\{!,?\} \times Port)$ tel que :

- $Q' = Q$,
- $q'_0 = q_0$,
- $\rightarrow' \subseteq Q' \times (\Sigma \cup (\{!,?\} \times Port)) \times Q'$ est la relation de transitons définie par $q \rightarrow^{\{a\}}_{Del^\circ(\mathcal{A})} q'$ ssi une des deux conditions suivantes est satisfaite :
  1. $a \in \Sigma$ et $q \rightarrow^{\{a\}}_{\mathcal{A}} q'$,
  2. $a = \theta\pi$ et $q \rightarrow^{\{\theta\pi\}}_{\mathcal{A}} q'$ ou $q \rightarrow^{\{\theta\pi^\circ\}}_{\mathcal{A}} q'$, avec $\theta \in \{!,?\}$.

Concernant la fonction $Exec^\circ$, l'unique différence entre cette fonction et la fonction $Exec$ est que dans $Exec^\circ$ l'exécution de $!\pi^\circ$ et $?\pi^\circ$ permet de mettre à jour le port π au lieu du port π°. Formellement, la fonction $Exec^\circ$ est définie comme suit :

Définition 6.5.5 (**Fonction** $Exec^\circ$)**.** Soit I_0 un ensemble de littéraux et $\mathcal{A}c = (Q, q_0, \delta)$ un automate communicant conditionnel sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^\circ))$ et At. $Exec^\circ(\mathcal{A}c, I_0) = (Q', q'_0, \rightarrow')$ est un automate sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^\circ))$ tel que :

- $Q' = \{(q, \gamma, I) \mid q \in Q, \gamma : Port \rightarrow \mathbb{N}$ et $I \subseteq Li(At)$ est une partie maximale consistante$\}$,

- $q'_0 = (q_0, \gamma_0, I_0)$ où $\forall \pi \in Port, \gamma_0(\pi) = 0$,
- $\rightarrow' \subseteq Q' \times \Sigma \cup (\{!,?\} \times (Port \cup Port^\circ)) \times Q'$ est la relation de transition définie par $((q,\gamma,I),a,(q',\gamma',I')) \in \rightarrow'$ ssi il existe une transition $(J,q,a,q',J') \in \delta$ 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 l'une des cinq conditions suivantes est satisfaite :
  1. $a \in \Sigma$ et $\gamma = \gamma'$,
  2. $a = ?\pi$, $\gamma(\pi) = 1$, $\gamma'(\pi) = 0$ et $\gamma'(\pi') = \gamma(\pi')$ pour $\pi' \neq \pi$
  3. $a = !\pi$, $\gamma(\pi) = 0$, $\gamma'(\pi) = 1$ et $\gamma'(\pi') = \gamma(\pi')$ pour $\pi' \neq \pi$
  4. $a = ?\pi^\circ$, $\gamma(\pi) = 1$, $\gamma'(\pi) = 0$ et $\gamma'(\pi') = \gamma(\pi')$ pour $\pi' \neq \pi$
  5. $a = !\pi^\circ$, $\gamma(\pi) = 0$, $\gamma'(\pi) = 1$ et $\gamma'(\pi') = \gamma(\pi')$ pour $\pi' \neq \pi$.

Le théorème suivant nous permet de réaliser la réduction du problème simplifié de $\mathcal{PBC}(\longleftrightarrow_{bi})$ vers le problème de la synthèse de contrôleurs $\mathcal{CS}$.

Théorème 6.5.6. *Soient les ACC $\mathcal{A}c$ et $\mathcal{A}c'$ sur $\Sigma \cup (\{!,?\} \times Port)$. Les conditions suivantes sont équivalentes :*

(1) Il existe un un service médiateur $\mathcal{A}c^{med}$ sur $\{!,?\} \times Port$ et At tel que :
$Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Exec(\mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0) \quad (\{!,?\} \times Port')$.

(2) Il existe un automate fini $\mathcal{C}$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^\circ))$ qui boucle sur $\Sigma \cup (\{!,?\} \times Port)$ et tel que :
$Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Del^\circ(Exec^\circ(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0) \times \mathcal{C}) \quad (\{!,?\} \times Port')$.

Démonstration. Voir l'annexe 2. □

Il est facile de constater que le premier point du théorème représente l'objectif du problème de la composition. Quant au second point, il correspond à l'objectif du problème de la synthèse de contrôleur.

Remarque 6.5.7. *Le théorème 6.5.6 reste vrai pour la relation d'équivalence de traces. En effet, dans la preuve de ce dernier nous montrons que $Exec(\mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0)$ et $Del^\circ(Exec^\circ(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0) \times \mathcal{C})$ sont isomorphes.*

6.5.4 Procédure basée sur le μ-calcul pour la résolution de $\mathcal{PBC}(\longleftrightarrow_{bi})$

Le μ-calcul [Koz97] a émergé grâce à des travaux reconnus dans le domaine de la logique et de l'informatique. Il est utilisé pour l'analyse et la

vérification de propriétés sur des programmes informatiques. Formellement, les propriétés à exprimer sont représentées par une formule du μ-calcul, le programme informatique est représenté par un automate et vérifier les propriétés sur le programme revient à vérifier si l'automate satisfait la formule du μ-calcul. Afin de résoudre le problème de la composition pour la relation de bisimulation, on s'intéresse à exprimer par une formule si un automate fini $\mathcal{A}$ est bisimilaire, modulo un ensemble d'actions de communication non observables, à un automate fini $Del^{\circ}(\mathcal{A}')$. Intuitivement, l'automate $\mathcal{A}$ représente le système composé du service but et du service client et l'automate $\mathcal{A}'$ représente le système composé des services disponibles, du médiateur large et du contrôleur. Nous pouvons montrer que vérifier la bisimulation, modulo un ensemble d'action de communications, entre ces deux automates revient à vérifier si l'automate $\mathcal{A}'$ satisfait une formule donnée du μ-calcul. Cette formule est construite à partir de l'automate $\mathcal{A}$ et de l'ensemble des actions de communications non observables. Dans l'annexe 1 se trouve des rappels sur la syntaxe et la sémantique du μ-calcul. Ainsi, que la preuve détaillée du lemme suivant.

Lemme 6.5.8. *Soient les ensembles $Port$ et $Port' \subseteq Port$ et $\mathcal{A}$ un automate sur $\Sigma \cup (\{!,?\} \times Port)$ et At, il est possible de construire en un temps polynomial une formule du μ-calcul $\varphi(\mathcal{A}, Port')$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ de taille polynomiale telle que pour tout automate fini $\mathcal{A}'$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$, on a :*

$$\mathcal{A} \longleftrightarrow_{bi} Del^{\circ}(\mathcal{A}') \quad (\{!,?\} \times Port') \text{ ssi } \mathcal{A}' \models \varphi(\mathcal{A}, Port').$$

Considérons le problème suivant :

Problème $\mathcal{CSM}$: problème de la synthèse de contrôleurs pour le μ-calcul

Instance : un ensemble fini d'actions Σ, une formule du μ-calcul ϕ sur Σ, un automate fini $\mathcal{G}$ sur Σ et des contraintes C_{obs} et C_{ctr}

Question : existe-il un automate $\mathcal{C}$ sur Σ qui satisfait C_{obs} et C_{ctr} et tel que : $\mathcal{G} \times C \models \phi$?

Ce problème à été étudié dans [AVW03]. Les auteurs ont proposé une solution pour ce problème. Plus précisément, ils ont proposé une procédure qui permet de construire un contrôleur. Cependant, dans cette procédure le système à contrôler est déterministe. Il a été prouvé dans [PR05a] que la complexité du problème considéré dans [AVW03] et la même que celle du problème de la synthèse de contrôleurs avec un automate non déterministe à contrôler. De plus, le problème de la synthèse de contrôleurs avec un système

à contrôler non déterministe à été étudié dans [BK06]. Dans ce cas, la complexité du problème reste exponentielle. En se basant sur le lemme 6.5.8 nous prouvons le théorème suivant.

Théorème 6.5.9. *Soient les ACC $\mathcal{A}c$ et $\mathcal{A}c'$ sur $\Sigma \cup (\{!,?\} \times Port)$. Les conditions suivantes sont équivalentes :*

(1) Il existe un automate fini $\mathcal{C}$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ qui boucle sur $\Sigma \cup (\{!,?\} \times Port)$ et tel que $Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Del^{\circ}(Exec^{\circ}(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port^{\circ}}}, I_0) \times \mathcal{C}) \quad (\{!,?\} \times Port')$.

(2) Il existe un automate fini $\mathcal{C}$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ qui boucle sur $\Sigma \cup (\{!,?\} \times Port)$ et tel que $Exec^{\circ}(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port^{\circ}}}, I_0) \times \mathcal{C} \models \varphi(Exec(\mathcal{A}c, I_0), Port')$.

Démonstration. Il suffit d'appliquer le lemme 6.5.8 en replaçant $\mathcal{A}$ par $Exec(\mathcal{A}c, I_0)$ et $\mathcal{A}'$ par $Exec^{\circ}(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port^{\circ}}}, I_0) \times \mathcal{C}$. □

A partir des théorèmes 6.5.6 et 6.5.9, en considérant que $\mathcal{A}c = \mathcal{A}c^{client} \circledast \mathcal{A}c^{but}$ et $\mathcal{A}c' = \mathcal{A}c^{client} \circledast \mathcal{A}c^{1} \circledast \ldots \circledast \mathcal{A}c^{n}$, nous obtenons que l'algorithme suivant, pour la résolution de $\mathcal{PC}(\longleftrightarrow_{bi})$, est correct et complet. Dans cet

Algorithm 4 Algorithme pour la résolution de $\mathcal{PBC}(\longleftrightarrow_{bi})$

Resoudre$(\Sigma, Port, Port', At, I_0, \mathcal{A}c^{client}, \mathcal{A}c^{but}, \mathcal{A}c1, \ldots, \mathcal{A}c^{n})$
$\mathcal{A} \leftarrow Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, I_0)$
$\phi \leftarrow \varphi(\mathcal{A}, Port')$
$\mathcal{A}c^{L_{Port^{\circ}}} \leftarrow LargeMed(Port^{\circ})$
$\mathcal{A}c' \leftarrow \mathcal{A}c^{client} \circledast \mathcal{A}c^{1} \circledast \ldots \circledast \mathcal{A}c^{n} \circledast \mathcal{A}c^{L_{Port^{\circ}}}$
$\mathcal{G} \leftarrow Exec^{\circ}(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port^{\circ}}}, I_0)$
$\Sigma_{nct} \leftarrow \Sigma \cup (\{!,?\} \times Port)$
$\Sigma_{nob} \leftarrow \Sigma \cup (\{!,?\} \times Port)$
if $(Controler(\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ})), \Sigma_{nct}, \Sigma_{nob}, \phi, \mathcal{G}) =$ Vrai**) then return** Vrai
else return Faux
end if
end procedure

algorithme, pour l'ensemble des actions Σ, Σ', Σ_{nct} et Σ_{nob}, une formule sur Σ et un automate fini $\mathcal{G}$, la fonction $Controler(\Sigma, \Sigma_{nct}, \Sigma_{nob}, \phi, \mathcal{G})$ retourne "Vrai" **ssi** il existe un contrôleur $\mathcal{C}$ sur Σ qui satisfait C_{obs} et C_{ctr} et tel que $\mathcal{G} \times \mathcal{C} \models \phi$. Le même raisonnement que celui utilisé pour déterminer le temps nécessaire pour retourner une solution pour l'Algorithme 2, page 148, peut être utilisé pour déterminer le temps nécessaire pour l'Algorithme 4. Dans

la ligne 2, la construction de $\mathcal{A}$ nécessite un temps **exponentiel** par rapport à la taille de $Port$ et de I_0. De plus la taille de $\mathcal{A}$ est **exponentielle** par rapport à la taille des ensembles $Port$ et I_0. Dans la ligne 3, la construction de ϕ (l'existence d'une telle fonction ϕ est déduite du lemme 6.5.8) nécessite un temps **polynomial** par rapport à la taille de $\mathcal{A}$ et de $Port'$. Dans la ligne 5, le temps nécessaire pour le calcul de $\mathcal{A}c'$ est **exponentiel** par rapport à n. Quant à la construction de $\mathcal{G}$, dans la ligne 6, elle est effectuée en un temps **exponentiel** par rapport à la taille de $Port$ et de I_0 et **polynomial** par rapport à la taille de $\mathcal{A}c'$. Sachant que les tailles de $\mathcal{G}$ et de ϕ sont **exponentielles** par rapport au nombre n, à la taille de $Port$ et de I_0 et que la fonction $Controler$ nécessite un temps **exponentiel** relativement à la taille de ses entrées [AVW03, PR05a, BK06], par conséquent, l'Algorithme 4 retourne une solution en un temps **doublement exponentiel** relativement à la taille de $Port$, de I_0 et au nombre n. Ainsi, nous obtenons le résultat suivant.

Théorème 6.5.10. *Le problème de la compostion $\mathcal{PBC}(\longleftrightarrow_{bi})$ est dans 2-EXPTIME.*

6.5.5 Procédure basée sur la filtration pour la résolution de $\mathcal{PBC}(\longleftrightarrow_{bi})$

Dans cette section, nous proposons une approche qui consiste à identifier des états d'un automate pour en réduire la taille. Cette approche est inspirée d'une technique bien connue en logique modale (la filtration) utilisée pour montrer la propriété du modèle fini [BdMRV01].

Notre objectif est d'utiliser cette approche pour la résolution du problème $\mathcal{PBC}(\longleftrightarrow_{bi})$. Cependant, cette approche ne peut être utilisée que dans le cas où $Port' = \emptyset$. Considérons le problème de décision suivant :

Problème $\mathcal{FIL}$: problème de la filtration

Instance : un ensemble fini d'actions Σ, un ensemble de ports $Port$, des automates finis $\mathcal{A}$ sur $\Sigma \cup (\{!,?\} \times Port)$ et $\mathcal{A}'$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$,

Question : existe-il un automate $\mathcal{C}$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ qui boucle sur $\Sigma \cup (\{!,?\} \times Port)$ et tel que :

$$\mathcal{A} \longleftrightarrow_{bi} Del^{\circ}(\mathcal{A}' \times \mathcal{C})\,?$$

Dans un premier temps nous proposons un algorithme en temps exponentiel pour la résolution du problème $\mathcal{FIL}$. Nous prouverons également que cet algorithme est correct et complet. Ensuite, grâce aux résultats du

théorème 6.5.6, nous proposons un algorithme correct et complet pour la résolution du problème $\mathcal{PBC}(\longleftrightarrow_{bi})$. Cet algorithme a la même complexité que l'algorithme 4. En d'autre terme, cet algorithme retourne une solution en un temps doublement exponentiel relativement à la taille de $Port$, de I_0 et au nombre n. Considérons les ensembles Σ et $Port$ et les automates $\mathcal{A} = (Q^{\mathcal{A}}, q_0^{\mathcal{A}}, \rightarrow_{\mathcal{A}})$ sur $\Sigma \cup (\{!,?\} \times Port)$, $\mathcal{A}' = (Q^{\mathcal{A}'}, q_0^{\mathcal{A}'}, \rightarrow_{\mathcal{A}'})$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$, et $\mathcal{C} = (Q^{\mathcal{C}}, q_0^{\mathcal{C}}, \rightarrow_{\mathcal{C}})$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ qui boucle sur $\Sigma \cup (\{!,?\} \times Port)$ tels qu'il existe une bisimulation Z entre $\mathcal{A}$ et $Del^{\circ}(\mathcal{A}' \times \mathcal{C})$ tel que $q_0^{\mathcal{A}}\ Z\ (q_0^{\mathcal{A}'}, q_0^{\mathcal{C}})$.

Définition 6.5.11 (**Relation binaire** $\equiv$). *La relation* $\equiv \subseteq Q^{\mathcal{C}} \times Q^{\mathcal{C}}$ *est la relation binaire telle que :*

- $q_1^{\mathcal{C}} \equiv q_2^{\mathcal{C}}$ *ssi pour tout* $q^{\mathcal{A}} \in Q^{\mathcal{A}}$ *et pour tout* $q^{\mathcal{A}'} \in Q^{\mathcal{A}'}$, $q^{\mathcal{A}} Z (q^{\mathcal{A}'}, q_1^{\mathcal{C}})$ *ssi* $q^{\mathcal{A}} Z (q^{\mathcal{A}'}, q_2^{\mathcal{C}})$

Notons que $\equiv$ est une relation d'équivalence. Soit $q^{\mathcal{C}} \in Q^{\mathcal{C}}$. L'ensemble de tout les états dans $Q^{\mathcal{C}}$ équivalent à $q^{\mathcal{C}}$ modulo $\equiv$, noté $\mid q^{\mathcal{C}} \mid$, est appelé classe d'équivalence de $q^{\mathcal{C}}$ dans $Q^{\mathcal{C}}$ modulo $\equiv$ avec $q^{\mathcal{C}}$ comme représentant. L'ensemble de toutes les classes d'équivalences de $Q^{\mathcal{C}}$ modulo $\equiv$, noté $Q^{\mathcal{C}}/\equiv$, est appelé l'ensemble quotient de $Q^{\mathcal{C}}$ modulo $\equiv$.

Définition 6.5.12 (**Automate obtenu par filtration** $\mathcal{C}^f$). *La filtration de* $\mathcal{C}$ *relativement à* $\mathcal{A}$ *et* $\mathcal{A}'$ *est l'automate* $\mathcal{C}^f = (Q^{\mathcal{C}^f}, q^{\mathcal{C}^f}_0, \rightarrow_{\mathcal{C}^f})$ *sur* $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ *qui boucle sur* $\Sigma \cup (\{!,?\} \times Port)$ *et tel que :*

- $Q^{\mathcal{C}^f} = Q^{\mathcal{C}}/\equiv$,
- $q_0^{\mathcal{C}^f} = \mid q_0^{\mathcal{C}} \mid$ *et*
- $\rightarrow_{\mathcal{C}^f}$ *est la relation telle que :*
  - $\mid q_1^{\mathcal{C}} \mid \rightarrow_{\mathcal{C}^f}^{\{\theta\pi^{\circ}\}} \mid q_2^{\mathcal{C}} \mid$, $\theta \in \{!,?\}$ *et* $\pi^{\circ} \in Port^{\circ}$, *ssi pour tout* $q_1^{\mathcal{A}} \in Q^{\mathcal{A}}$ *et pour tout* $q_1^{\mathcal{A}'}$, $q_2^{\mathcal{A}'} \in Q^{\mathcal{A}'}$ *si* $q_1^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{\theta\pi^{\circ}\}} q_2^{\mathcal{A}'}$ *et* $q_1^{\mathcal{A}} Z (q_1^{\mathcal{A}'}, q_1^{\mathcal{C}})$ *alors il existe* $q_2^{\mathcal{A}} \in Q^{\mathcal{A}}$ *tel que* $q_1^{\mathcal{A}} \rightarrow_{\mathcal{A}}^{\{\theta\pi\}} q_2^{\mathcal{A}}$ *et* $q_2^{\mathcal{A}} Z (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$.

Définition 6.5.13 (**La relation** Z^f **obtenue par filtration**). *Soit* $Z^f \subseteq Q^{\mathcal{A}} \times (Q^{\mathcal{A}'} \times Q^{\mathcal{C}^f})$ *la relation binaire telle que pour tout* $q^{\mathcal{A}} \in Q^{\mathcal{A}}$ *et pour tout* $(q^{\mathcal{A}'}, \mid q^{\mathcal{C}} \mid) \in Q^{\mathcal{A}'} \times Q^{\mathcal{C}^f}$, $q^{\mathcal{A}} Z^f (q^{\mathcal{A}'}, \mid q^{\mathcal{C}} \mid)$ *ssi* $q^{\mathcal{A}} Z (q^{\mathcal{A}'}, q^{\mathcal{C}})$.

Il est facile de vérifier que Z^f est une relation de bisimulation entre $\mathcal{A}$ et $Del^{\circ}(\mathcal{A}' \times \mathcal{C})$. Pour plus de détails, voir l'annexe 3.

Proposition 6.5.14. *Les propriétés suivantes sont équivalentes.*

- *il existe un automate fini* $\mathcal{C} = (Q^{\mathcal{C}}, q_0^{\mathcal{C}}, \rightarrow_{\mathcal{C}})$ *sur* $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ *qui boucle sur* $\Sigma \cup (\{!,?\} \times Port)$ *et il existe une relation* $Z \subseteq Q^{\mathcal{A}} \times (Q^{\mathcal{A}'} \times Q^{\mathcal{C}})$ *qui est une bisimulation entre* $\mathcal{A}$ *et* $Del^{\circ}(\mathcal{A}' \times \mathcal{C})$.

– *il existe un automate fini* $\mathcal{C} = (Q^{\mathcal{C}}, q_0^{\mathcal{C}}, \rightarrow_{\mathcal{C}})$ *sur* $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ *qui boucle sur* $\Sigma \cup (\{!,?\} \times Port)$ *et il existe une relation* $Z \subseteq Q^{\mathcal{A}} \times (Q^{\mathcal{A}'} \times Q^{\mathcal{C}})$ *qui est une bisimulation entre* $\mathcal{A}$ *et* $Del^{\circ}(\mathcal{A}' \times \mathcal{C})$ *tels que les conditions suivantes sont vérifiées :*
(C_1) $Q^{\mathcal{C}} \subseteq 2^{Q^{\mathcal{A}} \times Q^{\mathcal{A}'}}$,
(C_2) $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{\theta\pi^{\circ}\}} q_2^{\mathcal{C}}$ *et* $\theta \in \{!,?\}$ *ssi pour tout* $q^{\mathcal{A}} \in Q^{\mathcal{A}}$ *et pour tout* $q_1^{\mathcal{A}'}, q_2^{\mathcal{A}'} \in Q^{\mathcal{A}'}$, *si* $q_1^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{\theta\pi^{\circ}\}} q_2^{\mathcal{A}'}$ *et* $(q_1^{\mathcal{A}}, q_1^{\mathcal{A}'}) \in q_1^{\mathcal{C}}$ *alors il existe* $q_2^{\mathcal{C}} \in Q^{\mathcal{C}}$ *tel que* $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{\theta\pi^{\circ}\}} q_2^{\mathcal{C}}$ *et* $(q_2^{\mathcal{A}}, q_2^{\mathcal{A}'}) \in q_2^{\mathcal{C}}$,
(C_3) $q^{\mathcal{A}} Z (q^{\mathcal{A}'}, q^{\mathcal{C}})$ *ssi* $(q^{\mathcal{A}}, q^{\mathcal{A}'}) \in q^{\mathcal{C}}$

Démonstration. Considérons l'implication de gauche à droite. Supposons qu'il existe un automate fini $\mathcal{C} = (Q^{\mathcal{C}}, q_0^{\mathcal{C}}, \rightarrow_{\mathcal{C}})$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ qui boucle sur $\Sigma \cup (\{!,?\} \times Port)$ et il existe une relation $Z \subseteq Q^{\mathcal{A}} \times (Q^{\mathcal{A}'} \times Q^{\mathcal{C}})$ qui est une bisimulation entre $\mathcal{A}$ et $Del^{\circ}(\mathcal{A}' \times \mathcal{C})$. Il suffit de construire $\mathcal{C}^f$ à partir de $\mathcal{C}$ comme dans la définition 6.5.12 et Z^f à partir de Z comme dans la définition 6.5.13. Il est facile de montrer que $\mathcal{C}^f$ vérifie (C_1), (C_2) et (C_3). En effet, par construction de $Q^{\mathcal{C}^f}$, chaque état $q^{\mathcal{C}^f}$ représente la classe d'équivalence de $q^{\mathcal{C}}$ dans $Q^{\mathcal{C}}$ modulo $\equiv$. A partir de la définition de $\equiv$, on obtient $q^{\mathcal{C}^f} = \{(q^{\mathcal{A}}, q^{\mathcal{A}'}) \in Q^{\mathcal{A}} \times Q^{\mathcal{A}'} \mid q^{\mathcal{A}} Z (q^{\mathcal{A}'}, q^{\mathcal{C}})\}$. Ainsi la condition (C_1) est satisfaite par $Q^{\mathcal{C}^f}$. Sachant que pour tout $q^{\mathcal{A}} \in Q^{\mathcal{A}}$ et pour tout $q_1^{\mathcal{A}'} \in Q^{\mathcal{A}'}$, $(q^{\mathcal{A}}, q^{\mathcal{A}'}) \in q^{\mathcal{C}^f}$ ssi $q^{\mathcal{A}} Z (q^{\mathcal{A}'}, q^{\mathcal{C}})$ alors, à partir de la construction de $\mathcal{C}^f$ et de Z^f, on obtient que les conditions (C_2) et (C_3) sont satisfaites.

L'implication de droite à gauche est triviale. En effet, la deuxième propriété contient la première. □

Ainsi, nous pouvons proposer un algorithme simple pour la résolution de $\mathcal{FIL}$. Dans l'algorithme 5, la fonction $Condition1(q^{\mathcal{A}}, q^{A'}, q^{\mathcal{C}}, \theta, \pi^{\circ})$ retourne "vrai" ssi il existe un état $q_2^{\mathcal{A}} \in Q^{\mathcal{A}}$ tel que $q^{\mathcal{A}} \rightarrow_{\mathcal{A}}^{\{\theta\pi\}} q_2^{\mathcal{A}}$ et pour tout $q_2^{\mathcal{A}'} \in Q^{\mathcal{A}'}$ tel que $q^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{\theta\pi^{\circ}\}} q_2^{\mathcal{A}'}$ et pour tout $q_2^{\mathcal{C}} \in Q^{\mathcal{C}}$ tel que $q^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{\theta\pi^{\circ}\}} q_2^{\mathcal{C}}$, $q_2^{\mathcal{A}} Z (q^{\mathcal{A}'}, q^{\mathcal{C}})$ n'est pas vérifié. La fonction $Condition2(q^{\mathcal{A}}, q^{A'}, q^{\mathcal{C}}, \theta, \pi^{\circ})$ retourne "vrai" ssi il existe $q_2^{\mathcal{A}'} \in Q^{\mathcal{A}'}$ tel que $q^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{\theta\pi^{\circ}\}} q_2^{\mathcal{A}'}$ et il existe $q_2^{\mathcal{C}} \in Q^{\mathcal{C}}$ tel que $q^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{\theta\pi^{\circ}\}} q_2^{\mathcal{C}}$ et pour tout $q_2^{\mathcal{A}} \in Q^{\mathcal{A}}$ tel que $q^{\mathcal{A}} \rightarrow_{\mathcal{A}}^{\{\theta\pi\}} q_2^{\mathcal{A}}$, $q_2^{\mathcal{A}} Z (q^{\mathcal{A}'}, q^{\mathcal{C}})$ est vérifié. Intuitivement, l'ensemble Δ regroupe tous les états dans $Q^{\mathcal{C}}$ qui n'ont pas permis à Z d'être une bisimulation de $\mathcal{A}$ par $Del^{\circ}(\mathcal{A}' \times \mathcal{C})$. De plus, la mise à jour de $Q^{\mathcal{C}}$ consiste à supprimer ces états. L'algorithme 5, peut être implanté en un temps exponentiel par rapport à la taille de $\mathcal{A}$ et de $\mathcal{A}'$. En effet, le nombre de fois que la boucle $While$ est exécutée est au plus égale à $2^{Card(Q^{\mathcal{A}}) \times Card(Q^{\mathcal{A}'})}$. De plus les opérations à l'intérieur de la boucle sont obtenues en un temps polynomial relativement

Algorithm 5 Algorithme pour la résolution de $\mathcal{FIL}$

Resoudre-$\mathcal{FIL}(\Sigma, Port, \mathcal{A}, \mathcal{A}')$
$Q^{\mathcal{C}} \leftarrow 2^{Q^{\mathcal{A}} \times Q^{\mathcal{A}'}}$
$Fin \leftarrow Faux$
$(\rightarrow_{\mathcal{C}}) \leftarrow \{q^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{a\}} q^{\mathcal{C}} \mid q^{\mathcal{C}} \in Q^{\mathcal{C}}$ et $a \in \Sigma \cup (\{!,?\} \times Port)\} \cup \{ q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{\theta\pi^\circ\}} q_2^{\mathcal{C}} \mid \theta \in \{!,?\}, \pi^\circ \in Port^\circ$ et la condition C_2 est satisfaite$\}$
while ($Fin = Faux$ et $Q^{\mathcal{C}} \neq \emptyset$) **do**
  $Z \leftarrow \{(q^{\mathcal{A}}, (q^{\mathcal{A}'}, q^{\mathcal{C}})) \mid (q^{\mathcal{A}}, q^{\mathcal{A}'}) \in q^{\mathcal{C}}\}$
  if Z est une bisimution de $\mathcal{A}$ par $Del^\circ(\mathcal{A}' \times \mathcal{C})$ **then** $Fin \leftarrow Vrai$
  else
    $\Delta \leftarrow \{q^{\mathcal{C}} \in Q^{\mathcal{C}} \mid$ il existe $q^{\mathcal{A}} \in Q^{\mathcal{A}}$ et $q^{\mathcal{A}'} \in Q^{\mathcal{A}'}$ avec $q^{\mathcal{A}} Z (q^{\mathcal{A}'}, q^{\mathcal{C}})$ et il existent $\theta \in \{!,?\}$ et $\pi^\circ \in Port^\circ$ tels que $Condition1(q^{\mathcal{A}}, q^{A'}, q^{\mathcal{C}}, \theta, \pi^\circ)$ ou $Condition2(q^{\mathcal{A}}, q^{A'}, q^{\mathcal{C}}, \theta, \pi^\circ)\}$
    $Q^{\mathcal{C}} \leftarrow Q^{\mathcal{C}} \setminus \Delta$
    $(\rightarrow_{\mathcal{C}}) \leftarrow (\rightarrow_{\mathcal{C}}) \setminus \{q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{\theta\pi^\circ\}} q_2^{\mathcal{C}} \mid q_1^{\mathcal{C}} \in \Delta$ ou $q_2^{\mathcal{C}} \in \Delta\}$
  end if
end while
end procedure

à la taille de $Q^{\mathcal{C}}$ (qui est bornée par $2^{Card(Q^{\mathcal{A}}) \times Card(Q^{\mathcal{A}'})}$). Le lemme suivant nous permet de prouver que l'algorithme 5, proposé pour la résolution de $\mathcal{FIL}$, est correct et complet. En d'autre terme, le lemme montre que l'algorithme retourne la valeur "Vrai" **ssi** il existe un automate fini $\mathcal{C}$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^\circ))$ qui boucle sur $\Sigma \cup (\{!,?\} \times Port)$ et tel que $\mathcal{A} \longleftrightarrow_{bi} Del^\circ(\mathcal{A}' \times \mathcal{C})$. Nous notons m le nombre d'itération de la boucle **while**, Q^{C^m} les état de $\mathcal{C}$ à l'itération m et Z^m la relation Z obtenu à l'itération m.

Lemme 6.5.15. *Soit $\mathcal{C} = (Q^{\mathcal{C}}, q_0^{\mathcal{C}}, \rightarrow_{\mathcal{C}})$ sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^\circ))$ qui boucle sur $\Sigma \cup (\{!,?\} \times Port)$ et $Z \subseteq Q^{\mathcal{A}} \times (Q^{\mathcal{A}'} \times \mathcal{C})$ une bisimulation entre $\mathcal{A}$ et $Del^\circ(\mathcal{A}' \times \mathcal{C})$ tels que les conditions suivantes sont vérifiées :*

(C_1) $Q^{\mathcal{C}} \subseteq 2^{Q^{\mathcal{A}} \times Q^{\mathcal{A}'}}$,

(C_2) $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{\theta\pi^\circ\}} q_2^{\mathcal{C}}$ et $\theta \in \{!,?\}$ ssi pour tout $q^{\mathcal{A}} \in Q^{\mathcal{A}}$ et pour tout $q_1^{\mathcal{A}'}, q_2^{\mathcal{A}'}$, si $q_1^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{\theta\pi^\circ\}} q_2^{\mathcal{A}'}$ et $(q_1^{\mathcal{A}}, q_1^{\mathcal{A}'}) \in q_1^{\mathcal{C}}$ alors il existe $q_2^{\mathcal{C}} \in Q^{\mathcal{C}}$ tel que $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{\theta\pi^\circ\}} q_2^{\mathcal{C}}$ et $(q_2^{\mathcal{A}}, q_2^{\mathcal{A}'}) \in q_2^{\mathcal{C}}$,

(C_3) $q^{\mathcal{A}} Z (q^{\mathcal{A}'}, q^{\mathcal{C}})$ ssi $(q^{\mathcal{A}}, q^{\mathcal{A}'}) \in q^{\mathcal{C}}$

Alors, pour tout $q^{\mathcal{C}} \in Q^{\mathcal{C}}$ et pour tout entier non-négatif m, $q^{\mathcal{C}} \in Q^{\mathcal{C}^m}$.

Démonstration. Soit $q^{\mathcal{C}} \in Q^{\mathcal{C}}$. S'il existe un entier non-négatif m tel que $q^{\mathcal{C}}$

$\in {Q_{\mathcal{C}}}^m$ et $q_{\mathcal{C}} \notin {Q_{\mathcal{C}}}^{m+1}$ alors pour un état $q^{\mathcal{A}} \in Q^{\mathcal{A}}$ et un état $q^{\mathcal{A}'} \in Q^{\mathcal{A}'}$ tel que $q^{\mathcal{A}}\ Z^m\ (q^{\mathcal{A}'}, q^{\mathcal{C}})$, il existe $\theta \in \{!, ?\}$ et $\pi^\circ \in Port^\circ$ tels que l'une des propriétés suivantes est vraie :

- il existe $q_2^{\mathcal{A}} \in Q^{\mathcal{A}}$ tel que $q_1^{\mathcal{A}} \rightarrow_{\mathcal{A}}^{\{\theta\pi\}} q_2^{\mathcal{A}}$ et pour tout $q_2^{\mathcal{A}'} \in Q^{\mathcal{A}'}$ et $q_2^{\mathcal{C}} \in Q^{\mathcal{C}}$ tels que $q_1^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{\theta\pi\circ\}} q_2^{\mathcal{A}'}$ et pour tout $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{\theta\pi^\circ\}} q_2^{\mathcal{C}}$, il n'est pas vrai que $q_2^{\mathcal{A}} Z^m (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$,
- il existe $q_2^{\mathcal{A}'} \in Q^{\mathcal{A}'}$ et $q_2^{\mathcal{C}} \in Q^{\mathcal{C}}$ tels que $q_1^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{\theta\pi\circ\}} q_2^{\mathcal{A}'}$, $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{\theta\pi^\circ\}} q_2^{\mathcal{C}}$ et pour tout $q_2^{\mathcal{A}} \in Q^{\mathcal{A}}$ tel que $q_1^{\mathcal{A}} \rightarrow_{\mathcal{A}}^{\{\theta\pi\}} q_2^{\mathcal{A}}$, il n'est pas vrai que $q_2^{\mathcal{A}} Z^m (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$.

Les deux cas mènent à une contradiction avec la propriété (C_2). □

A partir du théorème 6.5.6, nous obtenons que l'algorithme suivant, pour la résolution de $\mathcal{PC}(\longleftrightarrow_{bi})$, est correct et complet. Il est clair que la taille

Algorithm 6 Algorithme pour la résolution de $\mathcal{PBC}(\longleftrightarrow_{bi})$

1: **Résoudre-**$\mathcal{PBC}(\longleftrightarrow_{bi})(\Sigma, Port, At, I_0, \mathcal{A}c^{client}, \mathcal{A}c^{but}, \mathcal{A}c1, \ldots, \mathcal{A}c^n)$
2: $\mathcal{A} \leftarrow Exec(\mathcal{A}c^{client} \circledast \mathcal{A}c^{but}, I_0)$
3: $\mathcal{A}c^{L_{Port^\circ}} \leftarrow LargeMed(Port^\circ)$
4: $\mathcal{A}c' \leftarrow \mathcal{A}c^{client} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n \circledast \mathcal{A}c^{L_{Port^\circ}}$
5: $\mathcal{A}' \leftarrow Exec^\circ(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0)$
6: **if** $(Resoudre - \mathcal{FIL}(\Sigma, Port, \mathcal{A}, \mathcal{A}') =$ Vrai$)$ **then return** Vrai
7: **else return** Faux
8: **end if**
9: **end procedure**

de $\mathcal{A}$ et $\mathcal{A}'$ est exponentielle par rapport à n et à la taille de I_0 et $Port$. Sachant que la procédure $Resoudre - \mathcal{FIL}$ retourne une solution après un temps exponentiel relativement à la taille de son entrée, on en déduit que l'algorithme 6 utilise un temps doublement exponentiel par rapport à la taille des entrées du problème $\mathcal{PBC}(\longleftrightarrow_{bi})$.

6.6 Conclusion

Nous avons présenté un modèle dans lequel les services sont représentés par des automates communicants conditionnels. Ces derniers communiquent à travers des ports k-bornés, c'est à dire qu'un port ne peut contenir qu'au plus k messages. Nous avons défini le problème de la composition pour l'inclusion de traces, l'équivalence de traces, la simulation et la bisimulation.

Relation	Complexity
Trace inclusion	*EXPSPACE-complet*
Trace equivalence	*EXPSPACE-difficile*
Simulation	*EXPTIME-complet*
Bisimulation	*EXPTIME-difficile* *2-EXPTIME*

TAB. 6.1 – Tableau récapitulatif, dans le cas où la communication entre les services est asynchrone avec des ports bornés.

Les résultats de la complexité des problèmes sont récapitulés dans le Tableau 6.1.
Concernant le problème de la composition, en considérant l'inclusion de traces et la simulation, nous avons donné la complexité exacte 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 traces et EXPTIME-difficile pour la bisimulation.
L'approche que nous avons utilisé pour résoudre le problème de la composition pour l'inclusion de traces et la simulation est basée sur l'utilisation du médiateur large. Dans un premier temps, nous avons montré par un contre-exemple que cette approche ne peut pas être appliquée pour la bisimulation et l'équivalence de traces. Ensuite, nous avons proposé une méthode basée sur la synthèse de contrôleur, pour la résolution du problème de la composition pour la bisimulation et l'équivalence de traces. Enfin, pour résoudre le problème de la synthèse de contrôleur pour la bisimulation, nous avons proposé deux algorithmes qui retournent tous deux une solution en un temps doublement exponentiel, par rapport à la taille de l'entrée du problème. Le premier algorithme est basé sur la satisfaction d'une formule du μ-calcul et le second algorithme est basé sur la filtration. L'avantage de la méthode basée sur le μ-calcul est qu'elle permet d'exprimer d'autres propriétés, en plus de la bisimulation entre deux systèmes. Une de ces propriétés peut être la vivacité du système ou sa tolérance aux fautes. Nous proposons également une méthode basée sur la filtration. Cette méthode ne s'applique que pour la résolution du problème de la composition pour la bisimulation, mais elle a l'avantage d'être résolue par un algorithme simple à implémenter. Ce qui n'est pas le cas de l'algorithme basée sur le μ-calcul qui nécessite d'avoir la procédure pour la résolution du problème de satisfiabilité d'une formule du

μ-calcul.
Les deux algorithmes proposés nécessitent un calcul explicite du produit des services, ce qui est une étape particulièrement coûteuse lorsque le nombre des services est élevé. Cependant, les bornes inférieures de complexité montrent qu'il n'existe pas d'algorithme de moindre complexité permettant de résoudre le problème dans le cas de la simulation et de l'inclusion de traces.

Chapitre 7

Composition de services dans un environnement avec communication synchrone

À la différence des Chapitres 4 et 6, nous considérons dans ce chapitre une communication synchrone entre les services [BCHK09]. En d'autres termes, nous supposons qu'un service ne peut envoyer de message que s'il existe un autre service prêt à le recevoir. Réciproquement, un service ne peut recevoir de message que s'il existe un autre service prêt à l'envoyer. Ainsi, les services doivent se synchroniser, afin de communiquer. Dans le reste de ce chapitre, nous présentons le modèle formel que nous considérons pour représenter les services, ainsi que les trois variantes suivantes du problème de la composition :

1. Problème de l'existence d'une évaluation $\mathcal{PEE}$.
2. Problème de décision d'une formule caractéristique $\mathcal{PDFC}$.
3. Problème de la synthèse d'une formule caractéristique $\mathcal{PSFC}$.

Ces trois problèmes sont définis pour les relations suivantes : inclusion de traces, équivalence de traces, simulation et bisimulation. Nous donnons, également, les résultats concernant la complexité de ces problèmes, pour ces différentes relations.

7.1 Modèle des services

Comme dans les chapitres précédents, nous considérons un ensemble fini d'actions Σ, un ensemble fini de formules atomiques At et un ensemble fini

de ports $Port$. L'envoi de messages sur un port π est représenté par $!\pi$ et la réception de messages sur ce port, par $?\pi$. Les actions de communication sont ainsi des éléments de l'ensemble $\{!,?\} \times Port$. Contrairement aux chapitres précédents, dans ce chapitre, nous supposons que les ports ne contiennent pas de messages. C'est-à-dire que tout message envoyé à travers un port est instantanément reçus. Par ailleurs, il n'y a pas de file de messages dans un port, en attente d'une réception. Les actions internes dans Σ ainsi que les actions de communications dans $\{!,?\} \times Port$, peuvent être conditionnées par un sous-ensemble de $Li(At)$. Rappelons que $Li(At)$ est l'ensemble des littéraux qu'il est possible de construire à partir de At. Afin de simplifier le problème de la composition, nous considérons que les actions n'engendre pas d'effets [1]. C'est-à-dire que l'évaluation des formules atomiques ne peut être modifiée par l'exécution d'une action. Donnons les définitions formelles d'un service, du produit de services avec communication synchrone et de l'exécution d'un service.

Définition 7.1.1 (**Service**). Un *service* est un automate communicant conditionnel $\mathcal{A}c = (Q, q_0, F, \delta)$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At tel que :

- Q est un ensemble fini d'états,
- q_0 est l'état initial,
- F est l'ensemble des états finaux,
- $\delta \subseteq 2^{Li(At)} \times Q \times (\Sigma \cup (\{!,?\} \times Port)) \times Q \times 2^{\emptyset}$ est la relation de transition.

Dans le modèle de ce chapitre, nous ne considérons pas de service client ni de service médiateur. Nous considérons uniquement un service but et des services disponibles. Sachant que nous ne considérons pas de service client alors nous supposerons que le service but n'effectue aucune communication. En d'autres termes, ce service effectue des actions internes uniquement. Les restrictions sur le service but et le service client nous permettent de simplifier la présentation des preuves. Néanmoins, les preuves restent applicables dans le cas général. Le produit que nous considérons entre les services est un produit asynchrone pour les actions internes et synchrone pour les actions de communication.

Définition 7.1.2 (**Produit de ACC avec communication synchrone**). Soient $\mathcal{A}c^1 = (Q^1, q_0^1, F^1, \delta^1), \ldots, \mathcal{A}c^n = (Q^n, q_0^n, F^n, \delta^n)$ des ACC sur $\Sigma \cup (\{!,?\} \times Port)$ et At. Par $\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n$, nous notons le *produit avec communication synchrone* de $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$, c'est-à-dire l'automate communicant conditionnel $\mathcal{A}c = (Q, q_0, F, \delta)$ sur $\Sigma \cup \{\epsilon\}$ et At tel que :

[1] Dans ce chapitre, tous les ACC, sans exception, ont des transitions sans effets.

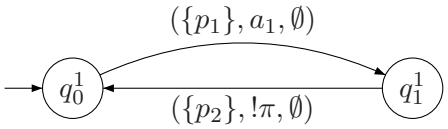

FIG. 7.1 – Automate communicant conditionnel $\mathcal{A}c^1$.

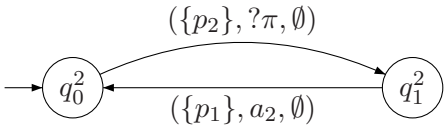

FIG. 7.2 – Automate communicant conditionnel $\mathcal{A}c^2$.

- $Q = Q^1 \times \ldots \times Q^n$,
- $q_0 = (q_0^1, \ldots, q_0^n)$,
- $F = F^1 \times \ldots \times F^n$ et
- $\delta \subseteq 2^{Li(At)} \times Q \times (\Sigma \cup \{\epsilon\}) \times Q \times 2^{\emptyset}$ est la relation de transition définie par $(J, (q_1^1, \ldots, q_1^n), a, (q_2^1, \ldots, q_2^n), \emptyset) \in \delta$ ssi une des deux conditions suivantes est satisfaite :
  (1) $a \in \Sigma$ et il existe un entier $i \in \{1, \ldots, n\}$ tel que $(J, q_1^i, a, q_2^i, \emptyset) \in \delta^i$ et pour tout $j \in \{1, \ldots, n\}$, si $j \neq i$ alors $q_2^j = q_1^j$.
  (2) $a = \epsilon$, il existe $i, k \in \{1, \ldots, n\}$, il existe des ensembles consistants $J^i, J^k \in 2^{Li(At)}$ et il existe un port $\pi \in Port$ tels que :
  - $i \neq k$,
  - l'ensemble $J^i \cup J^k$ est consistant et $J = J^i \cup J^k$,
  - $(J^i, q_1^i, !\pi, q_2^i, \emptyset) \in \delta^i$ et $(J^k, q_1^k, ?\pi, q_2^k, \emptyset) \in \delta^k$ et
  - pour tout $j \in \{1, \ldots, n\}$, si $j \notin \{i, k\}$ alors $q_2^j = q_1^j$.

Exemple 7.1.3. *Nous considérons l'ensemble d'actions $\Sigma = \{a_1, a_2\}$, l'ensemble de ports $Port = \{\pi\}$, l'ensemble des formules atomiques $At = \{p_1, p_2\}$ et les automates communicants conditionnels $\mathcal{A}c^1$ et $\mathcal{A}c^2$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At, représentés respectivement dans la Figure 7.1 et la Figure 7.2. Le produit avec communication synchrone de $\mathcal{A}c^1$ et $\mathcal{A}c^2$ est représenté dans la Figure 7.3.*

Concernant l'exécution d'un ACC, comme dans les chapitres précédents, l'initialisation des formules atomiques est représentée par un ensemble maximal consistant de littéraux I_0. Sachant que les actions exécutées par les ACC

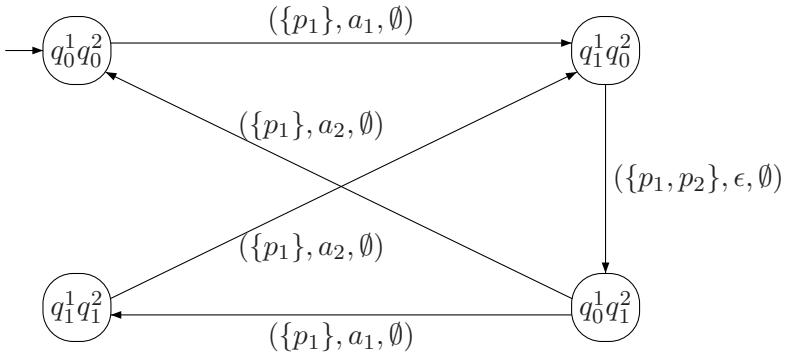

FIG. 7.3 – Automate communicant conditionnel $\mathcal{A}c^1 \circledast \mathcal{A}c^2$.

n'ont pas d'effets alors l'ensemble I_0 ne peut être modifié. De plus, les ports ne stockent pas de messages. Par conséquent, le contenu des ports n'est pas modifié. De ce fait, l'exécution d'un ACC $\mathcal{A}c$ ne contiendra que les transitions dans $\mathcal{A}c$ avec des conditions satisfaites.

Définition 7.1.4 (Exécution des ACC). Soit I_0 un ensemble maximal consistant et $\mathcal{A}c = (Q, q_0, F, \delta)$ un automate communicant conditionnel sur $(\Sigma \cup (\{!, ?\} \times Port) \cup \{\epsilon\})$ et At. Nous appelons *exécution* de $\mathcal{A}c$, l'automate $Exec(\mathcal{A}c, I_0) = (Q', q_0', F', \rightarrow')$ défini sur $\Sigma \cup (\{!, ?\} \times Port) \cup \{\epsilon\}$ tel que :

- $Q' = Q$,
- $q_0' = q_0$,
- $F' = F$ et
- $\rightarrow' \subseteq Q' \times (\Sigma \cup (\{!, ?\} \times Port) \cup \{\epsilon\}) \times Q'$ est la relation de transition définie par $(q, a, q') \in \rightarrow'$ ssi il existe une transition $(J, q, a, q', \emptyset) \in \delta$ telle que $J \subseteq I_0$.

Il est clair à partir de cette définition que l'exécution d'un ACC $\mathcal{A}c$ est de taille au plus égale à celle de $\mathcal{A}c$. Rappelons que la taille d'un ACC défini sur $(\Sigma \cup (\{!, ?\} \times Port) \cup \{\epsilon\})$ et At est de taille exponentielle par rapport à At et linéaire par rapport à Σ et $Port$.

Exemple 7.1.5. *Nous considérons les automates communicants conditionnels $\mathcal{A}c^1$ et $\mathcal{A}c^2$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At de l'exemple 7.1.3. En*

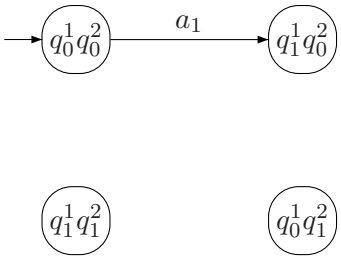

FIG. 7.4 – Automate fini $Exec(\mathcal{A}c^1 \circledast \mathcal{A}c^2, \{p_1, \neg p_2\})$.

considérant l'ensemble $I_0 = \{p_1, \neg p_2\}$*, l'exécution du produit avec communication synchrone des* ACC $\mathcal{A}c^1$ *et* $\mathcal{A}c^2$ *est représentée dans la Figure 7.4.*

7.2 Composition des services

Soit At un ensemble de formules atomiques et $\mathcal{F}(At)$ l'ensemble de toutes les formules booléennes ϕ construites à partir de At de la façon suivante :

$$\phi ::= p \mid \bot \mid \neg\phi \mid (\phi \vee \phi), \text{ où } p \in At.$$

Les autres constructeurs sont définis de façon usuelle. Soit I_0 un ensemble maximal consistant de littéraux sur At. Une évaluation compatible avec I_0 est une fonction V_0 de At vers $\{0, 1\}$ telle que :

$$V_0(p) = \begin{cases} 1 & \text{si } p \in I_0 \\ 0 & \text{sinon} \end{cases}$$

En utilisant la méthode usuelle, toute évaluation V_0 peut être étendue à une fonction $\widehat{V_0}$ de $\mathcal{F}(At)$ vers $\{0, 1\}$.

Dans ce chapitre, les problèmes de la composition auxquels nous nous intéressons se focalisent sur la construction ou l'existence d'une formule, au lieu de se focaliser sur la construction ou l'existence d'un médiateur. Cette formule lorsqu'elle est satisfaite garantit que le système composé des services disponibles est équivalent au service but. En pratique, cette formule permet, par exemple, de spécifier des restrictions sur les certificats que l'utilisateur des services doit fournir. Plus précisément, nous nous intéressons aux trois problèmes suivants :

1. Problème de l'existence d'une évaluation $\mathcal{PEE}$.
2. Problème de décision d'une formule caractéristique $\mathcal{PDFC}$.
3. Problème de la synthèse d'une formule caractéristique $\mathcal{PSFC}$.

Problème $\mathcal{PEE}(\approx)$: problème de l'existence d'une évaluation

Instance : les ensembles finis Σ, $Port$ et At, un automate fini conditionnel $\mathcal{A}c^{but}$ sur Σ et At et des automates communicants conditionnels $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At.

Question : existe-il un ensemble maximal consistant $I_0 \in 2^{Li(At)}$ tel que

$$Exec(\mathcal{A}c^{but}, I_0) \approx Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\}) \,?$$

où $\approx \in \{\subseteq_{tr}, \equiv_{tr}, \leq_{si}, \longleftrightarrow_{bi}\}$.

Le problème de l'existence d'une évaluation consiste, étant donnés un service but et des services disponibles, à déterminer l'existence d'une évaluation pour les formules atomiques telle que l'exécution du but soit équivalente à celles du système composé de l'ensemble des services disponibles. Intuitivement cette évaluation représente les critères que l'utilisateur des services doit satisfaire afin de lui garantir une réponse à sa requête.

Problème $\mathcal{PDFC}(\approx)$: problème de décision d'une formule caractéristique

Instance : les ensembles finis Σ, $Port$ et At, un automate fini conditionnel $\mathcal{A}c^{but}$ sur Σ et At, des automates communicants conditionnels $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At et une formule booléenne ϕ.

Question : est-il vrai que pour tout ensemble maximal consistant $I_0 \in 2^{Li(At)}$,

$$Exec(\mathcal{A}c^{but}, I_0) \approx Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$$
$$\text{ssi}$$
$$\widehat{V_0}(\phi) = 1 \,?$$

où $\approx \in \{\subseteq_{tr}, \equiv_{tr}, \leq_{si}, \longleftrightarrow_{bi}\}$.

Le problème de décision d'une formule caractéristique consiste, étant donnés un service but, des services disponibles et une formule booléenne, à décider si la formule caractérise la composition. En d'autres termes on veut décider si pour toute évaluation des formules atomiques, la composition est satisfaite ssi la formule booléenne est évaluée à vrai.

Problème $\mathcal{PSFC}(\approx)$: problème de la synthèse d'une formule caractéristique

Instance : les ensembles finis Σ, $Port$ et At, un automate fini conditionnel $\mathcal{A}c^{but}$ sur Σ et At et des automates communicants conditionnels $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At.

Question : synthétiser une formule ϕ telle que pour tout ensemble maximal consistant $I_0 \in 2^{Li(At)}$

$$Exec(\mathcal{A}c^{but}, I_0) \approx Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$$
$$\text{ssi}$$
$$\widehat{V_0}(\phi) = 1.$$

où $\approx \in \{\subseteq_{tr}, \equiv_{tr}, \leq_{si}, \longleftrightarrow_{bi}\}$.

Le problème de la synthèse d'une formule caractéristique est le problème de synthèse associé au problème précédent. En effet, le problème $\mathcal{PSFC}$ consiste à synthétiser la formule qui caractérise la composition, à partir du service but et des services disponibles. Une fois cette formule synthétisée, pour décider si $\mathcal{A}c^{but}$ peut être composé à partir de $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$, il suffira de décider si ϕ est vrai ou pas.

7.3 Résultats de complexité

Dans cette section nous présentons des résultats concernant la complexité des problèmes $\mathcal{PEE}$, $\mathcal{PDFC}$ et $\mathcal{PSFC}$, pour l'inclusion de traces, l'équivalence de traces, la simulation et la bisimulation. Nous rappelons que dans ce chapitre la communication effectuée entre les services est synchrone. Dans ce qui suit, nous montrons que les problèmes de décision $\mathcal{PEE}(\subseteq_{tr})$, $\mathcal{PEE}(\equiv_{tr})$, $\mathcal{PDFC}(\subseteq_{tr})$ et $\mathcal{PDFC}(\equiv_{tr})$ sont EXPSPACE-complets. Egalement, nous montrons que les problèmes de décisions $\mathcal{PEE}(\leq_{si})$ et $\mathcal{PDFC}(\leq_{si}$) sont EXPTIME-complets. De plus, nous montrons que le problème de synthèse $\mathcal{PSFC}$ est dans FEXPSPACE (resp. FEXPTIME) lorsque les relations d'inclusion de traces et d'équivalence de traces (resp. simulation et bisimulation) sont considérées. Concernant le problème $\mathcal{PEE}(\longleftrightarrow_{bi})$ et $\mathcal{PDFC}(\longleftrightarrow_{bi})$, nous montrons qu'ils sont PSPACE-difficiles et qu'ils sont dans EXPTIME. Notre conjecture est que ces derniers sont EXPTIME-complets. Pour prouver cela, il faut d'abord résoudre le problème suivant : prouver que le problème de savoir si un automate fini est bisimilaire à un produit d'automates finis est EXPTIME-difficile. À notre connaissance, la complexité exacte de ce problème n'est pas connue.

7.3.1 Bornes inférieures de complexité

Afin de montrer que les problèmes $\mathcal{PEE}(\subseteq_{tr})$, $\mathcal{PEE}(\equiv_{tr})$, $\mathcal{PDFC}(\subseteq_{tr})$ et $\mathcal{PDFC}(\equiv_{tr})$ sont EXPSPACE-difficiles, nous utilisons une réduction du problème de l'universalité des expressions régulières avec carré vers ces problèmes. En ce qui concerne les problèmes $\mathcal{PEE}(\leq_{si})$ et $\mathcal{PDFC}(\leq_{si})$, pour

montrer qu'ils sont EXPTIME-difficiles, nous utilisons une réduction du problème de la simulation entre un automate fini et un produit asynchrone d'automates finis vers ces problèmes. Quant aux problèmes $\mathcal{PEE}(\longleftrightarrow_{bi})$ et $\mathcal{PDFC}(\longleftrightarrow_{bi})$, afin de prouver qu'ils sont PSPACE-difficiles, nous procédons en deux étapes. Dans un premier temps, nous montrons que le problème de la bisimulation entre un automate fini et un produit asynchrone d'automates finis est PSPACE-difficile. Pour cela, nous utilisons une réduction du problème d'acceptance d'une machine de Turing déterministe linéairement bornée en espace, à ce problème. Dans un second temps, nous réduisons le problème de la bisimulation entre un automate fini et un produit asynchrone d'automates finis aux problèmes $\mathcal{PEE}(\longleftrightarrow_{bi})$ et $\mathcal{PDFC}(\longleftrightarrow_{bi})$.

Inclusion et équivalence de traces

Pour montrer que les problèmes $\mathcal{PEE}(\subseteq_{tr})$, $\mathcal{PEE}(\equiv_{tr})$, $\mathcal{PDFC}(\subseteq_{tr})$ et $\mathcal{PDFC}(\equiv_{tr})$ sont EXPSPACE-difficiles, nous utilisons une réduction du problème de l'universalité des langages réguliers avec carré, qui est EXPSPACE-difficile [MS72], à ces quatre problèmes. Cette réduction est similaire à celle que nous avons utilisée dans le Chapitre 6, pour prouver le théorème 7.3.4. Nous montrons également que cette réduction est faite en espace logarithmique. Nous rappelons ci-dessous, la définition d'une expression régulière avec carré et du problème de l'universalité des expressions régulières avec carré.

Définition 7.3.1 (**Expression régulière avec carré**). Soit Σ un alphabet fini et Σ^* l'ensemble des mots de longueur finie sur Σ. Nous notons le mot vide (de longueur 0) par ϵ. Les expressions régulières avec carré seront notées $\alpha, \beta, \gamma, \sigma$, etc. *Une expression régulière avec carré* α définie sur Σ est construite comme suit :

$$\alpha := a \mid \epsilon \mid \alpha \circ \beta \mid \alpha \cup \beta \mid \alpha^+ \mid \alpha^2, \text{ où } a \in \Sigma.$$

Chaque expression α définit un langage rationnel sur Σ. Nous notons par $L(\alpha)$ le langage défini par α. De plus, le nombre d'occurrences des opérations $\epsilon, \circ, \cup, ^+, ^2$ sera noté $Op(\alpha)$. Considérons le problème de décision suivant :

Problème UER **:** problème de l'universalité des expressions régulières avec carré

Instance : un alphabet fini Σ et une expression régulière avec carré α définie sur Σ

Question : est-il vrai que $L(\alpha)=\Sigma^*$?

Théorème 7.3.2. *Les problèmes $\mathcal{PEE}(\subseteq_{tr})$ et $\mathcal{PEE}(\equiv_{tr})$ sont EXPSPACE-difficiles.*

Démonstration. Dans ce qui suit, nous prouvons que le problème UER se réduit en espace logarithmique aux problèmes $\mathcal{PEE}(\subseteq_{tr})$ et $\mathcal{PEE}(\equiv_{tr})$. Soit une instance de UER donnée par Σ et α. L'instance $\rho(\Sigma, \alpha)$ de $\mathcal{PEE}(\subseteq_{tr})$ et $\mathcal{PEE}(\equiv_{tr})$ que nous construisons est donnée par Σ', $Port_\alpha$, At, $\mathcal{A}c^{but}$ et $\mathcal{A}c^0_\alpha, \mathcal{A}c^1, \ldots, \mathcal{A}c^{n_\alpha}$. Plus précisément :

- L'ensemble $\Sigma' = \Sigma$.
- L'ensemble $At = \emptyset$.
- Le nombre $n_\alpha = 2 \times Op(\alpha)$.
- L'ensemble $Port_\alpha = \{\pi_1, \ldots, \pi_{n_\alpha}\}$.
- La construction de $\mathcal{A}c^0_\alpha$ est faite par induction sur α. Le détail de cette construction est donné dans la suite de la preuve.
- Pour tout $i \in \{1, \ldots, n_\alpha\}$, $\mathcal{A}c^i = (\{q^i_0, q^i_1\}, q^i_0, \{q^i_1\}, \delta^i)$ est l'automate communicant conditionnel de la Figure 7.5.
- La construction de $\mathcal{A}c^{but}$ est faite à partir de Σ. Le détail de cette construction est donné ci-dessous.

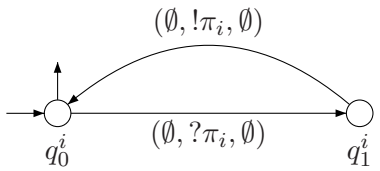

FIG. 7.5 – Automate communicant conditionnel $\mathcal{A}c^i$, $i \in \{1, \ldots, n_\alpha\}$.

1) Construction de $\mathcal{A}c^{but}$

L'automate communicant conditionnel $\mathcal{A}c^{but}$ est construit de façon à ne dépendre que de Σ. L'idée de cette construction est d'avoir, pour $I_0 = \emptyset$, le langage reconnu par l'exécution de $\mathcal{A}c^{but}$ égal à Σ^*. Pour cela, nous construisons $\mathcal{A}c^{but} = (Q^{but}, q^{but}_0, F^{but}, \delta^{but})$ tel que :

- $Q^{but} = \{q^{but}_0\}$,
- $F^{but} = \{q^{but}_0\}$ et
- $\delta^{but} = \{(\emptyset, q^{but}_0, a, q^{but}_0, \emptyset) \mid a \in \Sigma\}$.

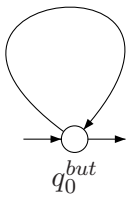

FIG. 7.6 – Automate communicant conditionnel $\mathcal{A}c^{but}$, lorsque $\Sigma = \{a_1, a_2, \ldots, a_m\}$.

En supposant que $\Sigma = \{a_1, a_2, \ldots, a_m\}$, l'automate communicant conditionnel $\mathcal{A}c^{but}$ est celui de la Figure 7.6. L'automate $Exec(\mathcal{A}c^{but}, \emptyset)$ est celui de la Figure 7.7. Le fait que $\mathcal{A}c^{but}$ ne contient aucune transition étiquetée par

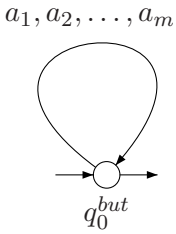

FIG. 7.7 – L'automate fini $Exec(\mathcal{A}c^{but}, \emptyset)$ pour le $\mathcal{A}c^{but}$ de la Figure 7.6.

ϵ implique :

$$Tr_\epsilon(Exec(\mathcal{A}c^{but}, \emptyset)) = Tr_\emptyset(Exec(\mathcal{A}c^{but}, \emptyset)) = \Sigma^*.$$

2) Construction de $\mathcal{A}c_\alpha^0$

La construction de $\mathcal{A}c_\alpha^0$ est telle que : $Tr_\epsilon(Exec(\mathcal{A}c_\alpha^0 \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^{n_\alpha}, \emptyset)) = L(\alpha)$. Ainsi, le rôle des $\mathcal{A}c^i$, $i \in \{1, \ldots, n_\alpha\}$ est uniquement de se synchroniser avec les actions de communications exécutées par $\mathcal{A}c_\alpha^0$. Bien entendu, une fois synchronisées les actions de communications seront non observables.

Remarque 7.3.3. *Afin de ne pas encombrer les Figures 7.10, 7.11, 7.12 et 7.13, les transitions étiquetées par* $(\emptyset, \theta\pi, \emptyset)$ *seront étiquetée par* $\theta\pi$*, où* $\theta \in \{!, ?\}$ *et* $\pi \in Port_\alpha$.

Base de l'induction : Cas où $\alpha = a$. Dans ce cas, $\mathcal{A}c_a^0 = (Q^a, q_0^a, F^a, \delta^a)$ est l'automate communicant conditionnel de la Figure 7.8. Dans $\mathcal{A}c_a^0$ l'ensemble des états finaux est un singleton $F^a = \{q_1^a\}$. De plus, $n_a = 0$ et $Port_a = \emptyset$, car $Op(\alpha) = 0$. Il est clair que :

$$Tr_\emptyset(Exec(\mathcal{A}c_a^0, \emptyset)) = L(a) = \{a\}.$$

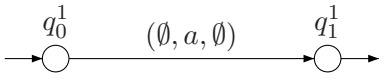

FIG. 7.8 – ACC $\mathcal{A}c_\alpha^0$, dans le cas où $\alpha = a$.

Hypothèse de l'induction : considérons deux expressions régulières avec carré γ et σ pour lesquels il existe des ACC $\mathcal{A}c_\gamma^1 = (Q^\gamma, q_0^\gamma, F^\gamma, \delta^\gamma)$ sur $\Sigma \cup (\{!, ?\} \times Port_\gamma)$ et $\mathcal{A}c_\sigma^1 = (Q^\sigma, q_0^\sigma, F^\sigma, \delta^\sigma)$ sur $\Sigma \cup (\{!, ?\} \times Port_\sigma)$ tels que :
– $n_\gamma = 2 \times Op(\gamma)$ et $n_\sigma = 2 \times Op(\sigma)$,
– $F^\gamma = \{q_1^\gamma\}$ et $F^\sigma = \{q_1^\sigma\}$,
– $Tr_\epsilon(Exec(\mathcal{A}c_\gamma^0 \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\gamma}, \emptyset)) = L(\gamma)$ et
– $Tr_\epsilon(Exec(\mathcal{A}c_\sigma^0 \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\sigma}, \emptyset)) = L(\sigma)$.

Pas de l'induction : Cas où $\alpha = \epsilon$. Dans ce cas, $Op(\epsilon) = 1$ donc $n_\epsilon = 2$ et $Port_\alpha = \{\pi_1, \pi_2\}$. De plus, $\mathcal{A}c_\epsilon^0 = (Q^\epsilon, q_0^\epsilon, \{q_1^\epsilon\}, \delta^\epsilon)$ est l'automate communicant conditionnel sur $\Sigma \cup (\{!, ?\} \times Port_\epsilon)$ de la Figure 7.9. Il est clair que :

$$Tr_\epsilon(Exec(\mathcal{A}c_\epsilon^0 \circledast \mathcal{A}c^1 \circledast \mathcal{A}c^2, \emptyset)) = L(\epsilon) = \{\epsilon\}.$$

La raison est que toutes les actions de communication se synchronisent et deviennent des ϵ. **Cas où** $\alpha = \gamma \circ \sigma$. Dans ce cas, $Op(\gamma \circ \sigma) = Op(\gamma) + Op(\Sigma) + 1$. De plus, $n_{\gamma\circ\sigma} = 2 \times Op(\gamma \circ \sigma)$. Ainsi, $n_{\gamma\circ\sigma} = 2 \times Op(\gamma) + 2 \times Op(\sigma) + 2$. Par hypothèse d'induction, $n_\gamma = 2 \times Op(\gamma)$ et $n_\sigma = 2 \times Op(\sigma)$. Par conséquent, $n_{\gamma\circ\sigma} = n_\gamma + n_\sigma + 2$. On pose $Port_{\gamma\circ\sigma} = Port_\gamma \cup Port_\sigma \cup \{\pi_{n_\gamma+n_\sigma+1}, \pi_{n_\gamma+n_\sigma+2}\}$ et $\mathcal{A}c_{\gamma\circ\sigma}^0$ est l'automate communicant conditionnel de la Figure 7.10. Il est facile de vérifier que : $Tr_\epsilon(Exec(\mathcal{A}c_{\gamma\circ\sigma}^0 \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^{n_\gamma} \circledast \mathcal{A}c^{n_\gamma+1} \circledast \ldots \circledast$

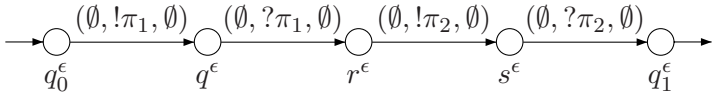

FIG. 7.9 – ACC $\mathcal{A}c_\alpha^0$, dans le cas où $\alpha = \epsilon$.

$\mathcal{A}c^{n_\gamma+n_\sigma} \circledast \mathcal{A}c^{n_\gamma+n_\sigma+1} \circledast \mathcal{A}cn_\gamma + n_\sigma + 2, \emptyset)) = L(\gamma \circ \sigma) = \{ww' \mid w \in L(\gamma)$ et $w' \in L(\sigma)\}$. Observons que $F^{\gamma\circ\sigma} = \{q_1^{\gamma\circ\sigma}\}$.

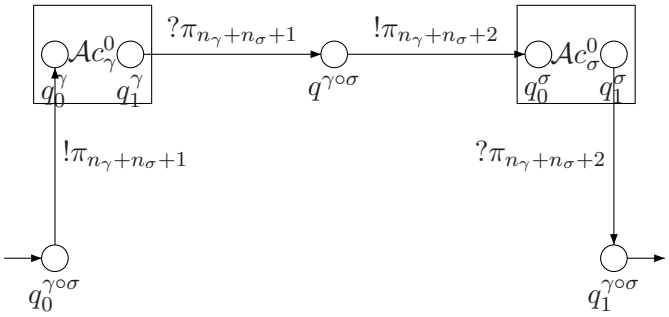

FIG. 7.10 – ACC $\mathcal{A}c_\alpha^0$, dans le cas où $\alpha = \gamma \circ \sigma$.

Cas où $\alpha = \gamma \cup \sigma$. Dans ce cas, $Op(\gamma \cup \sigma) = Op(\gamma) + Op(\Sigma) + 1$. De plus, $n_{\gamma\cup\sigma} = 2 \times Op(\gamma \cup \sigma)$. Ainsi, $n_{\gamma\cup\sigma} = 2 \times Op(\gamma) + 2 \times Op(\sigma) + 2$. Par hypothèse d'induction, $n_\gamma = 2 \times Op(\gamma)$ et $n_\sigma = 2 \times Op(\sigma)$. Par conséquent, $n_{\gamma\cup\sigma} = n_\gamma + n_\sigma + 2$. On pose $Port_{\gamma\cup\sigma} = Port_\gamma \cup Port_\sigma \cup \{\pi_{n_\gamma+n_\sigma+1}, \pi_{n_\gamma+n_\sigma+2}\}$ et $\mathcal{A}c_{\gamma\cup\sigma}^0$ est l'automate communicant conditionnel de la Figure 7.11. Il est facile de vérifier que : $Tr_\epsilon(Exec(\mathcal{A}c_{\gamma\cup\sigma}^0 \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^{n_\gamma} \circledast \mathcal{A}c^{n_\gamma+1} \circledast \ldots \circledast \mathcal{A}c^{n_\gamma+n_\sigma} \circledast \mathcal{A}c^{n_\gamma+n_\sigma+1} \circledast \mathcal{A}c^{n_\gamma+n_\sigma+2}, \emptyset)) = L(\gamma \cup \sigma) = \{w \mid w \in L(\gamma)$ ou $w \in L(\sigma)\}$. Observons que $F^{\gamma\cup\sigma} = \{q_1^{\gamma\cup\sigma}\}$.

Cas où $\alpha = \gamma^+$. Dans ce cas, $Op(\gamma^+) = Op(\gamma) + 1$. De plus, $n_{\gamma^+} = 2 \times Op(\gamma^+)$. Ainsi, $n_{\gamma^+} = 2 \times Op(\gamma) + 2$. Par hypothèse d'induction, $n_\gamma = 2 \times Op(\gamma)$. Par conséquent, $n_{\gamma^+} = n_\gamma + 2$. On pose $Port_{\gamma^+} = Port_\gamma \cup \{\pi_{n_\gamma+1}, \pi_{n_\gamma+2}\}$ et $\mathcal{A}c_{\gamma^+}^0$ est l'automate communicant conditionnel de la Figure 7.12. Il est facile de vérifier que : $Tr_\epsilon(Exec(\mathcal{A}c_{\gamma^+}^0 \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^{n_\gamma} \circledast \mathcal{A}c^{n_\gamma+1} \circledast \mathcal{A}c^{n_\gamma+2}, \emptyset)) = L(\gamma^+) = \{w^n \mid n \geq 0$ et $w \in L(\gamma)\}$. Observons que $F^{\gamma^+} = \{q_1^{\gamma^+}\}$.

Cas où $\alpha = \gamma^2$. Dans ce cas, $Op(\gamma^2) = Op(\gamma) + 1$. De plus, $n_{\gamma^2} =$

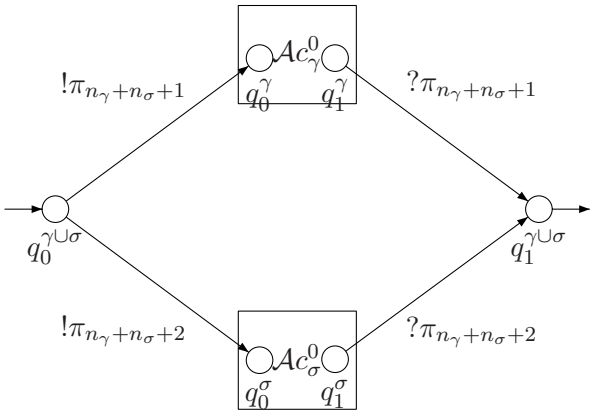

FIG. 7.11 – ACC $\mathcal{A}c^0_\alpha$, dans le cas où $\alpha = \gamma \cup \sigma$.

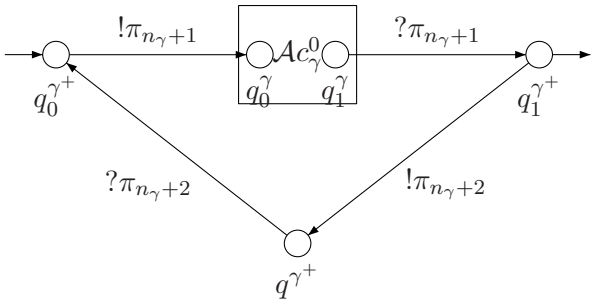

FIG. 7.12 – ACC $\mathcal{A}c^0_\alpha$, dans le cas où $\alpha = \gamma^+$.

$2\times Op(\gamma^+)$. Ainsi, $n_{\gamma^2} = 2\times Op(\gamma)+2$. Par hypothèse d'induction, $n_\gamma = 2 \times Op(\gamma)$. Par conséquent, $n_{\gamma^+} = n_\gamma + 2$. On pose $Port_{\gamma^2} = Port_\gamma \cup \{\pi_{n_\gamma+1}, \pi_{n_\gamma+2}\}$ et $\mathcal{A}c^0_{\gamma^2}$ est l'automate communicant conditionnel de la Figure 7.13. Il est facile de vérifier que : $Tr_\epsilon(Exec(\mathcal{A}c^0_{\gamma^2} \circledast \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^{n_\gamma} \circledast \mathcal{A}c^{n_\gamma+1} \circledast \mathcal{A}c^{n_\gamma+2}, \emptyset)) = L(\gamma^2) = \{w^2 \mid w \in L(\gamma)\}$. Observons que $F^{\gamma^2} = \{q_1^{\gamma^2}\}$.

Nous avons construit tous les éléments de l'instance des problèmes $\mathcal{PEEE}(\subseteq_{tr})$ et $\mathcal{PEEE}(\equiv_{tr})$, il nous reste à vérifier les trois points suivants :

(1) ρ est calculable par une machine de Turing déterministe en utilisant un espace logarithmique,

(2) $L(\alpha) = \Sigma^\star$ ssi pour tout $I_0 \in 2^\emptyset, Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) \subseteq Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0))$.

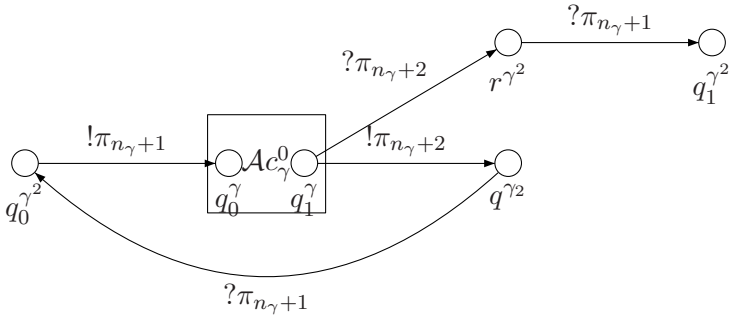

FIG. 7.13 – ACC $\mathcal{A}c_\alpha^0$, dans le cas où $\alpha = \gamma^2$.

(3) $L(\alpha) = \Sigma^\star$ ssi pour tout $I_0 \in 2^\emptyset, Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) = Tr_\epsilon(Exec(\mathcal{A}c_\alpha^0 \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0))$.

Sachant que $2^\emptyset = \{\emptyset\}$, pour vérifier les points (2) et (3) il suffira de vérifier le cas où $I_0 = \emptyset$.

Vérification du point (1)

Une machine de Turing $\mathcal{M}$ calcule ρ de la façon suivante :

(a) La machine $\mathcal{M}$ écrit d'abord sur son ruban de sortie les ensembles Σ et At. Elle écrit ensuite l'ensemble $\{q_0^{but}\}$, contenant l'unique état de $\mathcal{A}c^{but}$. Afin d'écrire les transitions de $\mathcal{A}c^{but}$ la machine utilise un ruban de travail contenant un compteur pour les éléments de Σ.

(b) Ensuite, la machine va écrire l'ensemble $Port_\alpha$ et les automates $\mathcal{A}c^1, \ldots, \mathcal{A}c^{n_\alpha}$, la machine utilisera un compteur avec comme valeur maximale le codage en base 2 de $n_\alpha = 2 \times Op(\alpha)$.

(c) Pour finir, la machine va écrire les éléments de $\mathcal{A}c_\alpha^0$. La fonction de transition de $\mathcal{A}c_\alpha^0$ est l'élément qui utilisera le plus d'espace dans les rubans de travail. Afin d'écrire la fonction de transition δ^α, la machine $\mathcal{M}$ utilise cinq rubans de travail. A savoir, un premier ruban sera dédié à un compteur pour les états [2] de $\mathcal{A}c_\alpha^0$. Un deuxième ruban sera dédié à un compteur pour les ports. Un troisième ruban permettra

[2] Il est facile de vérifier que la valeur maximale que peut atteindre ce compteur est bornée par un polynôme en fonction du nombre d'opérations dans α. Bien entendu cette valeur maximale sera codée en base 2.

de mémoriser l'état initial et l'état final de la dernière opération lue dans le ruban d'entrée. Cela permettra d'écrire les transitions entre les états générés par la dernière opération et les états qui sont générés par l'opération en cours de lecture (dans le ruban d'entrée). De la même façon, un quatrième ruban permettra de mémoriser les deux ports générés par l'opération en cours de lecture (sur le ruban d'entrée). Un cinquième ruban permettra de garder en mémoire tous les états générés par l'opération courante.

Toutes ces étapes peuvent bien-sûr être réalisées de façon déterministe en utilisant un espace logarithmique.

Vérification du point (2)

Nous avons montré que $\mathcal{A}c^{but}$, $\mathcal{A}c^0_\alpha$ et $\mathcal{A}c^1, \ldots, \mathcal{A}c^{n_\alpha}$ sont construits tels que :

$$Tr_\epsilon(Exec(\mathcal{A}c^{but}, \emptyset)) = \Sigma^\star \tag{7.1}$$

$$Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, \emptyset)) = L(\alpha) \tag{7.2}$$

Concernant le problème $\mathcal{PEE}(\subseteq_{tr})$, nous devons montrer que : $L(\alpha) = \Sigma^\star$ ssi pour tout $I_0 \in 2^\emptyset$,

$$Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) \subseteq Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0)). \tag{7.3}$$

$\Rightarrow$ Considérons l'implication de gauche à droite. Supposons que $L(\alpha) = \Sigma^\star$. Ainsi $\Sigma^\star \subseteq L(\alpha)$. A partir des égalités 7.1 et 7.2, on en déduit $Tr_\epsilon(Exec(\mathcal{A}c^{but}, \emptyset)) \subseteq Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, \emptyset)$. Par conséquent, sachant que $2^\emptyset = \{\emptyset\}$ alors pour tout $I_0 \in 2^\emptyset$ on a $Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) \subseteq Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0))$.

$\Leftarrow$ Considérons l'implication de droite à gauche. Supposons que pour tout $I_0 \in 2^\emptyset$ on a $Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) \subseteq Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0))$. Donc cet inclusion est vraie pour $I_0 = \emptyset$. A partir des égalités 7.1 et 7.2, on en déduit $\Sigma^\star \subseteq L(\alpha)$. Rappelons que par définition $L(\alpha) \subseteq \Sigma^\star$. Par conséquent, $L(\alpha) = \Sigma^\star$.

Vérification du point (3)

Concernant le problème $\mathcal{PEE}(\equiv_{tr})$, nous devons montrer que : $L(\alpha) = \Sigma^\star$ ssi pour tout $I_0 \in 2^\emptyset$,

$$Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) = Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0)). \tag{7.4}$$

Vérifier l'équivalence pour tout $I_0 \in 2^{\emptyset}$ revient à vérifier l'équivalence pour $I_0 = \emptyset$. A partir des égalités 7.1 et 7.2, on en déduit l'équivalence 7.4.

□

Théorème 7.3.4. *Les problèmes $\mathcal{PDFC}(\subseteq_{tr})$ et $\mathcal{PDFC}(\equiv_{tr})$ sont EXPSPACE-difficiles.*

Démonstration. Nous utilisons une réduction du problème UER aux problèmes $\mathcal{PDFC}(\subseteq_{tr})$ et $\mathcal{PDFC}(\equiv_{tr})$. Soit une instance du problème UER donnée par Σ et α. Une instance $\rho(\Sigma, \alpha)$ des problèmes $\mathcal{PDFC}(\subseteq_{tr})$ et $\mathcal{PDFC}(\equiv_{tr})$ est donnée par Σ', $Port_\alpha$, At, $\mathcal{A}c^{but}$, $\mathcal{A}c^0_\alpha, \mathcal{A}c^1, \ldots, \mathcal{A}c^{n_\alpha}$ et ϕ. Dans cette réduction, $\phi = \top$. Par ailleurs, la construction de Σ', $Port_\alpha$, At, $\mathcal{A}c^{but}$, $\mathcal{A}c^0_\alpha$ et $\mathcal{A}c^1, \ldots, \mathcal{A}c^{n_\alpha}$ est identique à celle décrite dans la preuve du théorème 7.3.2.

Concernant le problème $\mathcal{PDFC}(\subseteq_{tr})$, nous devons montrer que : $L(\alpha) = \Sigma^\star$ ssi (pour tout $I_0 \in 2^{\emptyset}, Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) \subseteq Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0))$ ssi $\widehat{V_0}(\phi) = 1$).

$\Rightarrow$ Considérons l'implication de gauche à droite. Supposons que $L(\alpha) = \Sigma^\star$. Nous avons montré dans la preuve du théorème 7.3.2, que :
$L(\alpha) = \Sigma^\star$ ssi pour tout $I_0 \in 2^{\emptyset}$,

$$Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) \subseteq Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0)).$$

Par conséquent, pour tout $I_0 \in 2^{\emptyset}, Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) \subseteq Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0))$. Sachant que $\phi = \top$ alors $\widehat{V_0}(\phi) = 1$. Ainsi, pour tout $I_0 \in 2^{\emptyset}$:

$$Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) \subseteq Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0))$$
$$\text{ssi}$$
$$\widehat{V_0}(\phi) = 1.$$

$\Leftarrow$ Considérons l'implication de droite à gauche. Supposons que pour tout $I_0 \in 2^{\emptyset}$:

$$Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) \subseteq Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0))$$
$$\text{ssi}$$
$$\widehat{V_0}(\phi) = 1.$$

Sachant que $\phi = \top$ alors pour tout $I_0 \in 2^{\emptyset}$, $\widehat{V_0}(\phi) = 1$. Par conséquent, pour tout $I_0 \in 2^{\emptyset}$, $Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) \subseteq Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0))$. A partir de l'équivalence 7.3, on en déduit que $L(\alpha) = \Sigma^\star$.

Concernant le problème $\mathcal{PDFC}(\equiv_{tr})$, on utilise le même raisonnement, en se basant sur l'équivalence suivante :

$L(\alpha) = \Sigma^\star$ ssi pour tout $I_0 \in 2^\emptyset$,

$$Tr_\epsilon(Exec(\mathcal{A}c^{but}, I_0)) = Tr_\epsilon(Exec(\mathcal{A}c^0_\alpha \circledast \mathcal{A}c^1 \ldots \circledast \mathcal{A}c^{n_\alpha}, I_0)).$$

Cette équivalence, nous l'avons démontrée dans la preuve du théorème 7.3.2. De plus, dans la preuve du théorème 7.3.2, nous avons prouvé que la réduction proposée est calculable en temps polynomial par rapport à la taille de Σ et de α. Sachant que le temps nécessaire pour la construction de ϕ est constant par rapport à la taille de Σ et de α alors l'instance $\rho(\Sigma, \alpha)$ des problèmes $\mathcal{PDFC}(\subseteq_{tr})$ et $\mathcal{PDFC}(\equiv_{tr})$ est calculable en temps polynomial par rapport à la taille de Σ et de α. □

Simulation

Les problèmes de décision $\mathcal{PEE}(\leq_{si})$ et $\mathcal{PDFC}(\leq_{si})$ sont EXPTIME-difficiles. Pour le prouver, nous utilisons des résultats connus concernant le problème de la simulation entre un automate fini est un produit asynchrone d'automates finis. Considérons le problème suivant :

Instance : des automates finis déterministes $\mathcal{A}$, $\mathcal{B}^1, \ldots, \mathcal{B}^n$ sur Σ.

Question : est il vrai que

$$\mathcal{A} \leq_{si} \mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n \,?$$

Dans [MW08], les auteurs ont montré que ce problème est EXPTIME-difficile. En se basant sur ce résultat, nous obtenons le théorème suivant.

Théorème 7.3.5. *Les problèmes de décision $\mathcal{PEE}(\leq_{si})$ et $\mathcal{PDFC}(\leq_{si})$ sont EXPTIME-difficiles.*

Démonstration. Rappelons que les automates finis sont des automates communicants conditionnels définis sur un ensemble vide de littéraux. Considérons un ensemble Σ et des automates finis $\mathcal{A}$, $\mathcal{B}^1, \ldots \mathcal{B}^n$ sur Σ. L'instance du problème $\mathcal{PEE}(\leq_{si})$ est construite comme suit :

- un ensemble fini d'actions $\Sigma' = \Sigma$,
- un ensemble de ports $Port = \emptyset$,
- un ensemble de formules atomiques $At = \emptyset$,
- un ACC $\mathcal{A}c^{but} = Auto^{-1}(\mathcal{A}^1)$ et
- les ACC $\mathcal{A}c^1 = Auto^{-1}(\mathcal{B}^1), \ldots, \mathcal{A}c^n = Auto^{-1}(\mathcal{B}^n)$.

Il est clair que cette réduction est calculable en espace logarithmique. De plus, $\mathcal{A} \leq_{si} \mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$ ssi il existe un ensemble maximal consistant $I_0 \in 2^{\emptyset}$ tel que :

$$Exec(\mathcal{A}c^{but}, I_0) \leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\}).$$

En effet, considérons d'abord l'implication de gauche à droite, supposons que $\mathcal{A} \leq_{si} \mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$. Pour $I_0 = \emptyset$ il est évident que $Exec(\mathcal{A}c^{but}, \emptyset) = Auto(\mathcal{A}c^{but})$ et $Exec(\mathcal{A}c^1, \emptyset) = Auto(\mathcal{A}c^1), \ldots, Exec(\mathcal{A}c^n, \emptyset) = Auto(\mathcal{A}c^n)$. Sachant que $\mathcal{A}c^{but} = Auto(\mathcal{A}^1)$ et $\mathcal{A}c^1 = Auto(\mathcal{B}^1), \ldots, \mathcal{A}c^n = Auto(\mathcal{B}^n)$ alors $Exec(\mathcal{A}c^{but}, \emptyset) = \mathcal{A}^1$ et $Exec(\mathcal{A}c^1, \emptyset) = \mathcal{B}^1, \ldots, Exec(\mathcal{A}c^n, \emptyset) = \mathcal{B}^n$. Par conséquent, $Exec(\mathcal{A}c^{but}, \emptyset) \leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, \emptyset)$. Sachant que $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$ sont définis sur Σ et At alors le produit est également défini sur ces deux ensembles. Ainsi, on a une simulation modulo l'ensemble vide ssi on a une simulation modulo $\{\epsilon\}$. Par conséquent, il existe $I_0 \in 2^{\emptyset}$ tel que $Exec(\mathcal{A}c^{but}, I_0) \leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$.

Considérons maintenant l'implication de droite à gauche et supposons que $\mathcal{A} \not\leq_{si} \mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$. Dans ce cas, si $I_0 = \emptyset$ alors $Exec(\mathcal{A}c^{but}, I_0) \not\leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$. Sachant que $At = \emptyset$ alors il n'existe pas de I_0 dans 2^{At} tel que $Exec(\mathcal{A}c^{but}, I_0) \leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$.

Pour monter que le problème $\mathcal{PDFC}(\leq_{si})$ est EXPTIME-difficile nous utilisons la même réduction avec $\phi = \top$. En effet, si $\mathcal{A} \leq_{si} \mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$ alors pour tout I_0 dans $2^{\emptyset}$ on a $Exec(\mathcal{A}c^{but}, I_0) \leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$. De plus, $\phi = \top$ et donc pour tout I_0, on a $\widehat{V_0}(\phi) = 1$. Par conséquent, pour tout $I_0 \in 2^{\emptyset}$, on a $Exec(\mathcal{A}c^{but}, I_0) \leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$ ssi $\widehat{V_0}(\phi) = 1$. Réciproquement, supposons que $\mathcal{A} \not\leq_{si} \mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$. Dans ce cas, $Exec(\mathcal{A}c^{but}, I_0) \not\leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$. Sachant que $\phi = \top$ alors il n'est pas vrai que pour tout $I_0 \in 2^{\emptyset}$, on a $Exec(\mathcal{A}c^{but}, I_0) \leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$ ssi $\widehat{V_0}(\phi) = 1$. □

Bisimulation

Les problèmes de décision $\mathcal{PEE}(\longleftrightarrow_{bi})$ et $\mathcal{PDFC}(\longleftrightarrow_{bi})$ sont PSPACE-difficiles. Dans un premier temps, nous montrons que le problème de la bisimulation entre un automate fini et un produit asynchrone d'automates finis est PSPACE-difficile. Pour cela, nous utilisons une réduction du problème d'acceptance d'une machine de Turing déterministe linéairement bornée en espace, qui est connue pour être PSPACE-difficile. Ensuite, nous utilisons un argument semblable à celui utilisé dans la preuve du théorème 7.3.5, pour prouver que $\mathcal{PEE}(\longleftrightarrow_{bi})$ et $\mathcal{PDFC}(\longleftrightarrow_{bi})$ sont PSPACE-difficiles.

Lemme 7.3.6. *Soit $\mathcal{A}$, $\mathcal{B}^1$, ..., $\mathcal{B}^n$ des automates finis sur Σ. Déterminer si un automate fini $\mathcal{A}$ est bisimilaire à un produit d'automates finis $\mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$ est un problème PSPACE-difficile.*

Démonstration. Par définition, décider si une Machine de Turing déterministe linéairement bornée en espace n'accepte pas un mot donné w est un problème PSPACE-difficile. Dans ce qui suit, nous montrons que ce problème peut être réduit au problème de décision suivant :

Instance : des automates finis déterministes $\mathcal{A}$, $\mathcal{B}^1, \ldots, \mathcal{B}^n$ sur Σ.

Question : est il vrai que

$$\mathcal{A} \longleftrightarrow_{bi} \mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n \,?$$

Soit $\mathcal{M} = (Q^{\mathcal{M}}, q_0^{\mathcal{M}}, q_1^{\mathcal{M}}, \Sigma^{\mathcal{M}}, \delta^{\mathcal{M}})$ une machine de Turing déterministe linéairement bornée en espace telle que :

- $Q^{\mathcal{M}}$ est ensemble fini d'états,
- $q_0^{\mathcal{M}}$ est l'état initial,
- $q_1^{\mathcal{M}}$ est l'état final,
- $\Sigma^{\mathcal{M}} = \{0, 1\}$ est une alphabet finie et
- $\delta^{\mathcal{M}} \subseteq Q^{\mathcal{M}} \times \Sigma^{\mathcal{M}} \times Q^{\mathcal{M}} \times \Sigma^{\mathcal{M}} \times \{+1, -1\}$ est la fonction de transition.

Soit $w = w_1 \ldots w_n \in \Sigma^{\mathcal{M}*}$ un mot de taille n dans $\Sigma^{\mathcal{M}}$. L'instance $\rho(\mathcal{M}, w)$ du problème de la bisimulation que nous construisons est donnée par un nombre d'automates $n' = n$, un ensemble fini d'actions [3] $\Sigma = \{acc, a_1, \ldots, a_n\}$et des automates $\mathcal{A}$, $\mathcal{B}^1, \ldots, \mathcal{B}^n$ sur Σ définis comme suit.

Construction de $\mathcal{A}$

L'automate fini $\mathcal{A} = (Q, q_0, F, \rightarrow_{\mathcal{A}})$ sur Σ est tel que :

- $Q = (Q^{\mathcal{M}} \times \{1, \ldots, n\}) \cup \bot$,
- $q_0 = (q_0^{\mathcal{M}}, 1)$,
- $F = \emptyset$ et
- $\rightarrow_{\mathcal{A}} \subseteq Q \times \Sigma \times Q$ est la relation de transition telle que :
  - $(q^{\mathcal{M}}, i) \rightarrow_{\mathcal{A}}^{\{a\}} (r^{\mathcal{M}}, i')$ ssi $a = a_i$ et il existe $u, u' \in \Sigma^{\mathcal{M}}$ et $d \in \{+1, -1\}$ tel que $i' = i + d$ et $\delta(q^{\mathcal{M}}, u, r^{\mathcal{M}}, u', d)$ est définie, ou $a = acc$, $q^{\mathcal{M}} = q_1^{\mathcal{M}}$ et $r^{\mathcal{M}} = q_1^{\mathcal{M}}$.
  - $(q^{\mathcal{M}}, i) \rightarrow_{\mathcal{A}}^{\{a_j\}} \bot$ ssi $j \neq i$ ou pour tout $u, u' \in \Sigma^{\mathcal{M}}$, $r^{\mathcal{M}} \in Q^{\mathcal{M}}$ et $d \in \{+1, -1\}$, $\delta(q^{\mathcal{M}}, u, r^{\mathcal{M}}, u', d)$ n'est pas définie.
  - pour tout $a \in \Sigma \setminus acc$, $\bot \rightarrow_{\mathcal{A}}^{\{a\}} \bot$.
  - pour tout $a \in \Sigma$, il n'existe pas $(q^{\mathcal{M}}, i) \in Q$ tel que $\bot \rightarrow_{\mathcal{A}}^{\{a\}} (q^{\mathcal{M}}, i)$.

[3] Nous verrons que l'action *acc* permettra de simuler le fait que $\mathcal{M}$ accepte le mot w.

Construction de $\mathcal{B}^i$, $i \in \{1, \ldots, n\}$

L'automate fini $\mathcal{B}^i = (Q^i, q_0^i, F^i, \rightarrow_i)$ sur Σ est tel que :
- $Q^i = (Q^{\mathcal{M}} \times \Sigma^{\mathcal{M}}) \cup \{\perp_i\}$,
- $q_0^i = (q_0^{\mathcal{M}}, w_i)$,
- $F^i = \emptyset$,
- $\rightarrow_i \subseteq Q^i \times \Sigma \times Q^i$ est la relation de transition telle que :
  - $(q^{\mathcal{M}}, u) \rightarrow_i^{\{a_i\}} (r^{\mathcal{M}}, u')$ ssi il existe $d \in \{+1, -1\}$ tel que $\delta(q^{\mathcal{M}}, u, r^{\mathcal{M}}, u', d)$ est définie.
  - $(q^{\mathcal{M}}, u) \rightarrow_i^{\{a_j\}} \perp_i$ ssi $j \neq i$ ou pour tout $r^{\mathcal{M}} \in Q^{\mathcal{M}}, u' \in \Sigma^{\mathcal{M}}$ et $d \in \{+1, -1\}$, $\delta(q, u, q', u', d)$ n'est pas définie.
  - pour tout $a \in \Sigma \setminus acc$, $\perp_i \rightarrow_i^{\{a\}} \perp_i$.
  - pour tout $a \in \Sigma$, il n'existe pas $(q^{\mathcal{M}}, u) \in Q^i$ tel que $\perp_i \rightarrow_i^{\{a\}} (q^{\mathcal{M}}, u)$.

Remarquons que pour tout $i \in \{1, \ldots n\}$ les transitions dans $\mathcal{B}^i$ sont étiquetées par des actions dans $\Sigma \setminus acc$. Maintenant, nous devons prouver les assertions suivantes :

(1) ρ est calculable par une machine de Turing déterministe qui utilise un espace logarithmique.

(2) $\mathcal{M}$ n'accepte pas w ssi $\mathcal{A} \longleftrightarrow_{bi} \mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$.

Vérification du point (1)

Une machine de Turing $\mathcal{M}'$ calcule ρ de la façon suivante :

(a) La machine $\mathcal{M}'$ écrit d'abord sur son ruban de sortie l'ensemble Σ. Pour cela elle aura besoin d'un compteur i qui aura comme valeur maximale le codage en base 2 de n.

(b) Ensuite, afin de pouvoir écrire les états de l'automate fini $\mathcal{A}$, la machine $\mathcal{M}'$ utilise un compteur j pour les états de $Q^{\mathcal{M}}$ et le compteur i. Concernant les transitions de $\rightarrow_{\mathcal{A}}$. La machine commence par écrire toutes les transitions de la forme $(\perp, a, \perp)$, $a \in \sigma \setminus acc$, ainsi que les transitions de la forme $((q_1^{\mathcal{M}}, i), acc, (q_1^{\mathcal{M}}, i))$. Pour cela elle a besoin uniquement du compteur i. Pour écrire toutes les transitions de la forme $((q^{\mathcal{M}}, i), a_i, (r^{\mathcal{M}}, i'))$ la machine $\mathcal{M}'$ doit utiliser un compteur pour $q^{\mathcal{M}}$, pour a_i, pour $r^{\mathcal{M}}$, pour $u \in \Sigma^{\mathcal{M}}$ pour $u' \in \Sigma^{\mathcal{M}}$ et pour $d \in \{+1, -1\}$, le but étant de tester si $\delta(q^{\mathcal{M}}, u, r^{\mathcal{M}}, u', d)$ est écrit sur le ruban d'entrée. Si c'est le cas alors la transition $((q^{\mathcal{M}}, i), a_i, (r^{\mathcal{M}}, i'))$ est écrite dans le ruban de sortie. Dans le cas où pour un $q^{\mathcal{M}}$ et un a_i il n'existe pas de $r^{\mathcal{M}}$, u, u', d tels que $\delta(q^{\mathcal{M}}, u, r^{\mathcal{M}}, u', d)$ alors la transition $((q^{\mathcal{M}}, i), a_i, \perp))$ est écrite sur le ruban de sortie.

(d) Pour finir, le raisonnement utilisé pour la construction de $\mathcal{A}$ est utilisé pour la construction des $\mathcal{A}^i$.

Toutes ces étapes peuvent bien-sûr être réalisées de façon déterministe en utilisant un espace logarithmique.

Vérification du point (2)

$\Leftarrow$ Considérons l'implication de droite à gauche. Supposons que la machine de Turing $\mathcal{M}$ accepte le mot w. Afin de montrer que $\mathcal{A}$ n'est pas bisimilaire à $\mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$, il suffit de montrer que $\mathcal{A}$ n'est pas similaire à $\mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$. Nous raisonnons par l'absurde. Soit $Z \subseteq Q \times (Q^1 \times \ldots \times Q^n)$ une relation de simulation entre $\mathcal{A}$ et $\mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$. Sachant que $\mathcal{M}$ accepte le mot w alors il existe un $i \in \{1, \ldots, n\}$ tel que $(q_0^{\mathcal{M}}, 1, w) \rightarrow_{\mathcal{M}^*} (q_1^{\mathcal{M}}, i, u_1, \ldots, u_n)$. Par construction de $\mathcal{A}$, il existe des états $q, r \in Q$ et des actions $a, a' \in \Sigma \backslash acc$ tels que $(q_0^{\mathcal{M}}, 1) \rightarrow_{\mathcal{A}}^{a} q \ldots q' \rightarrow_{\mathcal{A}}^{a'} (q_1^{\mathcal{M}}, i)$. Sachant que Z est une simulation alors par induction sur la taille des chemins dans $\mathcal{A}$, nous obtenons qu'il existe un état $(q^1, \ldots, q^n) \in Q^1 \times \ldots \times Q^n$ tel que $(q_1^{\mathcal{M}}, i) Z (q^1, \ldots, q^n)$. Rappelons, que $(q_1^{\mathcal{M}}, i) \rightarrow_{\mathcal{A}}^{acc} (q_1^{\mathcal{M}}, i)$. De plus, pour tout $i \in \{1, \ldots, n\}$ il n'existe pas de $r^i \in Q^i$ tel que $q^i \rightarrow^{\{acc\}} r^i$. Ceci contredis le fait que Z est une relation de simulation entre $\mathcal{A}$ et $\mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$. Par conséquent, si la machine de Turing $\mathcal{M}$ accepte le mot w alors $\mathcal{A}$ n'est pas bisimilaire à $\mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$.

$\Rightarrow$ Considérons l'implication de gauche à droite. Supposons que la machine de Turing $\mathcal{M}$ n'accepte pas le mot w. Soit la relation $Z \subseteq Q \times (Q^1 \times \ldots \times Q^n)$ définie comme suit :

- $(q^{\mathcal{M}}, i) Z ((r^{\mathcal{M}}, u_1), \ldots, (s^{\mathcal{M}}, u_i), \ldots, (t^{\mathcal{M}}, u_n))$ ssi $s^{\mathcal{M}} = q^{\mathcal{M}}$ et $(q_0^{\mathcal{M}}, 1, w) \rightarrow_{\mathcal{M}^*} (q^{\mathcal{M}}, i, u_1, \ldots, u_n)$, $i \in \{1, \ldots, n\}$,
- pour tout $q \in Q$ and $i \in \{1, \ldots, n\}$, $q Z (q^1, \ldots, \perp_i, \ldots, q^n)$ et
- pour tout $q^i \in Q^i, i \in \{1, \ldots, n\}$, $\perp Z (q^1, \ldots, q^n)$.

Dans ce qui suit, nous montrons que Z est une bisimulation entre $\mathcal{A}$ et $\mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$. Pour cela, nous montrons que Z et Z^{-1} sont des relations de simulation. Concernant les états initiaux. Sachant que $(q_0^{\mathcal{M}}, 1, w) \rightarrow_{\mathcal{M}^0} (q_0^{\mathcal{M}}, 1, w_1, \ldots, w_n)$, alors par construction de Z on obtient que $(q_0^{\mathcal{M}}, 1) Z ((q_0^{\mathcal{M}}, w_1), \ldots, (q_0^{\mathcal{M}}, w_i), \ldots, (q_0^{\mathcal{M}}, w_n))$. Ainsi, $q_0 Z (q_0^1, \ldots q_0^n)$.

Vérification que Z est une relation de simulation

Considérons les états $q \in Q, q^1 \in Q^1, \ldots, q^n \in Q^n$ tels que $q Z (q^1, \ldots, q^n)$. Supposons qu'il existe un état $r \in Q$ et une action $a \in \Sigma$ tels que $q \rightarrow_{\mathcal{A}}^{\{a\}} r$.

Il est évident que pour tout $i \in \{1,\ldots,n\}$, $q \neq (q_1^{\mathcal{M}}, i)$. En effet, par hypothèse $\mathcal{M}$ n'accepte pas w. En d'autres termes, pour tout $u_1,\ldots,u_n \in \Sigma^{\mathcal{M}}$, $(q_0^{\mathcal{M}}, 1, w) \not\rightarrow_{M^*} (q_1^{\mathcal{M}}, i, u_1, \ldots, u_n)$. De ce fait, par définition de Z, il n'existe pas $u_1,\ldots,u_n$ dans l'alphabet $\Sigma^{\mathcal{M}}$, ni d'état $r^{\mathcal{M}},\ldots,t^m$ tels que $(q_1^{\mathcal{M}}, i)Z((r^{\mathcal{M}}, u_1),\ldots,(q^{\mathcal{M}}, u_i),\ldots,(t^{\mathcal{M}}, u_n))$. Par conséquent, $a \neq acc$. Ainsi, il existe $i \in \{1,\ldots,n\}$ tel que $a = a_i$. Soit $j \in \{1,\ldots,n\}$ tel que $j \neq i$. Par construction de $\mathcal{B}^j$, $q^j \rightarrow_{\mathcal{B}^j}^{\{a_i\}} \perp_j$. Par définition du produit asynchrone, $(q^1,\ldots,q^j,\ldots,q^n) \rightarrow_{\mathcal{B}^1\otimes\ldots\otimes\mathcal{B}^n}^{\{a_i\}} (q^1,\ldots,\perp_j,\ldots,q^n)$. Par construction de Z on a $rZ(q^1,\ldots,\perp_j,\ldots,q^n)$.

Vérification que Z^{-1} est une relation de simulation

Considérons les états $q \in Q, q^1 \in Q^1,\ldots,q^n \in Q^n$ tels que $(q^1,\ldots,q^n)Z^{-1}q$. Supposons qu'il existe un état $(r^1,\ldots,r^n) \in Q^1 \times \ldots Q^n$ et un $i \in \{1,\ldots,n\}$ tels que $(q^1,\ldots,q^n) \rightarrow_{\mathcal{B}^1\otimes\ldots\otimes\mathcal{B}^n}^{\{a_i\}} (r^1,\ldots,r^n)$. Par définition du produit asynchrone, il existe $j \in \{1,\ldots,n\}$ tel que $q^j \rightarrow_{\mathcal{B}^j}^{\{a_i\}} r^j$ et pour tout $j \in \{1,\ldots,n\}$, si $k \neq j$ alors $r^k = q^j$. Deux cas sont possibles pour j. Le cas où $j \neq i$ et le cas où $j = i$.

Considérons le cas où $j \neq i$. Dans ce cas, $r^j = \perp_j$. Par construction de $\mathcal{A}$, il existe $r \in Q$ tel que $q \rightarrow_{\mathcal{A}}^{\{a_i\}} r$. Par construction de Z^{-1} on a $(q^1,\ldots,\perp_j,\ldots,q^n)Z^{-1}r$.

Considérons le cas où $j = i$. Ils existent deux possibilités : $r^i = \perp_i$ ou $r^i = (r^{\mathcal{M}}, u^{i'})$, avec $r^{\mathcal{M}} \in Q^{\mathcal{M}}$ et $u^{i'} \in \Sigma^{\mathcal{M}}$.

1. Supposons que $r^i = \perp_i$. Dans ce cas, le raisonnement est le même que celui du cas $j \neq i$. Plus précisément, par construction de $\mathcal{A}$, il existe $r \in Q$ tel que $q \rightarrow_{\mathcal{A}}^{\{a_i\}} r$. Par construction de Z^{-1} on a $(q^1,\ldots,\perp_i,\ldots,q^n)Z^{-1}r$.
2. Supposons qu'il existe $r^{\mathcal{M}} \in Q^{\mathcal{M}}$ et $u^{i\prime} \in \Sigma^{\mathcal{M}}$ tels que $r^i = (r^{\mathcal{M}}, u_i{}')$. Par conséquent, il existe $q^{\mathcal{M}} \in Q^{\mathcal{M}}$, $u_i \in \Sigma^{\mathcal{M}}$ et $d \in \{-1,+1\}$ tels que $q^i = (q^{\mathcal{M}}, u_i)$ et la transition $\delta(q^{\mathcal{M}}, u_i, r^{\mathcal{M}}, u_i{}', d)$ est définie. Sachant que $(q^1,\ldots,q^n)Z^{-1}q$ alors deux cas sont possibles : $q = \perp$ et $q = (q^{\mathcal{M}}, i)$. Dans le cas où $q = \perp$, on sait que $\perp \rightarrow_{\mathcal{A}}^{\{a_i\}} \perp$. Ce qui implique par construction de Z^{-1} que $(r^1,\ldots,r^n)Z^{-1}$ $\perp$. Considérons le cas où $q = (q^{\mathcal{M}}, i)$. Par construction de $\mathcal{A}$, sachant que $\delta(q^{\mathcal{M}}, u_i, r^{\mathcal{M}}, u_i{}', d)$ est définie, alors $(q^{\mathcal{M}}, i) \rightarrow_{\mathcal{A}}^{\{a_i\}} (r^{\mathcal{M}}, i+d)$. On sait par construction de Z^{-1} que $(q_0^{\mathcal{M}}, 1, w) \rightarrow_{\mathcal{M}^*} (q^{\mathcal{M}}, i, u_1, \ldots, u_n)$. De plus, $\delta(q^{\mathcal{M}}, u_i, r^{\mathcal{M}}, u_i{}', d)$ est définie. Par conséquent, $(q_0^{\mathcal{M}}, 1, w) \rightarrow_{\mathcal{M}^*}$ $(r^{\mathcal{M}}, j+d, u^1, \ldots, u^{i'}, \ldots u^n)$. Ainsi, $(r^1,\ldots,r^n)Z^{-1}r(r^{\mathcal{M}}, i+d)$.

Par conséquent, nous avons montré que si la machine de Turing $\mathcal{M}$ n'accepte pas le mot w alors $\mathcal{A}$ est bisimilaire à $\mathcal{B}^1 \otimes \ldots \otimes \mathcal{B}^n$. Ceci complète la preuve. □

Théorème 7.3.7. *Les problèmes de décision $\mathcal{PEE}(\longleftrightarrow_{bi})$ et $\mathcal{PDFC}(\longleftrightarrow_{bi})$ sont EXPTIME-difficiles.*

Démonstration. Utiliser l'argument de la preuve du théorème 7.3.5. □

7.3.2 Bornes supérieures de complexité

Dans cette section, nous montrons que les problèmes $\mathcal{PEE}$ et $\mathcal{PDFC}$ sont dans EXPSPACE (resp. EXPTIME) pour les relations d'inclusion de traces et d'équivalence de traces (resp. simulation et bisimulation). Concernant, le problème de synthèse $\mathcal{PSFC}$ nous montrons qu'il est dans FEXPSPACE [4] (resp. FEXPTIME [5]) pour les relations d'inclusion de traces et d'équivalence de traces (resp. simulation et bisimulation). Les algorithmes que nous proposons se basent sur des procédures existantes pour la résolution des problèmes d'équivalences entre automates finis [GJ90, HS96]. Dans les algorithmes de cette section, nous utilisons la fonction $MaxConst$, qui associe à un ensemble At l'ensemble contenant les parties maximales consistantes de $Li(At)$.

Inclusion de traces et équivalence de traces

Considérons un ACC $\mathcal{A}c^{but}$ sur Σ et At et des ACC $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At. L'algorithme 7 retourne la valeur "vrai" ssi il existe un ensemble maximal consistant I_0 tel que : $Exec(\mathcal{A}c^{but}, I_0) \subseteq_{tr} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$. Pour retourner une solution, l'algorithme 7 utilise un espace **exponentiel** par rapport à la taille de At et au nombre n de services disponibles. Les raisons sont les suivantes. (1) L'opération de la ligne 6 nécessite un espace **polynomial** [GJ90], par rapport à la taille de $\mathcal{A}$ et $\mathcal{A}'$. (2) La taille de $\mathcal{A}'$ est **exponentielle** par rapport à la taille de n. (3) La boucle For est parcouru $2^{Card(At)}$ fois. En remplaçant $\subseteq_{tr}$ par $\equiv_{tr}$ dans la ligne 6 de l'algorithme 7, on obtient un algorithme qui permet la résolution du problème $\mathcal{PEE}(\equiv_{tr})$, en utilisant un espace exponentiel. Par conséquent, nous obtenons le théorème suivant :

[4] La classe function EXPSPACE, notée FEXPSPACE, représente la classe des problèmes qui ont comme réponses des valeurs qui ne sont pas forcément "Vrai" ou "Faux" et qui peuvent être résolus en espace exponentiel.

[5] La classe function EXPTIME, notée FEXPTIME, représente la classe des problèmes qui ont comme réponses des valeurs qui ne sont pas forcément "Vrai" ou "Faux" et qui peuvent être résolus en temps exponentiel.

Algorithm 7 Algorithme pour la résolution de $\mathcal{PEE}(\subseteq_{tr})$

Resoudre-$\mathcal{PEE}(\subseteq_{tr})(\Sigma, Port, At, \mathcal{A}c^{but}, \mathcal{A}c1, \ldots, \mathcal{A}c^n)$
$\mathcal{A}c \leftarrow \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n$
for $I_0 \in MaxConst(At)$ **do**
    $\mathcal{A} \leftarrow Exec(\mathcal{A}c^{but}, I_0)$
    $\mathcal{A}' \leftarrow Exec(\mathcal{A}c, I_0)$
    if $\mathcal{A} \subseteq_{tr} \mathcal{A}' \quad (\{\epsilon\})$ **then** **retourner** Vrai
    end if
end for
retourner Faux
end procedure

Théorème 7.3.8. *Les problèmes* $\mathcal{PEE}(\subseteq_{tr})$ *et* $\mathcal{PEE}(\equiv_{tr})$ *sont dans EXPSPACE.*

Concernant le problème $\mathcal{PDFC}$, nous avons, en plus des ACC $\mathcal{A}c^{but}$ et $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$, une formule booléenne ϕ. L'algorithme 8 retourne la valeur "vrai" ssi pour tout ensemble maximal consistant I_0 on a : $Exec(\mathcal{A}c^{but}, I_0) \subseteq_{tr} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$ ssi $\widehat{V_0}(\phi) = 1$. Pour retourner une solution,

Algorithm 8 Algorithme pour la résolution de $\mathcal{PDFC}(\subseteq_{tr})$

Resoudre-$\mathcal{PDFC}(\subseteq_{tr})(\Sigma, Port, At, \mathcal{A}c^{but}, \mathcal{A}c1, \ldots, \mathcal{A}c^n, \phi)$
$Bool \leftarrow Vrai$
$\mathcal{A}c \leftarrow \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n$
for $I_0 \in MaxConst(At)$ **do**
    $Ev \leftarrow \widehat{V_0}(\phi)$
    $\mathcal{A} \leftarrow Exec(\mathcal{A}c^{but}, I_0)$
    $\mathcal{A}' \leftarrow Exec(\mathcal{A}c, I_0)$
    if $\mathcal{A} \subseteq_{tr} \mathcal{A}' \quad (\{\epsilon\})$ **then** $Inc \leftarrow Vrai$
    else$Inc \leftarrow Faux$
    end if
    if $Inc \neq Ev$ **then** $Bool \leftarrow Faux$
    end if
end for
retourner $Bool$
end procedure

l'algorithme 8 utilise un espace **exponentiel** par rapport à la taille de At et au nombre n de services disponibles. Les raisons sont identiques à celles

utilisées pour déterminer l'espace utilisé par l'algorithme 7. En remplaçant $\subseteq_{tr}$ par $\equiv_{tr}$ dans la ligne 8 de l'algorithme 8, on obtient un algorithme qui permet la résolution du problème $\mathcal{PDFC}(\equiv_{tr})$, en utilisant un espace exponentiel. Par conséquent.

Théorème 7.3.9. *Les problèmes $\mathcal{PDFC}(\subseteq_{tr})$ et $\mathcal{PDFC}(\equiv_{tr})$ sont dans EXPSPACE.*

Concernant le problème $\mathcal{PSFC}$. L'algorithme 9 retourne une formule ϕ telle que pour tout ensemble maximal consistant I_0 de littéraux sur At on a : $Exec(\mathcal{A}c^{but}, I_0) \subseteq_{tr} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$ ssi $\widehat{V_0}(\phi) = 1$. Pour retourner une solution, l'algorithme 9 utilise un espace **exponentiel**

Algorithm 9 Algorithme pour la résolution de $\mathcal{PSFC}(\subseteq_{tr})$

1: **Resoudre-**$\mathcal{PSFC}(\subseteq_{tr})(\Sigma, Port, At, \mathcal{A}c^{but}, \mathcal{A}c1, \ldots, \mathcal{A}c^n)$
2: $\phi \leftarrow \bot$
3: $\mathcal{A}c \leftarrow \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n$
4: **for** $I_0 \in MaxConst(At)$ **do**
5: $\quad \mathcal{A} \leftarrow Exec(\mathcal{A}c^{but}, I_0)$
6: $\quad \mathcal{A}' \leftarrow Exec(\mathcal{A}c, I_0)$
7: $\quad$ **if** $\mathcal{A} \subseteq_{tr} \mathcal{A}' \quad (\{\epsilon\})$ **then**
8: $\quad\quad \phi' \leftarrow \bigwedge_{r \in I_0} r$
9: $\quad\quad \phi \leftarrow \phi \vee \phi'$
10: $\quad$ **end if**
11: **end for**
12: **retourner** ϕ
13: **end procedure**

par rapport à la taille de At et au nombre n de services disponibles. Les raisons sont identiques à celles utilisées pour déterminer l'espace utilisé par l'algorithme 7. En remplaçant $\subseteq_{tr}$ par $\equiv_{tr}$ dans la ligne 7 de l'algorithme 9, on obtient un algorithme qui permet la résolution du problème $\mathcal{PSFC}(\equiv_{tr})$ en utilisant un espace exponentiel. Par conséquent.

Théorème 7.3.10. *Les problèmes $\mathcal{PSFC}(\subseteq_{tr})$ et $\mathcal{PSFC}(\equiv_{tr})$ sont dans FEXPSPACE.*

Simulation et bisimulation

Les algorithmes que nous proposons, pour la résolution des problèmes $\mathcal{PEE}$, $\mathcal{PDFC}$ et $\mathcal{PSFC}$, lorsque les relations de simulation et de bisimulation sont considérées sont des variantes des algorithmes proposés pour

la résolution de ces problèmes lorsque les relations d'inclusion de traces et d'équivalence de traces sont considérées. La différence est que les algorithmes qui considèrent la simulation et la bisimulation utilisent un temps exponentiel pour retourner une solution, tandis que les algorithmes qui considèrent l'inclusion de traces et l'équivalence de traces utilisent un espace exponentiel.

Considérons un ACC $\mathcal{A}c^{but}$ sur Σ et At et des ACC $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$ sur $\Sigma \cup (\{!, ?\} \times Port)$ et At. L'algorithme 10 retourne la valeur "vrai" ssi il existe un ensemble maximal consistant I_0 tel que : $Exec(\mathcal{A}c^{but}, I_0) \leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$. Pour retourner une solution, l'algorithme 10

Algorithm 10 Algorithme pour la résolution de $\mathcal{PEE}(\leq_{si})$

1: **Resoudre-**$\mathcal{PEE}(\leq_{si})(\Sigma, Port, At, \mathcal{A}c^{but}, \mathcal{A}c1, \ldots, \mathcal{A}c^n)$
2: $\mathcal{A}c \leftarrow \mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n$
3: **for** $I_0 \in MaxConst(At)$ **do**
4: $\quad \mathcal{A} \leftarrow Exec(\mathcal{A}c^{but}, I_0)$
5: $\quad \mathcal{A}' \leftarrow Exec(\mathcal{A}c, I_0)$
6: $\quad$ **if** $\mathcal{A} \leq_{si} \mathcal{A}' \quad (\{\epsilon\})$ **then** **retourner** Vrai
7: $\quad$ **end if**
8: **end for**
9: **retourner** Faux
10: **end procedure**

utilise un temps **exponentiel** par rapport à la taille de At et au nombre n de services disponibles. Les raisons sont les suivantes. (1) L'opération de la ligne 6 nécessite un temps **polynomial** [HS96], par rapport à la taille de $\mathcal{A}$ et $\mathcal{A}'$. (2) La taille de $\mathcal{A}'$ est **exponentielle** par rapport à la taille de n. (3) La boucle For est parcouru $2^{Card(At)}$ fois. En remplaçant $\leq_{si}$ par $\longleftrightarrow_{bi}$ dans la ligne 6 de l'algorithme 10, on obtient un algorithme qui permet la résolution du problème $\mathcal{PEE}(\longleftrightarrow_{bi})$ en utilisant un temps exponentiel. Par conséquent.

Théorème 7.3.11. *Les problèmes $\mathcal{PEE}(\leq_{si})$ et $\mathcal{PEE}(\longleftrightarrow_{bi})$ sont dans EXPTIME.*

Concernant le problème $\mathcal{PDFC}$, nous avons, en plus des ACC $\mathcal{A}c^{but}$ et $\mathcal{A}c^1, \ldots, \mathcal{A}c^n$, une formule booléenne ϕ. L'algorithme 11 retourne la valeur "vrai" ssi pour tout ensemble maximal consistant I_0 on a : $Exec(\mathcal{A}c^{but}, I_0) \leq_{si} Exec(\mathcal{A}c^1 \circledast \ldots \circledast \mathcal{A}c^n, I_0)(\{\epsilon\})$ ssi $\widehat{V_0}(\phi) = 1$. Pour retourner une solution, l'algorithme 11 utilise un temps **exponentiel** par rapport à la taille de At et au nombre n de services disponibles. Les raisons sont identiques à celles

Algorithm 11 Algorithme pour la résolution de $\mathcal{PDFC}(\leq_{si})$

1: **Resoudre-**$\mathcal{PDFC}(\leq_{si})(\Sigma, Port, At, \mathcal{A}c^{but}, \mathcal{A}c1, \ldots, \mathcal{A}c^{n}, \phi)$
2: $Bool \leftarrow Vrai$
3: $\mathcal{A}c \leftarrow \mathcal{A}c^{1} \circledast \ldots \circledast \mathcal{A}c^{n}$
4: **for** $I_0 \in MaxConst(At)$ **do**
5: $Ev \leftarrow \widehat{V_0}(\phi)$
6: $\mathcal{A} \leftarrow Exec(\mathcal{A}c^{but}, I_0)$
7: $\mathcal{A}' \leftarrow Exec(\mathcal{A}c, I_0)$
8: **if** $\mathcal{A} \leq_{si} \mathcal{A}' \quad (\{\epsilon\})$ **then** $Inc \leftarrow Vrai$
9: **else**$Inc \leftarrow Faux$
10: **end if**
11: **if** $Inc \neq Ev$ **then** $Bool \leftarrow Faux$
12: **end if**
13: **end for**
14: **retourner** $Bool$
15: **end procedure**

utilisées pour déterminer le temps utilisé par l'algorithme 10. En remplaçant $\leq_{si}$ par $\longleftrightarrow_{bi}$ dans la ligne 8 de l'algorithme 11, on obtient un algorithme qui permet la résolution du problème $\mathcal{PDFC}(\longleftrightarrow_{bi})$ en utilisant un temps exponentiel. Par conséquent.

Théorème 7.3.12. *Les problèmes $\mathcal{PDFC}(\leq_{si})$ et $\mathcal{PDFC}(\longleftrightarrow_{bi})$ sont dans EXPTIME.*

Concernant le problème $\mathcal{PSFC}$, l'algorithme 12 retourne une formule ϕ telle que pour tout ensemble maximal consistant I_0 on a : $Exec(\mathcal{A}c^{but}, I_0)$ $\leq_{si} Exec(\mathcal{A}c^{1} \circledast \ldots \circledast \mathcal{A}c^{n}, I_0)(\{\epsilon\})$ ssi $\widehat{V_0}(\phi) = 1$. Pour retourner une solution, l'algorithme 12 utilise un temps **exponentiel** par rapport à la taille de At et au nombre n de services disponibles. Les raisons sont identiques à celles utilisées pour déterminer le temps utilisé par l'algorithme 10. En remplaçant $\leq_{si}$ par $\longleftrightarrow_{bi}$ dans la ligne 7 de l'algorithme 12, on obtient un algorithme qui permet la résolution du problème $\mathcal{PSFC}(\longleftrightarrow_{bi})$ en utilisant un temps exponentiel. Par conséquent.

Théorème 7.3.13. *Les problèmes $\mathcal{PSFC}(\leq_{si})$ et $\mathcal{PSFC}(\longleftrightarrow_{bi})$ sont dans FEXPTIME.*

Algorithm 12 Algorithme pour la résolution de $\mathcal{PSFC}(\leq_{si})$

Resoudre-$\mathcal{PSFC}(\leq_{si})(\Sigma, Port, At, \mathcal{A}c^{but}, \mathcal{A}c1, \ldots, \mathcal{A}c^{n})$
$\phi \leftarrow \bot$
$\mathcal{A}c \leftarrow \mathcal{A}c^{1} \circledast \ldots \circledast \mathcal{A}c^{n}$
for $I_0 \in MaxConst(At)$ **do**
  $\mathcal{A} \leftarrow Exec(\mathcal{A}c^{but}, I_0)$
  $\mathcal{A}' \leftarrow Exec(\mathcal{A}c, I_0)$
  if $\mathcal{A} \leq_{si} \mathcal{A}' \quad (\{\epsilon\})$ **then**
    $\phi' \leftarrow \bigwedge_{r \in I_0} r$
    $\phi \leftarrow \phi \vee \phi'$
  end if
end for
retourner ϕ
end procedure

Relation	Existence d'une évaluation	Décision pour une formule canonique
$\subseteq_{tr}$	*EXPSPACE-complet*	*EXPSPACE-complet*
$\equiv_{tr}$	*EXPSPACE-complet*	*EXPSPACE-complet*
$\leq_{si}$	*EXPTIME-complet*	*EXPTIME-complet*
$\longleftrightarrow_{bi}$	*PSPACE-difficile* et dans *EXPTIME*	*PSPACE-difficile* et dans *EXPTIME*

Tab. 7.1 – Tableau récapitulatif dans le cas où la communication entre les services est synchrone.

7.4 Conclusion

Dans ce chapitre, nous avons considéré des services qui peuvent effectuer des actions internes ainsi que des actions de communication. Contrairement aux chapitres précédents les actions de communication sont synchrones. Formellement, cette différence apparaît dans la définition du produit entre automates communicants conditionnels. Nous avons également défini trois problèmes de la composition, à savoir les problèmes de décision $\mathcal{PEE}$ et $\mathcal{PDFC}$ ainsi que le problème de synthèse $\mathcal{PSFC}$. Concernant le problème de synthèse $\mathcal{PSFC}$ nous avons montré qu'il est dans FEXPSPACE (resp. FEXPTIME) lorsque les relations d'inclusion de traces et d'équivalence de traces (resp. simulation et bisimulation) sont considérées. Les résultats de complexité concernant les problèmes de décision sont récapitulés dans le Ta-

bleau 7.1. Nous remarquons que les problèmes $\mathcal{PEE}(\longleftrightarrow_{bi})$ et $\mathcal{PDFC}(\longleftrightarrow_{bi})$ n'ont pas une borne inférieure qui coïncide avec la borne supérieure. La raison est que nous ignorons si le problème de la bisimulation entre un automate fini et un produit d'automates finis est EXPTIME-difficile.

Chapitre 8

Conclusion et perspectives

8.1 Conclusion

Dans ce document, nous avons présenté un modèle formel pour la représentation des services Web. Ce modèle est basé sur les automates communicants conditionnels (ACC). Dans lesquels il est possible d'effectuer des actions de communication ainsi que des actions internes. De plus, l'exécution d'une transition peut exiger que des conditions soient satisfaites et que des effets soient appliqués. Ensuite, nous avons défini et étudié la complexité du problème de la composition pour les relations suivantes : inclusion de traces, équivalence de traces, simulation et bisimulation.

Dans le Chapitre 4, nous avons montré que le problème de la composition est indécidable pour l'équivalence de traces et la bisimulation lorsque les ports de communication sont non-bornés. Ce résultat repose sur une réduction du problème de l'équivalence de traces ou de la bisimulation entre des réseaux de Petri étiquetés. Cependant, nous avons montré que le problème est décidable pour la simulation et l'inclusion de traces. Pour le prouver, nous avons réduit le problème de la composition à un problème de simulation ou d'inclusion de traces entre un réseau de Petri étiqueté et un automate fini.

Suite aux résultats d'indécidablité pour l'inclusion de traces et la bisimulation, nous nous sommes intéressée à trois variantes de notre modèle. Dans la première variante, les services sont représentés par des automates conditionnels (AC) au lieu des ACC. Dans les AC, les actions de communication sont considérées comme des actions internes. En d'autres termes, ce modèle est une abstraction du modèle initial. Dans la seconde variante, les actions de communication sont effectuées à travers des ports bornés. Dans

la troisième variante, contrairement aux variantes précédentes, les actions de communication sont effectuées de façon synchrone. C'est-à-dire, qu'un service ne peut envoyer de message que s'il existe un autre service pour le recevoir. Réciproquement, un service ne peut recevoir de message que s'il existe un service prêt à l'envoyer.

Dans la première variante (voir Chapitre 5), le problème de la composition consiste à déterminer si un automate conditionnel est équivalent à un produit d'automates conditionnels. Nous avons montré que ce problème est EXPSPACE-complet pour l'inclusion de traces et l'équivalence de traces. Afin de montrer qu'il est EXPSPACE-difficile, nous avons utilisé une réduction à partir du problème de l'inclusion de traces ou de l'équivalence de traces entre réseaux de Petri saufs. Lorsque la simulation et la bisimulation sont considérées, nous avons montré que le problème devient EXPTIME-complet. Pour montrer qu'il est EXPTIME-difficile, nous avons utilisé la même réduction que celle utilisée pour l'inclusion de traces et l'équivalence de traces. Concernant la simulation, nous avons utilisé une réduction à partir du problème de la simulation entre un automate et un produit asynchrone d'automates.

Pour montrer que le problème est dans EXPSPACE (resp. EXPTIME) pour l'inclusion de traces et l'équivalence de traces (resp. simulation et bisimulation), nous avons élaboré des algorithmes basés sur des procédures connues permettant la résolution des problèmes d'équivalences entre automates finis. Le Tableau 8.1, récapitule ces résultats.

Problème	Complexité
$\mathcal{PCSC}(\subseteq_{tr})$	*EXPSPACE-complet*
$\mathcal{PCSC}(\equiv_{tr})$	*EXPSPACE-complet*
$\mathcal{PCSC}(\leq_{si})$	*EXPTIME-complet*
$\mathcal{PCSC}(\longleftrightarrow_{bi})$	*EXPTIME-complet*

TAB. 8.1 – Tableau récapitulatif, dans le cas où l'environnement est sans communication.

Dans la seconde variante (voir Chapitre 6), le problème de la composition consiste étant donnés un ACC représentant le but à réaliser et des ACC représentant les services disponibles, déterminer l'existence d'un ACC représentant le médiateur tel que le but est équivalent au produit du médiateur avec les services disponibles. Nous avons montré que ce problème est EXPSPACE-complet pour l'inclusion de traces et EXPTIME-

complet pour la simulation. Afin de prouver que pour l'inclusion de traces le problème est EXPSPACE-difficile, nous avons utilisé une réduction à partir du problème de l'universalité des expressions régulières avec carré. Concernant la simulation, pour montrer que le problème est EXPTIME-difficile, nous avons utilisé une réduction à partir du problème de la simulation entre un automate et un produit asynchrone d'automates. Également, nous avons montré que le problème est EXPSPACE-difficile pour l'équivalence de traces et EXPTIME-difficile pour la bisimulation. Ces résultats reposent sur une réduction à partir du problème de l'équivalence entre réseaux de Petri.

Dans le but de déterminer des algorithmes pour la résolution du problème pour la simulation et l'inclusion de traces, nous avons utilisé la notion du médiateur large. Cela nous a permis d'utiliser les procédures connues permettant la résolution des problèmes d'équivalences entre automates finis. Ne pouvant pas appliquer une approche similaire pour l'équivalence de traces et la bisimulation, nous avons utilisé des procédures basées sur la synthèse de contrôleur. Ce qui nous a permis de montrer que le problème est dans 2-EXPTIME pour la bisimulation. Concernant l'équivalence de traces, le problème peut être réduit à un problème de synthèse de contrôleur augmenté par la notion de masque. La borne inférieure de ce problème est connue, mais nous ignorons sa complexité exacte. Le Tableau 8.2, récapitule ces résultats.

Relation	Complexity
Trace inclusion	*EXPSPACE-complet*
Trace equivalence	*EXPSPACE-difficile*
Simulation	*EXPTIME-complet*
Bisimulation	*EXPTIME-difficile*
	2-EXPTIME

TAB. 8.2 – Tableau récapitulatif, dans le cas où la communication entre les services est asynchrone avec des ports bornés.

Dans la troisième variante (voir Chapitre 7), les problèmes de la composition considérés se focalisent sur la construction ou l'existence d'une formule, au lieu de se focaliser sur l'existence ou la synthèse d'un médiateur. Cette formule lorsqu'elle est satisfaite garantit que le système composé des services disponibles est équivalent au service but. Plus précisément, nous nous sommes intéressée aux problèmes suivants : existence d'une évaluation, décision pour une formule canonique et synthèse d'une formule canonique. Concernant le problème de la synthèse, nous avons montré qu'il est

dans FEXPSPACE (resp. FEXPTIME) pour l'inclusion de traces (resp. équivalence de traces).

Nous avons montré que le problème de l'existence d'une évaluation est EXPSPACE-complet pour l'inclusion de traces et l'équivalence de traces et EXPTIME-complet pour la simulation. Les approches utilisées sont similaires à celles que nous avons utilisées pour la seconde variante, à savoir des réductions à partir du problème de l'universalité des expressions régulières avec carré et du problème de la simulation entre un automate et un produit asynchrone d'automates. En ce qui concerne la bisimulation, nous avons montré que le problème est PSPACE-difficile au moyen d'une réduction à partir du problème de l'acceptance d'une machine de Turing déterministe linéairement bornée en espace. De plus, le problème est dans EXPTIME. Notre conjecture est que le problème est EXPTIME-difficile. Cependant, pour le prouver il faut d'abord résoudre le problème suivant : prouver que le problème de savoir si un automate fini est bisimilaire à un produit d'automates finis est EXPTIME-difficile. À notre connaissance, la complexité exacte de ce problème n'est pas connue.

Nous avons obtenu les mêmes résultats pour le problème de décision pour une formule canonique. Tous les algorithmes que nous avons utilisés sont basés sur les procédures permettant la résolution des problèmes d'équivalences entre automates finis. Le Tableau 8.3, récapitule ces résultats.

Relation	Existence d'une évaluation	Décision pour une formule canonique
$\subseteq_{tr}$	*EXPSPACE-complet*	*EXPSPACE-complet*
$\equiv_{tr}$	*EXPSPACE-complet*	*EXPSPACE-complet*
$\leq_{si}$	*EXPTIME-complet*	*EXPTIME-complet*
$\longleftrightarrow_{bi}$	*PSPACE-difficile* et dans *EXPTIME*	*PSPACE-difficile* et dans *EXPTIME*

Tab. 8.3 – Tableau récapitulatif dans le cas où la communication entre les services est synchrone.

8.2 Bilan

Dans un premier temps, nous avons considéré le problème de la composition avec des ports non bornés. Suite aux résultats d'indécidabilité de ce problème pour la bisimulation et l'équivalence de traces, nous avons

considéré trois variantes pour modéliser les services. Ces différentes variantes nous ont permis de considérer différentes définitions pour le problème de la composition.
La première variante nous permet de considérer le problème de la composition comme un problème d'équivalence entre deux systèmes.

La seconde variante nous permet de considérer le problème de la composition comme le problème de satisfaction d'une formule logique. L'avantage de cette approche et qu'elle permet d'exprimer, en plus de l'équivalence entre deux systèmes, le fait qu'un système vérifie certaines propriétés.

Enfin, dans la dernière variante, le problème de la composition est caractérisé par une formule logique. Cette dernière exprime des exigences que le client doit satisfaire. Ces exigences concernent les certificats qu'il doit posséder.

Ainsi, nous pouvons constater que le problème de la composition à l'avantage de se réduire à des problèmes de différents domaines. Cela permet d'exploiter les résultats connus dans ces derniers.

Cependant, les résultats que nous avons obtenus prouvent que le problème de la composition est difficile à résoudre. Cette difficulté réside dans le fait d'utiliser des automates communicants conditionnels et de calculer explicitement le produit des services. En effet, nous avons montré dans les Chapitre 4, 5 et 6 qu'il est possible d'associer à un automate communicant conditionnel un réseau de Petri dont l'exécution est isomorphe à celle de l'automate communicant conditionnel. Il est connu que le problème de vérifier si deux réseaux de Petri sont équivalents est un problème difficile à résoudre [JM96, Jan95].

De plus, notre résultat est prévisible sachant que le problème de la vérification de modèle (*model cheking*) devient également difficile lorsque le produit d'automates est considéré [DLS06].

Néanmoins, nous avons prouvé grâce aux bornes inférieures qu'il n'est pas possible de trouver de meilleurs résultats, sauf éventuellement pour :

- L'équivalence de traces et la bisimulation, pour la deuxième variante du modèle.
- La bisimulation, pour la troisième variante du modèle.

8.3 Travaux futurs

8.3.1 Problèmes ouverts

Dans un premier temps, nous envisageons de déterminer la complexité exacte du problème de la composition pour la seconde variante, lorsque la

bisimulation est considérée. Pour cela, il faudra trouver le moyen de ne pas calculer le produit des services disponibles, qui est de taille exponentielle par rapport au nombre de services.

Ensuite, pour la même variante, il faudra élaborer un algorithme pour la résolution du problème de la composition, lorsque l'équivalence de traces est considérée. Pour cela, il faudra d'abord résoudre le problème de la synthèse de contrôleur augmenté par des masques. Nous savons que ce problème est EXPSPACE-difficile et nous espérons prouver qu'il est EXPSPACE-complet.

Le dernier problème que nous n'avons pas résolu concerne la troisième variante, plus précisément le cas où la bisimulation est considérée. Nous avons montré que le problème est PSPACE-difficile est qu'il et dans EXPTIME. Cependant, nous aimerions prouver qu'il est EXPTIME-complet. Pour cela, il faut prouver que le problème de savoir si un automate fini est bisimilaire à un produit d'automates finis est EXPTIME-difficile.

Il serait également intéressant d'utiliser des heuristiques pour obtenir des algorithmes efficaces. L'objectif serait de ne pas calculer explicitement le produit des services. Enfin, nous aimerions considérer le cas particulier où les services sont déterministes et de comparer les résultats que nous obtiendrons avec les résultats actuels.

8.3.2 Variation sur le niveau d'abstraction du modèle

Il existe plusieurs variantes intéressantes de notre modèle. Parmi elles, nous distinguons deux catégories. Dans la première catégorie, le but est d'abstraire le modèle afin de réduire la complexité du problème de la composition. Dans la seconde catégorie, le but est de raffiner le modèle, de sorte à le rendre plus proche du modèle réel des services.

Considérons la première catégorie. Il est clair que pour notre modèle des services, le problème de la composition tel que nous l'avons défini est de complexité élevée. Afin de la réduire, nous comptons utiliser des méthodes symboliques. Pour cela, il faudra d'abord élaborer une méthode basée sur l'abstraction et le raffinement adéquate pour la composition de services, en s'inspirant des méthodes utilisées pour la vérification de systèmes [CGJ+03]. Par conséquent, il faudra déterminer un nouveau modèle pour les services et éventuellement une nouvelle définition du problème de la composition. L'idéal serait de trouver des propriétés que le modèle abstrait doit satisfaire, afin de garantir l'existence d'une solution dans le modèle raffiné qui lui est associé. Bien-entendu, la construction du modèle abstrait ainsi que la vérification des propriétés doivent être moins coûteuses que la composition dans le modèle actuel.

Considérons la seconde catégorie qui a pour objectif d'obtenir un modèle proche du modèle réel des services. Dans notre modèle, la communication s'effectue par un échange de messages vides. C'est-à-dire, nous savons seulement qu'un message est transmis ou reçu à travers un port, mais le message n'a pas de contenu. Hors, dans les services Web utilisés dans la pratique, le contenu d'un message est un fichier XML. Afin de se rapprocher de ce modèle, nous envisageons de représenter les messages par des prédicats de la logique du premier ordre. De plus, cela nous permettra par la suite de considérer des messages cryptés [CMR08] pour garantir la sécurité de nos services.

Par ailleurs, dans notre modèle, les conditions sur les transitions sont des variables propositionnelles. Ces conditions peuvent représenter le fait d'avoir confiance ou pas en un autre service [CGM06]. Cependant, cette valeur peut être différente de 0 et 1 et surtout elle peut évoluer selon le comportement des services [DHHvdT04, BJWW02, BNP08]. Un des travaux qui nous intéressent serait de prendre en compte la mise à jour de la confiance lors de la composition de services.

Cependant, pour éviter l'explosion combinatoire de la composition, lorsque nous considérons la notion de confiance ou des messages avec un contenu, il faut considérer des contraintes pour simplifier le modèle. Les simplifications possibles sont : fixer le nombre de services disponibles, fixer la taille du médiateur, considérer des services disponibles sans cycles, etc.

8.3.3 Conditions temporisées

Dans le modèle de services que nous considérons, les préconditions et les effets sont représentés par des variables propositionnelles. Cependant, il serait intéressant de considérer des conditions temporisées. En effet, pour éviter qu'un service ne soit bloqué en attente d'un message, il est possible de lui fixer un délai d'attente au-delà duquel on l'oblige à terminer. Il existe des modèles formels qui prennent en considération la notion du temps. Parmi ces modèles, nous citons les automates temporisés [AD94], les réseaux de Petri temporisés [Ram74] et les réseaux de Petri temporels [Mei74].

Actuellement, il existe des travaux concernant la composition en utilisant des automates temporisés [GPR08, KPP06]. Dans l'approche proposée dans [KPP06], l'objectif est de modéliser et d'analyser des propriétés temporelles. Par exemple, ces propriétés peuvent représenter le temps nécessaire à des services pour réaliser leurs tâches. Pour exprimer ces propriétés, les auteurs utilisent le calcul des durées [CHR91]. Cependant, dans l'approche proposée dans [GPR08], l'objectif est de vérifier la compatibilité des services

afin de les composer. C'est-à-dire, vérifier si les contraintes temporelles entre deux services qui veulent collaborer sont inconsistantes. Pour cela un service expose, en plus des actions qu'il peut effectuer, les contraintes temporelles sur ces actions.

8.3.4 Composition dynamique

Dans cette thèse, nous avons défini une composition statique des services. En effet, l'ensemble des services disponibles ne change pas et les services sont supposés parfaits. Par exemple, il n'y a pas de notion de délai temporel grâce auquel il est possible de vérifier l'échec d'une transition.

La composition dynamique [ZNB+08] consiste à déterminer le service médiateur pendant le déploiement des services. Cette composition doit prendre en compte la modification de l'environnement. Par exemple, des services peuvent ne plus être disponibles, pour différentes raisons. Par conséquent, le médiateur doit être synthétisé en fonction des actions effectivement exécutées par le client.

Il serait également intéressant de gérer l'échec de la composition de façon moins radicale que ce qu'on fait actuellement. Au lieu de répondre par oui ou non selon l'existence du médiateur, il faut expliquer la raison de l'échec et proposer une solution éventuellement partielle qui peut satisfaire le client. Pour cela, une notion de préférence [SPM06] du client pourrait être envisageable. Chaque action aurait un poids selon son importance du point de vue du client et l'objectif serait de maximiser la satisfaction du client.

Annexe 1

Le but de cette annexe est de prouver le lemme 6.5.8. Dans lequel la fonction Del° utilisée est celle de la définition 6.5.4. Afin de prouver le lemme 6.5.8, nous allons prouver un résultats plus général. En effet, au lui d'utiliser les ensembles fini d'actions Σ, $\{!,?\} \times Port$, $\{!,?\} \times Port'$ et $\{!,?\} \times Port^\circ$, nous utilisons les ensembles d'actions Σ et $\Sigma', \lambda \subseteq \Sigma$. L'ensemble Σ' représente les actions non observables, lorsque la bisimulation est effectuée et λ représente l'ensemble des actions pour lequel nous créons des copies. Par conséquent, nous devons définir la copie d'un ensemble d'actions que nous devons considérer dans la définition de la fonction Del°.

Dans ce qui suit, nous donnons la définition de la copie d'un ensemble et celle de Del°. Ensuite, nous proposons et prouvons une proposition qui généralise le lemme 6.5.8.

Définition 8.3.1 (**Copies d'un ensemble relativement à un autre ensemble**). Soit $\Sigma = \{a_1, \ldots, a_t\}$ et $\lambda \subseteq \Sigma$ des ensembles d'actions. L'ensemble $\Sigma^\circ_\lambda = \{a_1^\circ, \ldots, a_t^\circ\}$ des copies de Σ relativement à λ est défini tel que :

- pour tout $a \in \Sigma \setminus \lambda$, $a^\circ = a$ et
- pour tout $a \in \lambda$, $a^\circ \neq a$.

La copie d'un ensemble d'actions nous permet de faire le renomage de certains de ses éléments. Pour le problème de la composition ce qui nous intéresse c'est de renommer les actions de l'ensemble $\{!,?\} \times Port$.

Définition 8.3.2 (**Fonction** Del°). Soit $\Sigma, \lambda \subseteq \Sigma$ des ensembles d'actions et Σ°_λ l'ensemble des copies de Σ relativement à λ. Soit $\mathcal{A} = (Q, q_0, \rightarrow)$ un automate sur $\Sigma \cup \Sigma^\circ_\lambda$. $Del^\circ(\mathcal{A}, \lambda) = (Q', q'_0, \rightarrow')$ est un automate sur Σ tel que :

- $Q' = Q$,
- $q'_0 = q_0$,

– $\rightarrow' \subseteq Q' \times \Sigma \times Q'$ est la relation de transitons définie par $q \rightarrow^{\{a\}}_{Del^\circ(\mathcal{A})} q'$ ssi $q \rightarrow^{\{a^\circ\}}_{\mathcal{A}} q'$.

Lorsque $\lambda = \{!, ?\} \times Port$ la fonction Del° nous permet de renommer les actions dans $\{!, ?\} \times Port^\circ$ par des actions dans $\{!, ?\} \times Port$. Ainsi, la définition de $Del^\circ(\mathcal{A}, \{!, ?\} \times Port)$ sera celle donné dans la définition 6.5.4. Considérons la proposition.

Proposition 8.3.3. *Soient les ensembles Σ, $\Sigma', \lambda \subseteq \Sigma$ et un automate fini $\mathcal{A}$ sur Σ. Considérons l'ensemble λ° des copies de λ. Il est possible de construire une formule $\varphi(\mathcal{A}, \Sigma')$ du μ-calcul telle que pour tout automate fini $\mathcal{A}'$ sur $\Sigma \cup \lambda^\circ$ l'assertion suivante est satisfaite :*

$$\mathcal{A} \longleftrightarrow_{bi} Del^\circ(\mathcal{A}', \lambda) \quad (\Sigma') \text{ ssi } \mathcal{A}' \models \varphi(\mathcal{A}, \Sigma').$$

Il est clair que si nous prouvons la proposition 8.3.3, il suffira de remplacer Σ par $\Sigma \cup (\{!, ?\} \times Port)$, Σ' par $\{!, ?\} \times Port'$ et λ° par $\{!, ?\} \times Port^\circ$, pour obtenir la preuve du lemme 6.5.8. De plus, nous constaterons que la formule $\varphi(\mathcal{A}, \Sigma')$ est de taille polynomiale relativement à la taille de $\mathcal{A}$ et Σ'. Dans un premier temps, nous rappelons quelques notions du μ-calcul. Ensuite, nous proposons une méthode pour la construction de la formule $\varphi(\mathcal{A}, \Sigma')$ de la proposition. Nous terminerons en prouvant l'assertion de la proposition. Nous utilisons les notations suivantes :

– $q \overset{a,\Sigma'}{\rightsquigarrow}_{\mathcal{A}} q'$ lorsque :
$$q \rightarrow^{\Sigma'^\star}_{\mathcal{A}} \circ \rightarrow^{\{a\}}_{\mathcal{A}} \circ \rightarrow^{\Sigma'^\star}_{\mathcal{A}} q'$$
– $q \overset{a,\Sigma'}{\not\rightsquigarrow}_{\mathcal{A}} q'$ lorsqu'ils n'existent pas $q_1, q_2 \in Q$ tels que :
$$q \rightarrow^{\Sigma'^\star}_{\mathcal{A}} q_1 \rightarrow^{\{a\}}_{\mathcal{A}} q_2 \rightarrow^{\Sigma'^\star}_{\mathcal{A}} q'$$
– $Succ(q, a) = \{q' \in Q \mid q \rightarrow^{\{a\}}_{\mathcal{A}} q'\}$.

Définitions pour le μ-calcul

μ-calcul, syntaxe et sémantique

Définition 8.3.4 (Syntaxe du μ-calcul)**.** *Soit $Var = \{X, X_0, X_1, X_2, \ldots\}$ un ensemble fini de variables . L'ensemble des formules du μ-calcul est noté L_μ et il est défini à partir de la grammaire suivante :*

$$\beta := \texttt{true} \mid X \mid \langle a \rangle \beta_1 \mid \neg \beta_1 \mid \beta_1 \vee \beta_2 \mid \mu X.\beta_1(X)$$

où $a \in \Sigma$. Afin de garantir l'existence du point fixe, nous imposant, pour toute les formules $\mu X.\beta_1(X)$, que dans $\beta_1(X)$ la variable X soit dans le champ d'un nombre paire de symboles de négations $\neg$.

Nous utilisons les notations usuelles $\texttt{false}, [a]\beta_1, \beta_1 \wedge \beta_2, \beta_1 \Rightarrow \beta_2, \rightarrow^a$, $\not\rightarrow^a$ et $\nu X.\beta_1(X)$ pour les formules respectives $\neg\texttt{true}, \neg\langle a\rangle(\neg\beta_1), \neg(\neg\beta_1 \vee \neg\beta_2), \neg\beta_1 \vee \beta_2, \langle a\rangle\texttt{true}, [a]\texttt{false}$ et $\neg\mu X.\neg\beta_1(\neg X)$.
Une *sentence* est une formule du μ-calcul sans aucune variable libre : chaque variable X est dans le champ d'un opérateur μX ou νX.

L'interprétation d'une formule du μ-calcul sur un automate est le sous ensemble d'états de l'automate qui satisfont la formule relativement à une interprétation *val* des variables libres de la formule.

Définition 8.3.5 (Sémantique du μ-calcul)**.** *Une formule $\beta \in L_\mu$ est interprétée sur un automate $\mathcal{A} = (Q, q_0, \rightarrow)$ défini sur Σ relativement à une interprétation $val : Var \rightarrow 2^Q$, par l'ensemble $[\![\ \beta\ ]\!]^{val}_{\mathcal{A}} \subseteq Q$ qui est définit inductivement comme suit :*

$$
\begin{array}{l}
[\![\ \texttt{true}\ ]\!]^{val}_A = Q \\
[\![\ X\ ]\!]^{val}_A = val(X) \\
[\![\ \neg\beta\ ]\!]^{val}_A = Q \setminus [\![\ \beta\ ]\!]^{val}_A \\
[\![\ \beta_1 \vee \beta_2\ ]\!]^{val}_A = [\![\ \beta_1\ ]\!]^{val}_A \cup [\![\ \beta_2\ ]\!]^{val}_A \\
[\![\ \langle a\rangle\beta\ ]\!]^{val}_A = \{q \in Q \mid \exists q' : q' \in Succ(q,a) \text{ et } q' \in [\![\ \beta\ ]\!]^{val}_A\} \\
[\![\ \mu X.\beta(X)\ ]\!]^{val}_A = \cap\{V \subseteq Q \mid [\![\ \beta\ ]\!]^{[val(V/X)]}_A \subseteq V\}
\end{array}
$$

où $val(V/X) : Var \rightarrow 2^Q$ est la valuation $val(V/X)(X') = val(X')$ pour toute variable $X' \in Var$ telle que $X' \neq X$ et $val(V/X)(X) = V$.

Ainsi, l'interprétation $[\![\ \mu X.\beta(X)\ ]\!]^{val}_A$ (resp. $[\![\ \nu X.\beta(X)\ ]\!]^{val}_A$) est le plus petit (resp. grand) point fixe de la fonction de 2^Q dans 2^Q qui associe à U l'ensemble $[\![\ \beta(X)\ ]\!]^{val(U/X)}_{\mathcal{A}}$.
Observons que la sémantique des sentences du L_μ est indépendante de la valuation *val*. Nous notons par $[\![\ \beta\ ]\!]_{\mathcal{A}}$ l'interprétation d'une sentence β relativement à toute les valuations. Nous dirons que l'automate $\mathcal{A}$ *satisfait* la sentence β (noté $\mathcal{A} \models \beta$) si l'état initial q_0 de $\mathcal{A}$ appartient à $[\![\ \beta\ ]\!]_{\mathcal{A}}$.

μ-calcul vectoriel

Nous considérons une version restreinte pour le concept des systèmes d'équations du L_μ. Pour plus de détail le lecteur peut se référer à [AN01]. Soit m un entier et $\psi_1, \ldots \psi_m$ un ensemble de formules du L_μ avec $X_1, \ldots X_m$ comme variables libres. Le système d'équations que nous considérons est le

suivant :

$$\begin{array}{rcl} X_1 & = & \psi_1(X_1, \ldots X_m) \\ \vdots & & \vdots \\ X_m & = & \psi_m(X_1, \ldots X_m) \end{array}$$

Dans ce document, un tel système est interprété pour un automate $\mathcal{A} = (Q, q_0, \rightarrow)$ défini sur Σ comme un sous ensemble de $(2^Q)^m$. Une solution est un vecteur $< V_1, \ldots, V_m >$ de $(2^Q)^m$. Dans l'ensemble des solutions, nous considérons uniquement la plus grande. Donc, ce système peut être exprimé comme un plus grand point fixe :

$< V_1, \ldots, V_m > = [\![\nu < X_1, \ldots X_m > . < \psi_1, \ldots, \psi_m > (< X_1, \ldots, X_m >)]\!]_{\mathcal{A}}$

Ce système peut être résolu d'une manière symbolique est indépendamment d'une interprétation en utilisant le principe d'élimination de Gauss. Nous pouvons construire m sentences $\Psi_1, \ldots, \Psi_m$ de L_μ, tels que pour tout automate $\mathcal{A} = (Q, q_0, \rightarrow)$ sur Σ et pour tout $1 \leq i \leq m$, $[\![\Psi_i]\!]_{\mathcal{A}}$ est la $i^{\text{ème}}$ composante de la solution du système interprété sur $\mathcal{A}$. Par conséquent, $[\![\Psi_i]\!]_{\mathcal{A}} = V_i$. Nous notons :

$< \Psi_1, \ldots, \Psi_m > = \nu < X_1, \ldots, X_m > . < \psi_1, \ldots, \psi_m > (< X_1, \ldots, X_m >)$

Proposition 8.3.6 (Principe d'élimination de Gauss)**.** *Soient ψ_1 et ψ_2 des formules de L_μ avec X_1 et X_2 comme variables libres, soit $\phi_1(X_2) = \nu X_1.\psi_1(X_1, X_2)$ et soit*

$$< \Psi_1, \Psi_2 >= \nu < X_1, X_2 > . < \psi_1, \psi_2 > (< X_1, X_2 >).$$

Alors $\Psi_2 = \nu X_2.\psi_2(\phi_1(X_2), X_2)$ et $\Psi_1 = \phi_1(\Psi_2)$.

Cette proposition est une version restreinte du principe d'élimination de Gauss comme il est présenté dans [AN01] puisque elle concerne un μ-calcul donné et seulement les plus grands points fixes. La proposition permet de calculer récursivement $< \Psi_1, \ldots, \Psi_m > = \nu < X_1, \ldots X_m > . < \psi_1, \ldots, \psi_m > (< X_1, \ldots, X_m >)$

en utilisant l'algorithme suivant :

1. Calculer $\phi_1(X_2, \ldots X_m) = \nu X_1.\psi_1(X_1, \ldots, X_m)$,
2. Calculer $< \Psi_2, \ldots, \Psi_m > = \nu < X_2, \ldots X_m > . < \psi_2, \ldots \psi_m > (< \phi_1(X_2, \ldots X_m), X_2 \ldots, X_m >)$ en utilisant le même méthode,
3. Calculer $\Psi_1 = \phi_1(\Psi_2, \ldots, \Psi_m)$.

Expression d'une sentence pour la bisimulation modulo Σ'

Soient les ensembles $\Sigma = \{a_1, \ldots, a_t\}, \Sigma' \subseteq \Sigma$, $\lambda \subseteq \Sigma$ et $\Sigma^\circ_\lambda = \{a^\circ_1, \ldots, a^\circ_t\}$ l'ensemble des copies de Σ relativement à λ. Soit $\beta \in L_\mu$, val une valuation sur les variables de β, $A = (Q, q_0, \rightarrow)$ un automate fini sur Σ.

Dans ce qui suit, nous notons par $\langle \Sigma'^* \rangle \beta$ les formules $\mu X.\langle \Sigma' \rangle X \vee \beta$. De façon duale, nous notons par $[\Sigma'^*]\beta$ les formules $\nu X.[\Sigma']X \wedge \beta$. plus précisément :

$$\begin{array}{lcl} [\![\langle \Sigma'^* \rangle \beta]\!]^{val}_{\mathcal{A}} & = & \{q \in Q \mid \exists q' \in Q, q \xrightarrow{\Sigma'^*}_{\mathcal{A}} q' \text{ et } q' \in [\![\beta]\!]^{val}_{\mathcal{A}}\} \\ [\![[\Sigma'^*] \beta]\!]^{val}_{\mathcal{A}} & = & \{q \in Q \mid \forall q' \in Q, q \not\xrightarrow{\Sigma'^*}_{\mathcal{A}} q' \text{ ou } q' \in [\![\beta]\!]^{val}_{\mathcal{A}}\} \end{array}$$

Ainsi, nous avons également pour tout $a \in \Sigma$:

$$\begin{array}{lcl} [\![\langle \Sigma'^* \rangle \langle a \rangle \langle \Sigma'^* \rangle \beta]\!]^{val}_{\mathcal{A}} & = & \{q \in Q \mid \exists q' \in Q, q \overset{a,\Sigma'}{\rightsquigarrow}_{\mathcal{A}} q' \text{ et } \\ & & q' \in [\![\beta]\!]^{val}_{\mathcal{A}}\} \\ [\![[\Sigma'^*][a][\Sigma'^*] \beta]\!]^{val}_{\mathcal{A}} & = & \{q \in Q \mid \forall q' \in Q, q \not\overset{a,\Sigma'}{\rightsquigarrow}_{\mathcal{A}} q' \text{ ou } \\ & & q' \in [\![\beta]\!]^{val}_{\mathcal{A}}\} \end{array}$$

$$\begin{array}{lcl} [\![\langle \Sigma'^* \rangle \langle a \cup a^\circ \rangle \langle \Sigma'^* \rangle \beta]\!]^{val}_{\mathcal{A}} & = & \{q \in Q \mid \exists q' \in Q, (q \overset{a,\Sigma'}{\rightsquigarrow}_{\mathcal{A}} q' \\ \text{ ou } q \overset{a^\circ,\Sigma'}{\rightsquigarrow}_{\mathcal{A}} q') \text{ et } (q' \in [\![\beta]\!]^{val}_{\mathcal{A}})\} & & \\ [\![[\Sigma'^*][a \cup a^\circ][\Sigma'^*] \beta]\!]^{val}_{\mathcal{A}} & = & \{q \in Q \mid \forall q' \in Q, (q \not\overset{a,\Sigma'}{\rightsquigarrow}_{\mathcal{A}} q' \\ \text{ et } q \not\overset{a^\circ,\Sigma'}{\rightsquigarrow}_{\mathcal{A}} q') \text{ ou } (q' \in [\![\beta]\!]^{val}_{\mathcal{A}})\} & & \end{array}$$

Soit $\mathcal{A} = (Q, q_0, \rightarrow)$ un automate fini sur Σ avec $Q = \{q_0, q_1, \ldots q_m\}$. Nous définissons la sentence $\varphi(\mathcal{A}, \Sigma')$ associée à $\mathcal{A}$ et Σ'par $\varphi(\mathcal{A}, \Sigma') = \Psi_0$ dans l'expression suivante :

$$< \Psi_0, \ldots, \Psi_m > = \nu < X_0, \ldots, X_m > . < \psi_0, \ldots, \psi_m >$$

où, pour tout $0 \leq i \leq m$ nous avons :

$$\psi_i(X_0, \ldots, X_m) = \bigwedge_{a,j/q_j \in Succ(q_i,a)} \langle \Sigma'^* \rangle \langle a \cup a^\circ \rangle \langle \Sigma'^* \rangle X_j$$
$$\wedge \bigwedge_{a/Succ(q_i,a)=\emptyset} [\Sigma'^*] \not\rightarrow_a \wedge \bigwedge_{a/Succ(q_i,a)=\emptyset} [\Sigma'^*] \not\rightarrow_{a^\circ}$$
$$\wedge \bigwedge_{a/Succ(q_i,a)\neq\emptyset} ([\Sigma'^*][a \cup a^\circ][\Sigma'^*] \bigvee_{j/q_j \in Succ(q_i,a)} X_j)$$

Nous pouvons maintenant prouver la proposition 8.3.3. Sans perte de généralités, nous supposons que $\mathcal{A}$ ne contient aucune transition étiquetée par une action dans Σ'.

Preuve de la proposition 8.3.3. Soit $\mathcal{A} = (Q, q_0, \rightarrow_{\mathcal{A}})$ un automate fini sur Σ, soit $\varphi(\mathcal{A}, \Sigma')$ la sentence définie ci-dessus.

($\Rightarrow$) Pour tout automate $\mathcal{A}' = (Q', q'_0, \rightarrow_{\mathcal{A}'})$ sur $\Sigma \cup \Sigma^\circ$ tel que $\mathcal{A}' \models \varphi(\mathcal{A}, \Sigma')$, nous montrons que $Del^\circ(\mathcal{A}', \lambda)$ et $\mathcal{A}$ sont bisimilaire modulo Σ'. Soit $V_i = [\![\Psi_i]\!]_{\mathcal{A}'}$ et considérons la relation $Z : Q \times Q'$ tel que $q_i Z q'_i$ ssi $q'_i \in V_i$. Nous prouvons que Z est une bisimulation modulo Σ'. Concernant les étas initiaux, puisque $\mathcal{A}' \models \varphi(\mathcal{A}, \Sigma')$ et $\varphi(\mathcal{A}, \Sigma') = \Psi_0$, nous avons $q'_0 \in V_0$ alors $q_0 Z q'_0$. Soit $q_i \in Q$ et $q'_i \in Q'$ tel que $q_i Z q'_i$. Considérons une action $a \in \Sigma$.

- Soit $q_j \in Q$ tel que $q_i \rightarrow_{\mathcal{A}}^{\{a\}} q_j$. Sachant que $q'_i \in V_i$ (par définition de Z) et $q_i \overset{a,\Sigma'}{\rightsquigarrow}_{\mathcal{A}} q_j$, nous avons :

$$q'_i \in [\![\langle \Sigma'^* \rangle \langle a \cup a^\circ \rangle \langle \Sigma'^* \rangle X_j]\!]_{\mathcal{A}'}^{val(V_j/X_j)}$$

  ce qui implique l'existence de $q'_j \in V_j$ tel que $q'_i \overset{a,\Sigma'}{\rightsquigarrow}_{\mathcal{A}'} q'_j$ ou $q'_i \overset{a^\circ,\Sigma'}{\rightsquigarrow}_{\mathcal{A}'} q'_j$. Ainsi, $q'_i \rightsquigarrow_{Del^\circ(\mathcal{A}',\lambda)}^{a,\Sigma'} q'_j$. Puisque $q'_j \in V_j$, alors on obtient que $q_j Z q'_j$.
- Soit $q'_j \in Q'$ tel que $q'_i \rightsquigarrow_{Del^\circ(\mathcal{A}',\lambda)}^{a,\Sigma'} q'_j$. Deux cas sont possibles, $Succ(q_i, a) = \emptyset$ ou

$$q'_i \in [\![([\Sigma'^*][a \cup a^\circ][\Sigma'^*] \bigvee_{j/q_j \in Succ(q_i,a)} X_j)]\!]_{\mathcal{A}'}^{val(V_0/X_0,\ldots,V_m/X_m)}$$

  Dans le cas où $Succ(q_i, a) = \emptyset$ on obtient $q'_i \in [\![[\Sigma'^*] \not\rightarrow_a]\!]_{\mathcal{A}'}$ et $q'_i \in [\![[\Sigma'^*] \not\rightarrow_{a^\circ}]\!]_{\mathcal{A}'}$. Cela implique $q'_i \overset{a,\Sigma'}{\not\rightarrow}_{\mathcal{A}'} q'_j$ et $q'_i \overset{a^\circ,\Sigma'}{\not\rightarrow}_{\mathcal{A}'} q'_j$, par conséquent $q'_i \not\rightarrow_{Del^\circ(\mathcal{A}',\lambda)}^{a,\Sigma'} q'_j$, ce qui est une contradiction. Dans le cas où

$$q'_i \in [\![([\Sigma'^*][a \cup a^\circ][\Sigma'^*] \bigvee_{j/q_j \in Succ(q_i,a)} X_j)]\!]_{\mathcal{A}'}^{val(V_0/X_0,\ldots,V_m/X_m)}$$

  on obtient

$$q'_i \in [\![([\Sigma'^*][a][\Sigma'^*] \bigvee_{j/q_j \in Succ(q_i,a)} X_j)]\!]_{Del^\circ(\mathcal{A}',\lambda)}^{val(V_0/X_0,\ldots,V_m/X_m)}$$

  . Sachant que $q'_i \rightsquigarrow_{Del^\circ(\mathcal{A}',\lambda)}^{a,\Sigma'} q'_j$ alors il existe $q_j \in Q$ tel que $q'_j \in V_j$. Par définition de Z on obtient $q_j Z q'_j$.

($\Leftarrow$) Soit $\mathcal{A}' = (Q', q'_0, \rightarrow_{\mathcal{A}'})$ un automate fini sur $\Sigma \cup \Sigma'$ tel que $Del^\circ(\mathcal{A}', \lambda)$ et $\mathcal{A}$ sont bisimilaire modulo Σ'. Soit Z une bisimulation modulo Σ' entre $Del^\circ(\mathcal{A}', \lambda)$ et $\mathcal{A}$ et soit $V_0, \ldots, V_m$ tel que $V_i \subseteq Q'$ et $V_i = \{q'_1 \in Q' \mid q_i Z q'_i\}$. Pour montrer que $\mathcal{A}' \models \varphi(\mathcal{A}, \Sigma')$, nous montrons que $< V_0, \ldots, V_m >$ est un point pre-fixe de $< \psi_0, \ldots, \psi_m > (X_0, \ldots, X_m)$, c'est à dire pour tout $1 \leq i \leq m$, nous avons :

$$V_i \subseteq [\![\psi_i]\!]^{val(V_0/X_0, \ldots, V_n/X_n)}_{\mathcal{A}'}$$

Pour tout $q'_i \in V_i$.

- Soit a et j tel que $q_j \in Succ(q_i, a)$. Puisque $q_i Z q'_i$, alors il existe q'_j tel que $q'_i \rightsquigarrow^{a,\Sigma'}_{Del^\circ(\mathcal{A}',\lambda)} q'_j$ et $q_j Z q'_j$. Ce qui implique $q'_i \overset{a,\Sigma'}{\rightsquigarrow}_{\mathcal{A}'} q'_j$ ou $q'_i \overset{a^\circ,\Sigma'}{\rightsquigarrow}_{\mathcal{A}'} q'_j$ et $q_j Z q'_j$. Alors nous avons $q'_j \in V_j$ et donc $q'_i \in [\![\langle \Sigma'^* \rangle \langle a \cup a^\circ \rangle \langle \Sigma'^* \rangle X_j]\!]^{val(V_j/X_j)}_{\mathcal{A}'}$.
- Soit a tel que $Succ(q_i, a) = \emptyset$. Puisque $q_i Z q'_i$, alors nous avons $q'_i \not\rightsquigarrow^{a,\Sigma'}_{Del^\circ(\mathcal{A}',\lambda)}$. Ce qui implique $q'_i \overset{a,\Sigma'}{\not\rightsquigarrow}_{\mathcal{A}'}$ et $q'_i \overset{a^\circ,\Sigma'}{\not\rightsquigarrow}_{\mathcal{A}'}$ et donc $q'_i \in [\![[\Sigma'^*] \not\rightarrow_a \wedge [\Sigma'^*] \not\rightarrow_{a^\circ}]\!]_{\mathcal{A}'}$.
- Soit a et q'_j tel que $q'_i \rightsquigarrow^{a,\Sigma'}_{Del^\circ(\mathcal{A}',\lambda)} q'_j$. On a $q'_i \overset{a,\Sigma'}{\rightsquigarrow}_{\mathcal{A}'} q'_j$ ou $q'_i \overset{a^\circ,\Sigma'}{\rightsquigarrow}_{\mathcal{A}'} q'_j$. Puisque $q_i Z q'_i$, alors il existe un état $q_j \in Q$ tel que $q_j \in Succ(q_i, a)$. Donc

$$q'_i \in [\![[\Sigma'^*][a \cup a^\circ][\Sigma'^*] \bigvee_{j/q_j \in Succ(q_i,a)} X_j]\!]^{val(V_0/X_0, \ldots, V_n/X_m)}_{\mathcal{A}'}$$

Nous avons prouvé que pour tout $q_i \in Q$ et $q'_i \in Q'$ tel que $q_i Z q'_i$, nous avons :

$$q'_i \in [\![\psi_i]\!]^{val(V_0/X_0, \ldots, V_n/X_m)}_{\mathcal{A}'}$$

ce qui est suffisant pour montrer que pour tout $0 \leq i \leq m$

$$V_i \subseteq [\![\Psi_i]\!]_{\mathcal{A}'}$$

Ce qui signifie que $< V_0, \ldots, V_m >$ est un sous ensemble du plus grand point fixe. Sachant que $q_0 Z q'_0$, nous obtenons que $q'_0 \in V_0$. Par conséquent $\mathcal{A}' \models \varphi(\mathcal{A}, \Sigma')$.

□

Annexe 2

Le but de cette annexe est de prouver le théorème 6.5.6. Pour cela, nous allons d'abord prouver les lemmes suivants.

Lemme 8.3.7. *Soient I_0 un ensemble de littéraux, $\mathcal{A}c$ un ACC sur $\Sigma \cup (\{!,?\} \times Port)$ et At, et soit $\mathcal{A}c^{med}$ un ACC sur $\{!,?\} \times Port$ tel que $\delta^{med} \subseteq \{\emptyset\} \times \{!,?\} \times Port \times \{\emptyset\}$.*

$$Exec(\mathcal{A}c \circledast \mathcal{A}c^{med}, I_0) \simeq_{iso} Del^{\circ}(Exec^{\circ}(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^{\circ}}}, I_0) \times Ren^{\circ}(\mathcal{A}c^{med}))$$

Lemme 8.3.8. *Soient l'ensemble I_0, un ACC $\mathcal{A}c$ sur $\Sigma \cup (\{!,?\} \times Port)$ et At et $\mathcal{C}$ un automate fini sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ qui boucle sur $\Sigma \cup (\{!,?\} \times Port)$.*

$$Del^{\circ}(Exec^{\circ}(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^{\circ}}}, I_0)) \times \mathcal{C}) \simeq_{iso} Exec(\mathcal{A}c \circledast Del^{\circ}(\mathcal{A}c^{L_{Port^{\circ}}} * Auto^{-1}(\mathcal{C})), I_0)$$

Pour prouver ces lemmes nous avons besoin de la définition suivante :

Définition 8.3.9 (**Fonction** Ren°**).** Soit $\mathcal{A}c^{med} = (Q^{med}, q_0^{med}, \delta^{med})$ un automate communicant conditionnel sur $\{!,?\} \times Port$ et At. $Ren^{\circ}(\mathcal{A}c^{med}) = (Q', q_0', \rightarrow')$ est un automate sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ tel que :

- $Q' = Q^{med}$,
- $q_0' = q_0^{med}$,
- $\rightarrow' \subseteq Q' \times (\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))) \times Q'$ est la relation de transitons définie par $q \rightarrow^{a}_{Ren^{\circ}(\mathcal{A}c^{med})} q'$ ssi une des conditions suivantes est satisfaite :
  - $a \in \Sigma$ et $q = q'$,
  - $a = \theta\pi$ et $q = q'$, avec $\theta \in \{!,?\}$
  - $a = \theta\pi^{\circ}$ et il existe $J \in 2^{Li(At)}$ tel que $(J, q, \theta\pi, q', \emptyset) \in \delta^{med}$, avec $\theta \in \{!,?\}$.

Soulignons que $Ren^{\circ}(\mathcal{A}c^{med})$ boucle sur $\Sigma \cup (\{!,?\} \times Port)$.

Pour rendre les preuves plus simple, dans le reste de ce chapitre nous supposons pour tout les automates communicants conditionnels et pour tout les automates que tout les états sont finaux. Ainsi, il n'est pas nécessaire de considérer l'ensemble des états finaux. Cependant, les preuves restent applicables dans le cas générals.

Démonstration. (Lemme 8.3.7) Soient $\mathcal{A}c = (Q, q_0, \delta)$ un ACC sur $\Sigma \cup (\{!,?\} \times Port)$ et At et $\mathcal{A}c^{med} = (Q^{med}, q_0^{med}, \delta^{med})$ un ACC sur $\{!,?\} \times Port$ et $At = \emptyset$. Rappelons que l'ensemble de toute les fonctions de la forme $\gamma : Port \to \{1,0\}$ est noté Γ et que l'ensemble des littéraux sur At est noté $Li(At)$.

Pour rendre la preuve plus claire, nous utilisons l'automate fini $\mathcal{A} = (Q^{\mathcal{A}}, q_0^{\mathcal{A}}, \to_{\mathcal{A}})$ sur $\Sigma \cup (\{!,?\} \times Port)$ pour représenter l'automate fini $Exec(\mathcal{A}c \circledast \mathcal{A}c^{med}, I_0)$. De la même façon, nous utilisons l'automate fini $\mathcal{B} = (Q^{\mathcal{B}}, q_0^{\mathcal{B}}, \to_{\mathcal{B}})$ sur $\Sigma \cup (\{!,?\} \times Port)$ pour représenter l'automate fini $Del^{\circ}(Exec^{\circ}(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^{\circ}}}, I_0) \times Ren^{\circ}(\mathcal{A}c^{med}))$. Ainsi, $Q^{\mathcal{A}} = (Q \times Q^{med}) \times \Gamma \times 2^{Li(At)}$ et $Q^{\mathcal{B}} = ((Q \times Q^{L_{Port^{\circ}}}) \times \Gamma \times 2^{Li(At)}) \times Q^{med}$. Soit $g : Q^{\mathcal{A}} \to Q^{\mathcal{B}}$ la bijection telle que $g(((q, q^{med}), \gamma, I)) = (((q, q_0^{L_{Port^{\circ}}}), \gamma, I), q^{med})$. Il est clair que $g(q_0^{\mathcal{A}}) = q_0^{\mathcal{B}}$. Nous prouvons que pour toute action $a \in \Sigma \cup (\{!,?\} \times Port)$ et pour tout état $((q_1, q_1^{med}), \gamma_1, I_1), ((q_2, q_2^{med}), \gamma_2, I_2) \in Q^{\mathcal{A}}$ que :

$$\begin{array}{c} ((q_1, q_1^{med}), \gamma_1, I_1) \to_{\mathcal{A}}^{\{a\}} ((q_2, q_2^{med}), \gamma_2, I_2) \\ \Longleftrightarrow \\ (((q_1, q_0^{L_{Port^{\circ}}}), \gamma_1, I_1), q_1^{med}) \to_{\mathcal{B}}^{\{a\}} (((q_2, q_0^{L_{Port^{\circ}}}), \gamma_2, I_2), q_2^{med}) \end{array}$$

($\Rightarrow$) Considérons l'implication de gauche à droite. Supposons que :

$$((q_1, q_1^{med}), \gamma_1, I_1) \to_{\mathcal{A}}^{\{a\}} ((q_2, q_2^{med}), \gamma_2, I_2). \tag{8.1}$$

Deux cas sont possibles pour a : $a \in \Sigma$ ou $a \in \{!,?\} \times Port$. Considérons le cas où $a \in \Sigma$. L'action a ne peut être exécutée que par $\mathcal{A}c$. Dans ce cas, la transition 8.1 implique :

- $\gamma_2 = \gamma_1$,
- $q_2^{med} = q_1^{med}$ et
- l'existence de $J_1, J_2 \in 2^{Li(At)}$ tels que $J_1 \subseteq I_1$, $I_2 = (I_1 \setminus \neg J_2) \cup J_2$ et $(J_1, q_1, a, q_2, J_2) \in \delta$.

A partir de la définition de $Exec^{\circ}$ et du produit asynchrone, $J_1 \subseteq I_1$, $I_2 = (I_1 \setminus \neg J_2) \cup J_2$ et $(J_1, q_1, a, q_2, J_2) \in \delta$ implique :

$$((q_1, q_0^{L_{Port^{\circ}}}), \gamma_1, I_1) \to_{Exec^{\circ}(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^{\circ}}}, I_0)} ((q_2, q_0^{L_{Port^{\circ}}}), \gamma_1, I_2).$$

De plus, $q_1^{med} \rightarrow^{\{a\}}_{Ren^\circ(\mathcal{A}c^{med})} q_1^{med}$. D'après la définition du produit synchrone on obtient :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{med}) \rightarrow^{a}_{Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0) \times Ren^\circ(\mathcal{A}c^{med})} (((q_2, q_0^{L_{Port^\circ}}), \gamma_1, I_2), q_1^{med})$$

Par conséquent, d'après la définition de Del° :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{med}) \rightarrow^{\{a\}}_{\mathcal{B}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_1, I_2), q_1^{med})$$

Sachant que $\gamma_2 = \gamma_1$ et $q_2^{med} = q_1^{med}$ alors :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{med}) \rightarrow^{\{a\}}_{\mathcal{B}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2), q_2^{med})$$

Considérons le cas où $a = ?\pi$, $\pi \in Port$ (le raisonnement est le même pour $a = !\pi$), l'action $?\pi$ peut être exécutée par $\mathcal{A}c$ ou par $\mathcal{A}c^{med}$. Dans le cas ou l'action est effectuée par $\mathcal{A}c$, on utilise le même raisonnement que celui utilisé dans le cas où $a \in \Sigma$. Supposons que $\mathcal{A}c^{med}$ exécute l'action $?\pi$. Donc il existe $q_2^{med} \in Q^{med}$ tel que $(\emptyset, q_1^{med}, ?\pi, q_2^{med}, \emptyset) \in \delta^{med}$ et $I_2 = I_1$. De plus :

- $q_2 = q_1$,
- $\gamma_1(\pi) = 1$,
- $\gamma_2(\pi) = 0$ et pour tout $\pi' \neq \pi$, $\gamma_2(\pi') = \gamma_1(\pi')$.

Sachant que $(\emptyset, q_1^{med}, ?\pi, q_2^{med}, \emptyset) \in \delta^{med}$ alors :

$$q_1^{med} \rightarrow^{\{?\pi^\circ\}}_{Ren^\circ(\mathcal{A}c^{med})} q_2^{med}. \tag{8.2}$$

Nous savons que $(\emptyset, q_0^{L_{Port^\circ}}, ?\pi^\circ, q_0^{L_{Port^\circ}}, \emptyset) \in \delta^{L_{Port^\circ}}$ et que $\gamma_1(\pi) = 1$. Ainsi, à partir de la définition de $Exec^\circ$ et du produit asynchrone, on obtient :

$$((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1) \rightarrow^{\{?\pi^\circ\}}_{Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0)} (((q_1, q_0^{L_{Port^\circ}}), \gamma_2, I_1)). \tag{8.3}$$

A partir de la définition du produit synchrone, de la transition 8.2 et de la transition 8.3 on obtient :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{med}) \rightarrow^{?\pi^\circ}_{Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0) \times Ren^\circ(\mathcal{A}c^{med})} (((q_1, q_0^{L_{Port^\circ}}), \gamma_2, I_1), q_1^{med})$$

Par conséquent, d'après la définition de Del° :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{med}) \rightarrow^{\{?\pi\}}_{\mathcal{B}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_1, I_2), q_1^{med})$$

Sachant que $q_2 = q_1$ et $I_2 = I_1$ alors :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{med}) \rightarrow^{\{?\pi\}}_{\mathcal{B}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2), q_2^{med})$$

($\Leftarrow$) Considérons l'implication de droite à gauche. Supposons que :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{med}) \rightarrow_{\mathcal{B}}^{\{a\}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2), q_2^{med}). \quad (8.4)$$

Deux cas sont possibles pour a : $a \in \Sigma$ ou $a \in \{!, ?\} \times Port$. Considérons le cas où $a \in \Sigma$. Sachant que $\mathcal{A}c^{L_{Port^\circ}}$ ne peux exécuter l'action a alors la transition 8.4 implique :

- $\gamma_2 = \gamma_1$,
- $q_2^{med} = q_1^{med}$ et
- l'existence de $J_1, J_2 \in 2^{Li(At)}$ tels que $J_1 \subseteq I_1$, $I_2 = (I_1 \setminus \neg J_2) \cup J_2$ et $(J_1, q_1, a, q_2, J_2) \in \delta$.

A partir de la définition de $Exec$ et du produit asynchrone, l'existence de $J_1, J_2 \in 2^{Li(At)}$ tels que $J_1 \subseteq I_1$, $I_2 = (I_1 \setminus \neg J_2) \cup J_2$ et $(J_1, q_1, a, q_2, J_2) \in \delta$ implique :

$$((q_1, q_1^{med}), \gamma_1, I_1) \rightarrow_{\mathcal{A}}^{\{a\}} ((q_2, q_1^{med}), \gamma_1, I_2).$$

Sachant que $q_2^{med} = q_1^{med}$ et $\gamma_2 = \gamma_1$ alors :

$$((q_1, q_1^{med}), \gamma_1, I_1) \rightarrow_{\mathcal{A}}^{\{a\}} ((q_2, q_2^{med}), \gamma_2, I_2).$$

Considérons le cas $a = ?\pi$, $\pi \in Port$ (le raisonnement est le même dans le cas où $a = !\pi$). Dans ce cas, la transition 8.4 implique deux possibilités. La première est :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{med}) \rightarrow_{\mathcal{D}}^{\{?\pi\}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2), q_2^{med}). \quad (8.5)$$

La seconde est :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{med}) \rightarrow_{\mathcal{D}}^{\{?\pi^\circ\}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2), q_2^{med}). \quad (8.6)$$

Pour ne pas encombrer les transitions 8.5 et 8.6, nous avons noté par $\mathcal{D}$ l'automate $Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0) \times Ren^\circ(\mathcal{A}c^{med})$.

Concernant le premier cas, la transition 8.5 implique que l'action $?\pi$ est exécutée par $\mathcal{A}c$. Dans ce cas, le raisonnement est le même que celui utilisé dans le cas où $a \in \Sigma$. Concernant le second cas, la transition 8.6 implique d'une part qu'il existe un état $q_2^{med} \in Q^{med}$ tel que $q_1^{med} \rightarrow_{Ren^\circ(\mathcal{A}c^{med})}^{\{?\pi^\circ\}} q_2^{med}$ et d'autre part que :

$$((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1) \rightarrow_{Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0)}^{\{?\pi^\circ\}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2)). \quad (8.7)$$

Sachant que $\mathcal{A}c$ ne peux pas exécuter l'action $?\pi^\circ$ alors la transition 8.7 implique :

– $q_2 = q_1$,
– $\gamma_1(\pi) = 1$, $\gamma_2(\pi) = 0$ et pour tout $\pi' \neq \pi$ on a $\gamma_2(\pi') = \gamma_1(\pi')$ et
– $I_2 = I_1$.

Puisque $q_1^{med} \rightarrow^{\{?\pi^\circ\}}_{Ren^\circ(\mathcal{A}c^{med})} q_2^{med}$ alors $(\emptyset, q_1^{med}, ?\pi, q_2^{med}, \emptyset) \in \delta^{med}$. Par conséquent, sachant que $\gamma_1(\pi) = 1$ alors à partir des définitions du produit synchrone et de $Exec$ on obtient :

$$((q_1, q_1^{med}), \gamma_1, I_1) \rightarrow^{\{?\pi\}}_{\mathcal{A}} ((q_1, q_2^{med}), \gamma_2, I_1).$$

Sachant que $q_2 = q_1$ et $I_2 = I_1$ alors :

$$((q_1, q_1^{med}), \gamma_1, I_1) \rightarrow^{\{?\pi\}}_{\mathcal{A}} ((q_2, q_2^{med}), \gamma_2, I_2).$$

□

Démonstration. (Lemme 8.3.8) Soient $\mathcal{A}c = (Q, q_0, \delta)$ un ACC sur $\Sigma \cup (\{!, ?\} \times Port)$ et At et $\mathcal{C} = (Q^{\mathcal{C}}, q_0^{\mathcal{C}}, \delta^{\mathcal{C}})$ un automate fini sur $\Sigma \cup (\{!, ?\} \times (Port \cup Port^\circ))$ qui boucle sur $\Sigma \cup (\{!, ?\} \times Port)$. Pour rendre la preuve plus claire, nous utilisons l'automate fini $\mathcal{A} = (Q^{\mathcal{A}}, q_0^{\mathcal{A}}, \rightarrow_{\mathcal{A}})$ sur $\Sigma \cup (\{!, ?\} \times Port)$ pour représenter l'automate fini $Del^\circ(Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0)) \times \mathcal{C})$. De la même façon, nous utilisons l'automate fini $\mathcal{B} = (Q^{\mathcal{B}}, q_0^{\mathcal{B}}, \rightarrow_{\mathcal{B}})$ sur $\Sigma \cup (\{!, ?\} \times (Port \cup Port^\circ))$ pour représenter l'automate fini $Exec(\mathcal{A}c \circledast Del^\circ(\mathcal{A}c^{L_{Port^\circ}} * Auto^{-1}(\mathcal{C})), I_0)$. Ainsi, $Q^{\mathcal{A}} = ((Q \times Q^{L_{Port^\circ}}) \times 2^{Li(At)}) \times Q^{\mathcal{C}}$ et $Q^{\mathcal{B}} = (Q \times (Q^{L_{Port^\circ}} \times Q^{\mathcal{C}})) \times 2^{Li(At)}$. Soit $g : Q^{\mathcal{A}} \rightarrow Q^{\mathcal{B}}$ la bijection telle que $g((((q, q_0^{L_{Port^\circ}}), I), q^{\mathcal{C}})) = ((q, (q_0^{L_{Port^\circ}}, q^{\mathcal{C}})), I)$. Il est clair que $g(q_0^{\mathcal{A}}) = q_0^{\mathcal{B}}$. Il reste à prouver que pour toute action $a \in \Sigma \cup (\{!, ?\} \times Port)$ et pour tout état $(((q_1, q_0^{L_{Port^\circ}}), I_1), q_1^{\mathcal{C}})$ et $(((q_2, q_0^{L_{Port^\circ}}), I_2), q_2^{\mathcal{C}})$:

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{\mathcal{C}}) \rightarrow^{\{a\}}_{\mathcal{A}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2), q_2^{\mathcal{C}})$$
$$\text{ssi}$$
$$((q_1, (q_0^{L_{Port^\circ}}, q_1^{\mathcal{C}})), \gamma_1, I_1) \rightarrow^{\{a\}}_{\mathcal{B}} ((q_2, (q_0^{L_{Port^\circ}}, q_2^{\mathcal{C}})), \gamma_2, I_2)$$

($\Rightarrow$) Considérons l'implication de gauche à droite. Supposons que :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{\mathcal{C}}) \rightarrow^{\{a\}}_{\mathcal{A}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2), q_2^{\mathcal{C}}) \qquad (8.8)$$

Deux cas sont possibles pour a : $a \in \Sigma$ ou $a \in \{!, ?\} \times Port$. Considérons le cas où $a \in \Sigma$. A partir de la définition du produit synchrone l'equation 8.8 implique que $((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1) \rightarrow^{\{a\}}_{Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0)} ((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2)$. Sachant que $\mathcal{A}c^{L_{Port^\circ}}$ ne peut pas exécuter l'action a alors :

– il existe des ensembles $J_1, J_2 \in 2^{Li(At)}$ tels que $J_1 \subseteq I_1$, $I_2 = (I_1 \setminus \neg J_2) \cup J_2$ et $(J_1, q_1, a, q_2, J_2) \in \delta$ et
– $\gamma_2 = \gamma_1$.

De plus, $\mathcal{C}$ boucle sur a c'est à dire pour tout $q_3^{\mathcal{C}} \in Q^{\mathcal{C}}$ si $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{a\}} q_3^{\mathcal{C}}$ alors $q_3^{\mathcal{C}} = q_1^{\mathcal{C}}$. Par conséquent, $q_2^{\mathcal{C}} = q_1^{\mathcal{C}}$. A partir de la définition de $Exec$ et du produit asynchrone on obtient :

$$((q_1, (q_0^{L_{Port^\circ}}, q_1^{\mathcal{C}})), \gamma_1, I_1) \rightarrow_{\mathcal{B}}^{\{a\}} ((q_2, (q_0^{L_{Port^\circ}}, q_1^{\mathcal{C}})), \gamma_1, I_2)$$

Sachant que $\gamma_2 = \gamma_1$ et $q_2^{\mathcal{C}} = q_1^{\mathcal{C}}$ alors :

$$((q_1, (q_0^{L_{Port^\circ}}, q_1^{\mathcal{C}})), \gamma_1, I_1) \rightarrow_{\mathcal{B}}^{\{a\}} ((q_2, (q_0^{L_{Port^\circ}}, q_2^{\mathcal{C}})), \gamma_2, I_2) \tag{8.9}$$

Considérons le cas où $a = ?\pi$, $\pi \in Port$ (le raisonnement est le même pour $a = !\pi$). A partir de la définition de Del°, la transition 8.9 implique deux possibilités. La première possibilité est que :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{\mathcal{C}}) \rightarrow_{\mathcal{D}}^{\{?\pi\}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2), q_2^{\mathcal{C}}). \tag{8.10}$$

La seconde possibilité est que :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{\mathcal{C}}) \rightarrow_{\mathcal{D}}^{\{?\pi^\circ\}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2), q_2^{\mathcal{C}}). \tag{8.11}$$

Pour ne pas encombrer les transitions 8.10 et 8.11, nous avons noté par $\mathcal{D}$, l'automate $Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0) \times \mathcal{C}$.

Supposons que l'action $?\pi$ est exécutée. Dans ce cas le raisonnement est identique à celui utilisé dans le cas où $a \in \Sigma$. Supposons que l'action $?\pi^\circ$ est exécutée. Dans ce cas, par définition du produit synchrone on déduit que $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{?\pi^\circ\}} q_2^{\mathcal{C}}$ et que :

$$((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1) \rightarrow_{Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}})}^{\{?\pi^\circ\}} ((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2). \tag{8.12}$$

Sachant que $(\emptyset, q_0^{L_{Port^\circ}}, ?\pi^\circ, q^{L_{Port_0^\circ}}, \emptyset) \in \delta^{L_{Port^\circ}}$ alors la transition 8.12 implique :

- $\gamma_1(\pi) = 1$, $\gamma_2(\pi) = 0$ et pour tout $\pi' \neq \pi$, $\gamma_2(\pi') = \gamma_1(\pi')$,
- $q_2 = q_1$ et
- $I_2 = I_1$.

Puisque $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{?\pi^\circ\}} q_2^{\mathcal{C}}$ alors à partir de la définition de $Auto^{-1}(\mathcal{C})$ en déduit que $(\emptyset, q_1^{\mathcal{C}}, ?\pi^\circ, q_2^{\mathcal{C}}, \emptyset) \in \delta^{Auto^{-1}(\mathcal{C})}$. De plus, nous savons que $(\emptyset, q_0^{L_{Port^\circ}}, ?\pi^\circ, q_0^{L_{Port^\circ}}, \emptyset) \in \delta^{L_{Port^\circ}}$. A partir de la définition du produit synchrone entre ACC, on en déduit que $(\emptyset, (q_0^{L_{Port^\circ}}, q_1^{\mathcal{C}}), ?\pi^\circ, (q^{L_{Port^\circ}}, q_2^{\mathcal{C}}), \emptyset) \in \delta^{\mathcal{A}c^{L_{Port^\circ}} * Auto^{-1}(\mathcal{C})}$. Ainsi, $(\emptyset, (q_0^{L_{Port^\circ}}, q_1^{\mathcal{C}}), ?\pi, (q^{L_{Port^\circ}}, q_2^{\mathcal{C}}), \emptyset) \in \delta^{Del^\circ(\mathcal{A}c^{L_{Port^\circ}} * Auto^{-1}(\mathcal{C}))}$. Nous savons que $\gamma_1(\pi) = 1$. Par conséquent, on obtient :

$$((q_1, (q_0^{L_{Port^\circ}}, q_1^{\mathcal{C}})), \gamma_1, I_1) \rightarrow_{\mathcal{B}}^{\{?\pi\}} ((q_1, (q_0^{L_{Port^\circ}}, q_2^{\mathcal{C}})), \gamma_2, I_1)$$

Sachant que $q_2 = q_1$ et que $I_2 = I_1$ alors :

$$((q_1, (q_0^{L_{Port^\circ}}, q_1^{\mathcal{C}})), \gamma_1, I_1) \rightarrow_{\mathcal{B}}^{\{?\pi\}} ((q_2, (q_0^{L_{Port^\circ}}, q_2^{\mathcal{C}})), \gamma_2, I_2)$$

($\Leftarrow$) Considérons l'implication de droite à gauche. Supposons que :

$$((q_1, (q_0^{L_{Port^\circ}}, q_1^{\mathcal{C}})), \gamma_1, I_1) \rightarrow_{\mathcal{B}}^{\{a\}} ((q_2, (q_0^{L_{Port^\circ}}, q_2^{\mathcal{C}})), \gamma_2, I_2) \tag{8.13}$$

Il existe deux cas possibles pour a : $a \in \Sigma$ ou $a \in \{!, ?\} \times Port$. Considérons le cas $a \in \Sigma$. Sachant que $\mathcal{A}c^{L_{Port^\circ}}$ ne peut pas exécuter l'action a alors cette dernière est exécutée par $\mathcal{A}c$. Plus précisément, la transition 8.13 implique :

- l'existence des ensembles $J_1, J_2 \in 2^{Li(At)}$ tels que $J_1 \subseteq I_1$, $I_2 = (I_1 \setminus \neg J_2) \cup J_2$ et $(J_1, q_1, a, q_2, J_2) \in \delta$,
- $q_2^{\mathcal{C}} = q_1^{\mathcal{C}}$ et
- $\gamma_2 = \gamma_1$.

A partir de la définition de $Exec^\circ$ et du produit asynchrone on obtient :

$$((q_1, q^{L_{Port^\circ}}), \gamma_1, I_1) \rightarrow_{Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}})}^{\{a\}} ((q_2, q^{L_{Port^\circ}}), \gamma_1, I_2)$$

De plus, l'automate $\mathcal{C}$ boucle sur a. Formellement, $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{a\}} q_1^{\mathcal{C}}$. Par conséquent, à partir de la définition du produit synchrone et de la fonction Del° on a :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{\mathcal{C}}) \rightarrow_{\mathcal{A}}^{\{a\}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_1, I_2), q_1^{\mathcal{C}})$$

Sachant que $q_2^{\mathcal{C}} = q_1^{\mathcal{C}}$ et $\gamma_2 = \gamma_1$ alors :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{\mathcal{C}}) \rightarrow_{\mathcal{A}}^{\{a\}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2), q_2^{\mathcal{C}})$$

Considérons le cas où $a = ?\pi$, $\pi \in Port$ (le raisonnement est le même dans le cas où $a = !\pi$). Dans ce cas, l'action $?\pi$ est exécutée par $\mathcal{A}c$ ou par $Del^\circ(\mathcal{A}c^{L_{Port^\circ}} * Auto^{-1}(\mathcal{C}))$. Le premier cas est traité de la même façon que le cas où $a \in \Sigma$. Considérons le second cas. Dans ce cas, $q_2 = q_1$. Sachant que $\mathcal{A}c^{L_{Port^\circ}}$ ne peut exécuter que des actions dans $\{!, ?\} \times Port^\circ$ alors le produit synchrone $\mathcal{A}c^{L_{Port^\circ}} * Auto^{-1}(\mathcal{C})$ exécute l'action $?\pi^\circ$ et par conséquent :

$$q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{?\pi^\circ\}} q_2^{\mathcal{C}} \tag{8.14}$$

De plus, à partir de la définition de $Exec$ on a :

- $\gamma_1(\pi) = 1$, $\gamma_2(\pi) = 0$ et pour tout $\pi' \neq \pi$, $\gamma_2(\pi') = \gamma_1(\pi')$ et
- $I_2 = I_1$.

Puisque $(\emptyset, q_0^{L_{Port^\circ}}, ?\pi^\circ, q^{L_{Port_0^\circ}}, \emptyset) \in \delta^{L_{Port^\circ}}$ et que $\gamma_1(\pi) = 1$ alors :

$$(q_1, q_0^{L^\circ_{Port}}) \rightarrow^{\{?\pi^\circ\}}_{Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0)} (q_1, q_0^{L^\circ_{Port}}) \tag{8.15}$$

A partir des transitions 8.14 et 8.15 et de la définition du produit synchrone on a :

$$((q_1, q_0^{L^\circ_{Port}}), I_1) \rightarrow^{\{?\pi^\circ\}}_{Exec^\circ(\mathcal{A}c \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0) \times \mathcal{C}} ((q_1, q_0^{L^\circ_{Port}}), I_1)$$

Ainsi, à partir de la définition de Del° on a :

$$((q_1, q_0^{L^\circ_{Port}}), I_1) \rightarrow^{\{?\pi^\circ\}}_{\mathcal{A}} ((q_1, q_0^{L^\circ_{Port}}), I_1)$$

Sachant que $q_2 = q_1$ et $I_2 = I_1$ on a :

$$(((q_1, q_0^{L_{Port^\circ}}), \gamma_1, I_1), q_1^{\mathcal{C}}) \rightarrow^{\{a\}}_{\mathcal{A}} (((q_2, q_0^{L_{Port^\circ}}), \gamma_2, I_2), q_2^{\mathcal{C}})$$

□

Remarque 8.3.10. *Dans ce qui suit, toutes les relations de bisimulation considérées sont des bisimulation modulo $\{!, ?\} \times Port'$. Plus précisement, dans les équivalences 8.16, 8.17, 8.18, 8.19, 8.20 et 8.21.*

En se basant sur les lemmes 8.3.7 et 8.3.8 nous prouvons le théorème 6.5.6.

Démonstration. Considérons l'équivalence entre les points (1) et (2). Concernant l'implication de gauche à droite. Soit $\mathcal{A}c^{med}$ sur $\{!, ?\} \times Port$ un automate fini tel que :

$$Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Exec(\mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0). \tag{8.16}$$

D'après le lemme 8.3.7, l'équivalence 8.16 implique :

$$Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Del^\circ(Exec^\circ(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0)) \times Ren^\circ(\mathcal{A}c^{med}). \tag{8.17}$$

Il suffit de considérer l'automate fini $\mathcal{C} = Ren^\circ(\mathcal{A}c^{med})$ sur $\Sigma \cup (\{!, ?\} \times (Port \cup Port^\circ))$ pour avoir un automate qui boucle sur $\Sigma \cup (\{!, ?\} \times Port)$ tel que :

$$Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Del^\circ(Exec^\circ(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port^\circ}}, I_0) \times \mathcal{C}). \tag{8.18}$$

Concernant l'implication de droite à gauche. Soit $\mathcal{C}$ un automate fini sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^{\circ}))$ qui boucle sur $\Sigma \cup (\{!,?\} \times Port)$ tel que :

$$Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Del^{\circ}(Exec^{\circ}(\mathcal{A}c' \circledast \mathcal{A}c^{L_{Port^{\circ}}}, I_0) \times \mathcal{C}). \tag{8.19}$$

D'après le lemme 8.3.8, l'équivalence 8.19 implique :

$$Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Exec(\mathcal{A}c' \circledast Del^{\circ}(\mathcal{A}c^{L_{Port^{\circ}}} * Auto^{-1}(\mathcal{C})), I_0). \tag{8.20}$$

Il suffit de considérer l'automate communicant conditionnel $\mathcal{A}c^{med} = Del^{\circ}(\mathcal{A}c^{L_{Port^{\circ}}} * Auto^{-1}(\mathcal{C})$ sur $\{!,?\} \times Port$ pour avoir :

$$Exec(\mathcal{A}c, I_0) \longleftrightarrow_{bi} Exec(\mathcal{A}c' \circledast \mathcal{A}c^{med}, I_0). \tag{8.21}$$

□

Annexe 3

Le but de cette annexe est de prouver que la relation Z^f de la définition 6.5.13 est une bisimulation entre $\mathcal{A}$ et $Del^\circ(\mathcal{A}' \times \mathcal{C}^f)$, où $\mathcal{C}^f$ est l'automate fini de la définition 6.5.12. Rappelons la définition de la Relation binaire $\equiv$, de $\mathcal{C}^f$ et de Z^f. Soit $\mathcal{A} = (Q^{\mathcal{A}}, q_0^{\mathcal{A}}, \rightarrow_{\mathcal{A}})$ un automate fini sur $\Sigma \cup (\{!,?\} \times Port)$, $\mathcal{A}' = (Q^{\mathcal{A}'}, q_0^{\mathcal{A}'}, \rightarrow_{\mathcal{A}'})$ un automate fini sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^\circ))$, et $\mathcal{C} = (Q^{\mathcal{C}}, q_0^{\mathcal{C}}, \rightarrow_{\mathcal{C}})$ un automate fini sur $\Sigma \cup (\{!,?\} \times (Port \cup Port^\circ))$ qui boucle sur $\Sigma \cup (\{!,?\} \times Port)$ et Z une relation de bisimulation entre $\mathcal{A}$ et $Del^\circ(\mathcal{A}' \times \mathcal{C})$.

Définition 8.3.11 (**Relation binaire** $\equiv$)**.** *La relation* $\equiv \subseteq Q^{\mathcal{C}} \times Q^{\mathcal{C}}$ *est la relation binaire telle que :*

- $q_1^{\mathcal{C}} \equiv q_2^{\mathcal{C}}$ *ssi pour tout* $q^{\mathcal{A}} \in Q^{\mathcal{A}}$ *et pour tout* $q^{\mathcal{A}'} \in Q^{\mathcal{A}'}$, $q^{\mathcal{A}} Z(q^{\mathcal{A}'}, q_1^{\mathcal{C}})$ *ssi* $q^{\mathcal{A}} Z(q^{\mathcal{A}'}, q_2^{\mathcal{C}})$

Définition 8.3.12 (**Automate obtenu par filtration** $\mathcal{C}^f$)**.** *La filtration de* $\mathcal{C}$ *relativement à* $\mathcal{A}$ *et* $\mathcal{A}'$ *est l'automate* $\mathcal{C}^f = (Q^{\mathcal{C}^f}, q_0^{\mathcal{C}^f}, \rightarrow_{\mathcal{C}^f})$ *sur* $\Sigma \cup (\{!,?\} \times (Port \cup Port^\circ))$ *qui boucle sur* $\Sigma \cup (\{!,?\} \times Port)$ *et tel que :*

- $Q^{\mathcal{C}^f} = Q^{\mathcal{C}} / \equiv$,
- $q_0^{\mathcal{C}^f} = \mid q_0^{\mathcal{C}} \mid$ *et*
- $\rightarrow_{\mathcal{C}^f}$ *est la relation telle que :*
  - $\mid q_1^{\mathcal{C}} \mid \rightarrow_{\mathcal{C}^f}^{\{\theta\pi^\circ\}} \mid q_2^{\mathcal{C}} \mid$, $\theta \in \{!,?\}$ *et* $\pi^\circ \in Port^\circ$, *ssi pour tout* $q_1^{\mathcal{A}} \in Q^{\mathcal{A}}$ *et pour tout* $q_1^{\mathcal{A}'}$, $q_2^{\mathcal{A}'} \in Q^{\mathcal{A}'}$ *si* $q_1^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{\theta\pi^\circ\}} q_2^{\mathcal{A}'}$ *et* $q_1^{\mathcal{A}} Z(q_1^{\mathcal{A}'}, q_1^{\mathcal{C}})$ *alors il existe* $q_2^{\mathcal{A}} \in Q^{\mathcal{A}}$ *tel que* $q_1^{\mathcal{A}} \rightarrow_{\mathcal{A}}^{\{\theta\pi\}} q_2^{\mathcal{A}}$ *et* $q_2^{\mathcal{A}} Z(q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$.

Définition 8.3.13 (**La relation** Z^f **obtenue par filtration**)**.** *Soit* $Z^f \subseteq Q^{\mathcal{A}} \times (Q^{\mathcal{A}'} \times Q^{\mathcal{C}^f})$ *la relation binaire telle que pour tout* $q^{\mathcal{A}} \in Q^{\mathcal{A}}$ *et pour tout* $(q^{\mathcal{A}'}, \mid q^{\mathcal{C}} \mid) \in Q^{\mathcal{A}'} \times Q^{\mathcal{C}^f}$, $q^{\mathcal{A}} Z^f (q^{\mathcal{A}'}, \mid q^{\mathcal{C}} \mid)$ *ssi* $q^{\mathcal{A}} Z(q^{\mathcal{A}'}, q^{\mathcal{C}})$.

Proposition 8.3.14. *La relation* Z^f *obtenu par filtration est une relation de bisimulation entre* $\mathcal{A}$ *et* $Del^\circ(\mathcal{A}' \times \mathcal{C}^f)$.

Démonstration. Dans ce qui suit, nous montrons que Z^f et $Z^{f^{-1}}$ sont des relation de simulation. Concernant les états initiaux. Sachant que Z est une relation de bisimulation alors $q_0^{\mathcal{A}} Z (q_0^{\mathcal{A}'}, q_0^{\mathcal{C}})$. A partir de la définition de Z^f on a $q_0^{\mathcal{A}} Z^f (q_0^{\mathcal{A}'}, \mid q_0^{\mathcal{C}} \mid)$. Par conséquent, $q_0^{\mathcal{A}} Z^f (q_0^{\mathcal{A}'}, q_0^{\mathcal{C}^f})$.

Vérification que Z^f est une relation de simulation

Considérons les états $q_1^{\mathcal{A}} \in Q^{\mathcal{A}}$, $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \in Q^{\mathcal{A}'} \times Q^{\mathcal{C}^f}$ tels que $q_1^{\mathcal{A}} Z^f (q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid)$. Supposons qu'il existe un état $q_2^{\mathcal{A}} \in Q^{\mathcal{A}}$ et une action $a \in \Sigma \cup (\{!, ?\} \times Port)$ tel que $q_1^{\mathcal{A}} \rightarrow_{\mathcal{A}}^{\{a\}} q_2^{\mathcal{A}}$.

Sachant que $q_1^{\mathcal{A}} Z^f (q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid)$ alors par définition de Z^f on a : $q_1^{\mathcal{A}} Z (q_1^{\mathcal{A}'}, q_1^{\mathcal{C}})$. De plus Z est une relation de bisimulation entre $\mathcal{A}$ et $Del^\circ(\mathcal{A}' \times \mathcal{C})$, alors il existe $(q_2^{\mathcal{A}'}, q_2^{\mathcal{C}}) \in Q^{\mathcal{A}'} \times Q^{\mathcal{C}}$ tel que : $(q_1^{\mathcal{A}'}, q_1^{\mathcal{C}}) \rightarrow_{Del^\circ(\mathcal{A}' \times \mathcal{C})}^{\{a\}} (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$ et $q_2^{\mathcal{A}} Z (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$.

Il existe deux cas possibles pour a : $a \in \Sigma$ et $a \in \{!, ?\} \times Port$.

Cas où $a \in \Sigma$

On sait que $(q_1^{\mathcal{A}'}, q_1^{\mathcal{C}}) \rightarrow_{Del^\circ(\mathcal{A}' \times \mathcal{C})}^{\{a\}} (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$ alors par définition de Del° on a : $(q_1^{\mathcal{A}'}, q_1^{\mathcal{C}}) \rightarrow_{\mathcal{A}' \times \mathcal{C}}^{\{a\}} (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$. Cela implique, par définition du produit synchrone, que :

1. $q_1^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{a\}} q_2^{\mathcal{A}'}$ et
2. $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{a\}} q_2^{\mathcal{C}}$.

De plus, $\mathcal{C}$ boucle sur Σ alors $q_2^{\mathcal{C}} = q_1^{\mathcal{C}}$.

Sachant que $\mathcal{C}^f$ boucle sur Σ et que $q_1^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{a\}} q_2^{\mathcal{A}'}$, alors par définition du produit synchrone on a : $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow_{\mathcal{A}' \times \mathcal{C}^f}^{\{a\}} (q_2^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid)$. Par définition de Del° on obtient : $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow_{Del^\circ(\mathcal{A}' \times \mathcal{C}^f)}^{\{a\}} (q_2^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid)$. Sachant que $q_2^{\mathcal{A}} Z (q_2^{\mathcal{A}'}, q_1^{\mathcal{C}})$ alors par définition de Z^f on a : $q_2^{\mathcal{A}} Z^f (q_2^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid)$. Par conséquent, sachant que $\mid q_2^{\mathcal{C}} \mid = \mid q_1^{\mathcal{C}} \mid$, on a : $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow_{Del^\circ(\mathcal{A}' \times \mathcal{C}^f)}^{\{a\}} (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$ et $q_2^{\mathcal{A}} Z^f (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$.

Cas où $a \in \{!, ?\} \times Port$

Considérons le cas où $a = !\pi$ avec $\pi \in Port$, (le raisonnement est le même pour le cas où $a = ?\pi$). On sait que $(q_1^{\mathcal{A}'}, q_1^{\mathcal{C}}) \rightarrow_{Del^\circ(\mathcal{A}' \times \mathcal{C})}^{\{!\pi\}} (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$ alors par définition de Del° on a deux cas possibles.

(1) $(q_1^{\mathcal{A}'}, q_1^{\mathcal{C}}) \rightarrow_{\mathcal{A}' \times \mathcal{C}}^{\{!\pi\}} (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$ et

(2) $(q_1^{\mathcal{A}'}, q_1^{\mathcal{C}}) \rightarrow_{\mathcal{A}'\times\mathcal{C}}^{\{!\pi^\circ\}} (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$

Pour le cas (1), le raisonement est le même que celui présenté dans le cas où $a \in \Sigma$.

Considérons le cas (2). Sachant $(q_1^{\mathcal{A}'}, q_1^{\mathcal{C}}) \rightarrow_{\mathcal{A}'\times\mathcal{C}}^{\{!\pi^\circ\}} (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$ alors par définition du produit synchrone on a :

1. $q_1^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{!\pi^\circ\}} q_2^{\mathcal{A}'}$ et
2. $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{!\pi^\circ\}} q_2^{\mathcal{C}}$.

Montrons que : $\mid q_1^{\mathcal{C}} \mid \rightarrow_{\mathcal{C}^f}^{\{!\pi^\circ\}} \mid q_2^{\mathcal{C}} \mid$. Pour cela nous devons considérer des états $q_3^{\mathcal{A}} \in Q^{\mathcal{A}}$, $q_3^{\mathcal{A}'}$, $q_4^{\mathcal{A}'} \in Q^{\mathcal{A}'}$ tel que $q_3^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{!\pi^\circ\}} q_4^{\mathcal{A}'}$ et $q_3^{\mathcal{A}} Z (q_3^{\mathcal{A}'}, q_1^{\mathcal{C}})$. Ensuite, nous devons montrer qu'il existe un état $q_4^{\mathcal{A}} \in Q^{\mathcal{A}}$ tel que $q_3^{\mathcal{A}} \rightarrow_{\mathcal{A}}^{\{!\pi\}} q_4^{\mathcal{A}}$ et $q_4^{\mathcal{A}} Z (q_4^{\mathcal{A}'}, q_2^{\mathcal{C}})$.

Sachant que $q_3^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{!\pi^\circ\}} q_4^{\mathcal{A}'}$ et $q_1^{\mathcal{C}} \rightarrow_{\mathcal{C}}^{\{!\pi^\circ\}} q_2^{\mathcal{C}}$ alors $(q_3^{\mathcal{A}'}, q_1^{\mathcal{C}}) \rightarrow_{\mathcal{A}'\times\mathcal{C}}^{\{!\pi^\circ\}} (q_4^{\mathcal{A}'}, q_2^{\mathcal{C}})$. Par définition de Del° on a : $(q_3^{\mathcal{A}'}, q_1^{\mathcal{C}}) \rightarrow_{Del^\circ(\mathcal{A}'\times\mathcal{C})}^{\{!\pi\}} (q_4^{\mathcal{A}'}, q_2^{\mathcal{C}})$. De plus, la relation Z est une bisimulation entre $\mathcal{A}$ et $Del^\circ(\mathcal{A}' \times \mathcal{C})$. Par conséquent, il existe un état $q_4^{\mathcal{A}} \in Q^{\mathcal{A}}$ tel que $q_3^{\mathcal{A}} \rightarrow_{\mathcal{A}}^{\{!\pi\}} q_4^{\mathcal{A}}$ et $q_4^{\mathcal{A}} Z (q_4^{\mathcal{A}'}, q_2^{\mathcal{C}})$. Donc : $\mid q_1^{\mathcal{C}} \mid \rightarrow_{\mathcal{C}^f}^{\{!\pi^\circ\}} \mid q_2^{\mathcal{C}} \mid$.

Par définition du produit synchrone, sachant que $q_1^{\mathcal{A}'} \rightarrow_{\mathcal{A}'}^{\{!\pi^\circ\}} q_2^{\mathcal{A}'}$ et $\mid q_1^{\mathcal{C}} \mid \rightarrow_{\mathcal{C}^f}^{\{!\pi^\circ\}} \mid q_2^{\mathcal{C}} \mid$ alors $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow_{\mathcal{A}'\times\mathcal{C}^f}^{\{!\pi^\circ\}} (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$. Par conséquent, par définition de Del°, $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow_{Del^\circ(\mathcal{A}'\times\mathcal{C}^f)}^{\{!\pi\}} (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$. De plus, on sait que $q_2^{\mathcal{A}} Z (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$. Donc, par définition de Z^f on a : $q_2^{\mathcal{A}} Z^f (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$. Ainsi nous avons montré que : $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow_{Del^\circ(\mathcal{A}'\times\mathcal{C}^f)}^{\{a\}} (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$ et $q_2^{\mathcal{A}} Z^f (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$.

À partir des deux cas que nous avons considéré pour a, on obtient que Z^f est une relation de simulation entre $\mathcal{A}$ et $Del^\circ(\mathcal{A}' \times \mathcal{C}^f)$.

Vérification que $Z^{f^{-1}}$ est une relation de simulation

Considérons les états $q_1^{\mathcal{A}} \in Q^{\mathcal{A}}$, $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \in Q^{\mathcal{A}'} \times Q^{\mathcal{C}^f}$ tels que $q_1^{\mathcal{A}} Z^f (q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid)$. Supposons qu'il existe un état $(q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid) \in Q^{\mathcal{A}'} \times Q^{\mathcal{C}^f}$ tel que $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow_{Del^\circ(\mathcal{A}'\times\mathcal{C}^f)}^{\{a\}} (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$.

Sachant que $q_1^{\mathcal{A}} Z^f (q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid)$ alors par définition de Z^f on a : $q_1^{\mathcal{A}} Z (q_1^{\mathcal{A}'}, q_1^{\mathcal{C}})$.

Il existe deux cas possibles pour a : $a \in \Sigma$ et $a \in \{!, ?\} \times Port$.

Cas où $a \in \Sigma$

On sait que $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow^{\{a\}}_{Del^\circ(\mathcal{A}' \times \mathcal{C}^f)} (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$ alors par définition de Del° on a : $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow^{\{a\}}_{\mathcal{A}' \times \mathcal{C}^f} (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$. Cela implique, par définition du produit synchrone, que :

1. $q_1^{\mathcal{A}'} \rightarrow^{\{a\}}_{\mathcal{A}'} q_2^{\mathcal{A}'}$ et
2. $\mid q_1^{\mathcal{C}} \mid \rightarrow^{\{a\}}_{\mathcal{C}^f} \mid q_2^{\mathcal{C}} \mid$.

De plus, $\mathcal{C}$ boucle sur Σ alors $\mid q_2^{\mathcal{C}} \mid = \mid q_1^{\mathcal{C}} \mid$.

Sachant que $\mathcal{C}$ boucle sur Σ et que $q_1^{\mathcal{A}'} \rightarrow^{\{a\}}_{\mathcal{A}'} q_2^{\mathcal{A}'}$, alors par définition du produit synchrone on a : $(q_1^{\mathcal{A}'}, q_1^{\mathcal{C}}) \rightarrow^{\{a\}}_{\mathcal{A}' \times \mathcal{C}} (q_2^{\mathcal{A}'}, q_1^{\mathcal{C}})$. Par définition de Del° on obtient : $(q_1^{\mathcal{A}'}, q_1^{\mathcal{C}}) \rightarrow^{\{a\}}_{Del^\circ(\mathcal{A}' \times \mathcal{C})} (q_2^{\mathcal{A}'}, q_1^{\mathcal{C}})$. Sachant que $q_1^{\mathcal{A}} Z (q_1^{\mathcal{A}'}, q_1^{\mathcal{C}})$ et que Z est une bisimulation entre $\mathcal{A}$ et $Del^\circ(\mathcal{A}' \times \mathcal{C})$. Alors il existe $q_2^{\mathcal{A}} \in Q^{\mathcal{A}}$ tel que $q_1^{\mathcal{A}} \rightarrow^{\{a\}}_{\mathcal{A}} q_2^{\mathcal{A}}$ et $q_2^{\mathcal{A}} Z (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$. Par définition de Z^f on a : $q_2^{\mathcal{A}} Z^f (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$.

Ainsi, nous avons montré qu' il existe $q_2^{\mathcal{A}} \in Q^{\mathcal{A}}$ tel que $q_1^{\mathcal{A}} \rightarrow^{\{a\}}_{\mathcal{A}} q_2^{\mathcal{A}}$ tel que $q_2^{\mathcal{A}} Z^f (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$.

Cas où $a \in \{!, ?\} \times Port$

Considérons le cas où $a = !\pi$ avec $\pi \in Port$, (le raisonnement est le même pour le cas où $a = ?\pi$). On sait que $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow^{\{!\pi\}}_{Del^\circ(\mathcal{A}' \times \mathcal{C}^f)} (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$ alors par définition de Del° on a deux cas possibles.

(1) $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow^{\{!\pi\}}_{\mathcal{A}' \times \mathcal{C}^f} (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$ et

(2) $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow^{\{!\pi^\circ\}}_{\mathcal{A}' \times \mathcal{C}^f} (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$

Pour le cas (1), le raisonement est le même que celui présenté dans le cas où $a \in \Sigma$.

Considérons le cas (2). Sachant $(q_1^{\mathcal{A}'}, \mid q_1^{\mathcal{C}} \mid) \rightarrow^{\{!\pi^\circ\}}_{\mathcal{A}' \times \mathcal{C}^f} (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$ alors par définition du produit synchrone on a :

1. $q_1^{\mathcal{A}'} \rightarrow^{\{!\pi^\circ\}}_{\mathcal{A}'} q_2^{\mathcal{A}'}$ et
2. $\mid q_1^{\mathcal{C}} \mid \rightarrow^{\{!\pi^\circ\}}_{\mathcal{C}^f} \mid q_2^{\mathcal{C}} \mid$.

Montrons que $q_1^{\mathcal{A}} \rightarrow^{\{!\pi\}}_{\mathcal{A}} q_2^{\mathcal{A}'}$. Pour cela, utilisons la définition de $\mathcal{C}^f$. Sachant que $\mid q_1^{\mathcal{C}} \mid \rightarrow^{\{!\pi^\circ\}}_{\mathcal{C}^f} \mid q_2^{\mathcal{C}} \mid$, $q_1^{\mathcal{A}'} \rightarrow^{\{!\pi^\circ\}}_{\mathcal{A}'} q_2^{\mathcal{A}'}$ et $q_1^{\mathcal{A}} Z (q_1^{\mathcal{A}'}, q_1^{\mathcal{C}})$ alors il existe $q_2^{\mathcal{A}} \in Q^{\mathcal{A}}$ tel que $q_1^{\mathcal{A}} \rightarrow^{\{!\pi\}}_{\mathcal{A}} q_2^{\mathcal{A}}$ et $q_2^{\mathcal{A}} Z (q_2^{\mathcal{A}'}, q_2^{\mathcal{C}})$. À partir de la définition de Z^f on déduit que $q_2^{\mathcal{A}} Z^f (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$.

Par conséquent, nous avons montré qu'il existe $q_2^{\mathcal{A}} \in Q^{\mathcal{A}}$ tel que $q_1^{\mathcal{A}} \rightarrow^{\{!\pi\}}_{\mathcal{A}} q_2^{\mathcal{A}}$ et $q_2^{\mathcal{A}} Z^f (q_2^{\mathcal{A}'}, \mid q_2^{\mathcal{C}} \mid)$.

À partir des deux cas que nous avons considéré pour a, on obtient que $Z^{f^{-1}}$ est une relation de simulation entre $\mathcal{A}$ et $Del^{\circ}(\mathcal{A}' \times \mathcal{C}^f)$. □

Bibliographie

[ABH+02] A. Ankolekar, M. H. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, D. V. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, and K. P. Sycara. Daml-s : Web service description for the semantic web. In *Proceedings of the 1st International Semantic Web Conference (ISWC)*, pages 348–363, 348-363, 2002. Springer.

[ACKM03] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services - Concepts, Architectures and Applications*. Springer, 2003.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2) :183–235, 1994.

[AN01] A. Arnold and D. Niwinski. *Rudiments of mu-calculus*. North-Holland, 2001.

[AVW03] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303(1) :7–34, 2003.

[BCF07] P. Balbiani, F. Cheikh, and G. Feuillade. Considérations relatives à la décidabilité et à la complexité du problème de la composition de services. In *Proceedings of the Journées Francophones MODELES FORMELS de l'INTERACTION (MFI 2007)*, pages 261–268, Paris, France, 2007. Annales du LAMSADE.

[BCF08a] P. Balbiani, F. Cheikh, and G. Feuillade. Composition of interactive web services based on controller synthesis. *2nd International Workshop on Web Service Composition and Adaptation (WSCA 08)*, 0(0) :521–528, 2008.

[BCF08b] P. Balbiani, F. Cheikh, and G. Feuillade. Composition of web services : algorithms and complexity. *1st Interaction and Concurrency Experience (ICE 08)*, 0 :96–107, 2008.

[BCF09] P. Balbiani, F. Cheikh, and G. Feuillade. Résultats de complexité pour le problème de la composition d'agents. Cinquièmes Journées Francophones MODELES FORMELS de l'INTERACTION (MFI 09), to appear, 2009.

[BCG+05] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic composition of transition-based semantic web services with messaging. In *Proceedings of the 31st VLDB Conference. on Very Large Data Bases (VLDB 05)*, pages 613–624, Tronheim, Norway, 2005. ACM.

[BCGM06] D. Berardi, D. Calvanese, G. De Giacomo, and M. Mecella. Composing web services with nondeterministic behavior. In *Proceedings of the International Conference on Web Services (ICWS 06).*, pages 909–912, Chicago, Illinois, USA, 2006. IEEE.

[BCHK09] P. Balbiani, F. Cheikh, P. Héam, and O. Kouchnarenko. Composition of services with constraints. *Being published*, 2009.

[BCR01] P. Bertoli, A. Cimatti, and M. Roveri. Conditional planning under partial observability as heuristic-symbolic search in belief space. In *Proceedings of the 6th European Conference on Planning (ECP 01)*, pages 379–384, Toledo, Spain, 2001.

[BdMRV01] P. Blackburn, de M. Rijke, and Y. Venema. *Modal logic*. Cambridge University Press, New York, NY, USA, 2001.

[Ber05] D. Berardi. *Automatic Service Composition Models, Techniques and Tools*. PhD thesis, Università degli Studi di Roma"La Sapienza", Roma, Italy, 2005.

[BFDP08] D. Berardi, F.Cheikh, G. DeGiacomo, and F. Patrizi. Automatic service composition via simulation. *International Journal of Foundations of Computer Science*, 19(2) :429–451, 2008.

[BIM95] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *International Journal of the ACM*, 42(1) :232–268, 1995.

[BJWW02] C. Bettini, S. Jajodia, X. Wang, and D. Wijesekera. Obligation monitoring in policy management. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY 02)*, pages 2–12, Washington, DC, USA, 2002. IEEE.

[BK06] S. Basu and R. Kumar. Quotient-based approach to control of nondeterministic discrete-event systems with μ-calculus specification. Submited in American Control Conference, 2006.

[BNP08] J. Ben-Naim and H. Prade. Evaluating trustworthiness from past performances : Interval-based approaches. In *Proceedings of the 2nd international conference on Scalable Uncertainty Management (SUM 08)*, pages 33–46, Naples, Italy, 2008. Springer.

[BPT06] P. Bertoli, M. Pistore, and P. Traverso. Automated web service composition by on-the-fly belief space search. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS 06)*, pages 358–361, Cumbria, UK, 2006. AAAI.

[BS02] SAP Sun Microsystems BEA Systems, Intalio. Web service choreography interface (wsci) 1.0. Technical report, BEA Systems, Intalio, SAP, Sun Microsystems, 2002.

[CCMW01] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (wsdl) 1.1. 2001.

[CGJ+03] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *International Journal of the ACM*, 50(5) :752–794, 2003.

[CGM06] F. Cheikh, G. De Giacomo, and M. Mecella. Automatic web services composition in trustaware communities. In *Proceedings of the 3rd Internationa ACM workshop on Secure web services (SWS 06)*, pages 43–52, Alexandria, USA, 2006. ACM.

[Cha07] Y. Charif. *Chorégraphie dynamique de services basée sur la coordination d'agents introspectifs*. PhD thesis, Université Pierre et Marie Curie, Paris, France, 2007.

[CHR91] Z. Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5) :269–276, 1991.

[CMR08] Y. Chevalier, M. A. Mekki, and M. Rusinowitch. Automatic composition of services with security policies. In *Proceedings of the 2008 IEEE Congress on Services (SERVICES 08)*, pages 529–537, Washington, DC, USA, 2008. IEEE.

[CS06] G. C. Cassandras and S. Lafortune Stephane. *Introduction to Discrete Event Systems*. Springer, 2006.

[DHHvdT04] M. Dastani, A. Herzig, J. Hulstijn, and L.ert W. N. van der Torre. Inferring trust. In *Proceedings of the 5th International Workshop, CLIMA V*, pages 144–160, Lisbon, Portugal, 2004. Springer.

[DLS06] S. Demri, F. Laroussinie, and P. Schnoebelen. A parametric analysis of the state-explosion problem in model checking. *Proceedings of the 19th Symposium on Theoretical Aspects of Computer Science (STACS 02)*, 2285 :620–631, 2006.

[DM04] J. Hobbs O. Lassila D. McDermott S. McIlraith S. Narayanan M. Paolucci B. Parsia T. Payne E. Sirin N. Srinivasan K. Sycara D. Martin, M. Burstein. Owl-s : Semantic markup for web services. Technical report, France Telecom, Maryland Information and Network Dynamics Lab at the University of Maryland, National Institute of Standards and Technology (NIST), Network Inference, Nokia, SRI International, Stanford University, Toshiba Corporation, and University of Southampton, 2004.

[Dra01] V. Draluk. Discovering web services : An overview. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 01)*, pages 637–640, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[FP07] G. Feuillade and S. Pinchinat. Modal specifications for the control theory of discrete event systems. *Discrete Event Dynamic Systems*, 17(2) :211–232, 2007.

[GdLMP07] G. De Giacomo, M. de Leoni, M. Mecella, and F. Patrizi. Automaticworkflows composition of mobile services. In *Proceedings of the IEEE International Conference on Web Services (ICWS 07)*, pages 823–830, Salt Lake City, Utah, USA, 2007. IEEE.

[GJ90] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. H. Freeman and Company, 1990.

[GPR08] N. Guermouche, O. Perrin, and C. Ringeissen. Timed specification for web services compatibility analysis. *Electronic Notes in Theoretical Computer Science (ENTCS 08)*, 200(3) :155–170, 2008.

[GV92] J. F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2) :202–260, 1992.

[HB03] R. Hamadi and B. Benatallah. A petri net-based model for web service composition. In *Proceedings of the 14th Australasian database conference (ADC 03)*, pages 191–200, Adelaide, Australia, 2003. Australian Computer Society.

[HBP07] J. Hoffmann, P. Bertoli, and M. Pistore. Web service composition as planning, revisited : In between background theories and initial state uncertainty. In *Proceedings of the 22nd National Conference of the American Association for Artificial Intelligence (AAAI 07)*, pages 1013–1018, Vancouver, British Columbia, Canada, 2007. AAAI.

[Hoa78] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8) :666–677, 1978.

[Hoa85] C. A. R. Hoare. *Communicating sequential processes*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985.

[HS96] H. Hüttel and S. Shukla. On the complexity of deciding behavioural equivalences and preorders, a survey. Technical report, Department of Computer Science University of Aarhus, 1996.

[IBM07] Microsoft SAP AG Siebel Systems IBM, BEA Systems. Business process execution language for web services version 1.1. 2007.

[Jan94] P. Jancar. Decidability questions for bisimilarity of petri nets and some related problems. In *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science (STACS 94)*, pages 581–592, Caen, France, 1994. Springer.

[Jan95] P. Jančar. Undecidability of bisimilarity for petri nets and some related problems. *Theoretical Computer Science*, 148(2) :281–301, 1995.

[JC] Webopedia Jupitermedia Corporation. Online computer dictionary for computer and internet terms. *http ://www.webopedia.com/*.

[JK06] S. Jiang and R. Kumar. Supervisory control of discrete event systems with ctl* temporal logic specifications. *SIAM Journal on Control and Optimization*, 44(6) :2079–2103, 2006.

[JM96] L. Jategaonkar and A. R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154(1) :107–143, 1996.

[KBR04] N. Kavantzas, D. Burdet, and G. Ritzinger. Web services choreography description language version 1.0. In *http ://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/*, 2004.

[KG95] R. Kumar and V. K. Garg. *Modeling and Control of Logical Discrete Event Systems*. Kluwer, 1995.

[Koz97] D. Kozen. *Automata and Computability*. Springer, 1997.

[KPP06] R. Kazhamiakin, P. K. Pandya, and M. Pistore. Timed modelling and analysis in web service compositions. In *Proceedings of the 1st International Conference on Availability, Reliability and Security (ARES 06)*, pages 840–846, Vienna University of Technology, Austria, 2006. IEEE.

[LPT02] U. D. Lago, M. Pistore, and P. Traverso. Planning with a language for extended goals. In *Proceedings of the 18th International Conference on the American Association for Artificial Intelligence (AAAI 02)*, pages 447–454, Alberta, Canada, 2002. AAAI.

[LS00] F. Laroussinie and P. Schnoebelen. The state explosion problem from trace to bisimulation equivalence. In *Proceedings of the 3rd International Conference on Foundations of Software Science and Computation Structures (FOSSACS 00)*, pages 192–207, Berlin,Germany, 2000. Springer.

[Mei74] M. P. Meir. *A study of the recoverability of computing systems*. PhD thesis, University of California, Irvine, California, USA, 1974.

[Mil95] R. Milner. *Communication and concurrency*. Prentice Hall International (UK) Ltd, Upper Saddle River, NJ, USA, 1995.

[MKB06] I. Müller, R. Kowalczyk, and P. Braun. Towards agent-based coalition formation for service composition. In *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology (IAT 06)*, pages 73–80, Washington, DC, USA, 2006. IEEE.

[MKB07] S. Mitra, R. J. Kumar, and S. Basu. Automated choreographer synthesis for web services composition using i/o automata.

In *Proceedings of the IEEE International Conference on Web Services (ICWS 2007)*, pages 364–371, Salt Lake City, Utah, USA, 2007. IEEE.

[MM07] M. Morge and P. Mancarella. Argumentation-based decision making for selecting communication services in ambient home environments. In *Proceedings of the Symposium on Artificial Societies for Ambient Intelligence*, pages 46–49, Newcastle University, UK, 2007.

[MMM06] F. McCabe P. Brown M. MacKenzie, K. Laskey and R. Metz. Reference model for service-oriented architecture 1.0. Technical report, Advancing open standards for the information society (OASIS), 2006.

[MPPP02] M. Mecella, F. Parisi-Presicce, and B. Pernici. Modeling e -service orchestration through petri nets. In *Proceedings of the 3rd VLDB International Workshop on Technologies for E-Services (VLDB-TES 2002)*, pages 38–47, Hong Kong, China, 2002. Springer.

[MPPT07] A. Marconi, M. Pistore, P. Poccianti, and P. Traverso. Automatedweb service composition at work : the amazon/mps case study. In *Proceedings of the IEEE International Conference on Web Services (ICWS 07)*, pages 767–774, Salt Lake City, Utah, USA, 2007. IEEE.

[MPT08] A. Marconi, M. Pistore, and P. Traverso. Automated composition of web services : the astro approach. *IEEE Data Engineering Bulletin*, 31(3) :23–26, 2008.

[MS72] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the Annual Symposium on Switching and Automata Theory.*, pages 125–129, 1972.

[Mur89] T. Murata. Petri nets : Properties, analysis and applications. *Proceedings of the IEEE*, 77(4) :541–580, 1989.

[MW08] A. Muscholl and I. Walukiewicz. A lower bound on web services composition. In *Proceedings of the 10th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 07)*, pages 274–286, Braga, Portugal, 2008. Springer.

[Opr07] M. J. Oprescu. *Découverte et composition de services dans des réseaux ambiants*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, 2007.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, pages 167–183, Prague, Czech Republic, 1981. Springer.

[PBL08] V. Prêtre, F. Bouquet, and C. Lang. Automating uml models merge for web services testing. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications Services (iiWAS 08)*, pages 55–62, Linz, Austria, 2008. ACM.

[PBLH06] J. Pathak, S. Basu, R. Lutz, and V. Honavar. Parallel web service composition in moscoe : A choreography-based approach. In *Proceedings of the European Conference on Web Services (ECOWS 06)*, pages 3–12, Washington, DC, USA, 2006. IEEE.

[Pet62] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.

[PMBT05] M. Pistore, A. Marconi, P. Bertoli, and P. Traverso. Automated composition of web services by planning at the knowledge level. In *Proceedings of the 2008 ACM symposium on Applied computing (IJCAI-05)*, pages 2279–2285, Fortaleza, Ceara, Brazil, 2005. ACM.

[PR05a] S. Pinchinat and J. Raclet. Supervisory control problems for nondeterministic discrete-event systems : A logical approach. *16th IFAC World Congress 2005*, 2005.

[PR05b] S. Pinchinat and S. Riedweg. A decidable class of problems for control under partial observation. *Information Processing Letters*, 95(4) :454–460, 2005.

[PTB05] M. Pistore, P. Traverso, and P. Bertoli. Automated composition of web services by planning in asynchronous domains. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 2–11, California, USA, 2005. AAAI.

[PTBM05] M. Pistore, P. Traverso, P. Bertoli, and A. Marconi. Automated synthesis of composite bpel4ws web services. In *Procee-*

dings of the IEEE International Conference on Web Services (ICWS 05), pages 293–301, Orlando, FL, USA, 2005. IEEE.

[Ram74] C. Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

[RKM04] J. Rao, P. Küngas, and M. Matskin. Logic-based web services composition : From service description to process model. In *Proceedings of the IEEE International Conference on Web Services (ICWS 04)*, pages 446– 453, Washington, DC, USA, 2004. IEEE.

[RW89] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1) :81–98, 1989.

[SH05] M. Singh and M. N. Huhns. *Service-Oriented Computing - Semantics, Processes, Agents*. John Wiley & sons, Inc., 2005.

[SPG08] S. Sardiña, F. Patrizi, and G. De Giacomo. Behavior composition in the presence of failure. In *Proceedings of Principles of Knowledge Representation and Reasoning (KR 08)*, pages 640–650, Sydney, Australia, 2008. AAAI.

[SPM06] S. Sohrabi, N. Prokoshyna, and S. A. McIlraith. Web service composition via generic procedures and customizing user preferences. In *Proceedings of the 5th International Semantic Web Conference (ISWC 06)*, pages 597–611, Athens, GA, USA, 2006. Springer.

[TPC+05] M. Trainotti, M. Pistore, G. Calabrese, G. Zacco, G. Lucchese, F. Barbon, P. Bertoli, and P. Traverso. Astro : Supporting composition and execution of web services. In *Proceedings of the 3rd International Conference on Service-Oriented Computing (ICSOC 2005)*, pages 495–501, Amsterdam, The Netherlands, 2005. Springer.

[Tsi89] J.N. Tsitsiklis. On the control of discrete-event dynamical systems. *Mathematics of Control, Signals and Systems*, 2(2) :95–107, 1989.

[vG90] R. J. van Glabbeek. The linear time-branching time spectrum (extended abstract). In *Proceedings of the Theories of Concurrency : Unification and Extension(CONCUR 90)*, pages 278–297, Amsterdam, 1990. Springer.

[ZKJ06] C. Zhou, R. Kumar, and S. Jiang. Control of nondeterministic discrete-event systems for bisimulation equivalence. *IEEE transactions on automatic control*, 51(5) :754– 765, 2006.

[ZNB+08] L. Zeng, A. H. H. Ngu, B. Benatallah, R. M. Podorozhny, and H. Lei. Dynamic composition and optimization of web services. *Distributed and Parallel Databases*, 24(1) :45–72, 2008.

Index

Composition de services: algorithmes et complexité

Fahima CHEIKH

Le problème de la combinaison des services, autrement appelé problème de la composition, constitue le foyer d'une intense activité de recherche. Composer les services entre eux, c'est entrelacer leurs séquences d'actions, de manière à obtenir des séquences qui satisfassent les exigences des clients. Le problème de la composition de services est difficile à résoudre en général.

Dans cette thèse nous considérons des services qui peuvent à la fois exécuter des actions de communications ainsi que des actions internes. De plus, des conditions peuvent être exigées et des effets peuvent être appliqués sur les transitions. Formellement, les services sont représentés par des automates communicants conditionnels.

Nous définissons, pour ce modèle, le problème de la composition et nous étudions sa décidabilité pour différentes relations d'équivalence et de préordre à savoir : l'inclusion de traces, l'équivalence de traces, la simulation et la bisimulation.

Suite aux résultats de décidabilité obtenus, nous proposons trois variantes pour le modèle initial. Pour chacune d'elles, nous définissions le problème de la composition et nous étudions sa complexité pour les relations citées ci-dessus.

Mots-clès

Composition de services, automates, relations d'équivalence et de préordre, décidabilité et complexité.

Université de Toulouse – Institut de Recherche en Informatique de Toulouse
118 route de Narbonne, 31062 Toulouse Cedex 9, France