UNIVERSITÉ DE MONTRÉAL

PRÉSERVATION DE L'OPACITÉ PAR RAFFINEMENT DE SYSTÈMES SPÉCIFIÉS PAR DES CHAÎNES DE MARKOV DISCRÈTES À INTERVALLES

GAËTAN DUPEUBLE
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
MAI 2017

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

PRÉSERVATION DE L'OPACITÉ PAR RAFFINEMENT DE SYSTÈMES SPÉCIFIÉS PAR DES CHAÎNES DE MARKOV DISCRÈTES À INTERVALLES

présenté par : DUPEUBLE Gaëtan
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. KHOMH Foutse, Ph. D., président
M. MULLINS John, Ph. D., membre et directeur de recherche
M. TAHAR Sofiène, Ph. D., membre

DÉDICACE

À ma famille
À Alicia

REMERCIEMENTS

"Tous les hommes pensent que le bonheur se trouve au sommet de la montagne alors qu'il réside dans la façon de la gravir."

— Confucius

Ce mémoire conclut une aventure sur les bancs de la recherche formelle. Comme toute aventure, elle a eu son lot de rebondissements, d'émotions. Mais surtout, comme toute aventure, elle a eu son lot de protagonistes.

À l'heure du bilan, je tiens à remercier tout particulièrement mon directeur de recherche, Monsieur John Mullins, qui a cru en moi dès notre première rencontre, et m'a guidé et soutenu depuis lors. Je salue son enthousiasme communicatif ainsi que sa motivation sans égale. Un grand merci également à Monsieur Foutse Khomh, président du jury, ainsi qu'à Monsieur Sofiène Tahar, membre du jury, pour avoir accepté de prendre sur leur temps pour évaluer le présent mémoire.

Je tiens également à remercier mes parents qui, malgré l'océan qui nous sépare, sont toujours là pour m'écouter et me rassurer.

Enfin, je tiens à remercier mon âme sœur Alicia, pour sa relecture avisée d'une part, pour son soutien et sa présence sans faille au quotidien, d'autre part, malgré la distance.

RÉSUMÉ

Les méthodes formelles permettent de modéliser et concevoir des systèmes informatiques critiques, notamment dans les domaines à fort risque humain que sont les transports de personne ou les centrales énergétiques, par exemple. L'une des méthodes de conception est celle dite de raffinements successifs, étapes lors desquelles les spécifications du système sont ajustées afin que le produit final soit le plus conforme possible aux exigences initiales. Le principe du raffinement est tel qu'il ne doit pas être destructif : le modèle raffiné doit vérifier au moins les mêmes requis déjà validés par le modèle précédent – par exemple, l'absence de blocage, ou la terminaison du programme dans un état acceptant.

Parmi ces requis, le système doit parfois valider des requis non-fonctionnels, tels que des propriétés de sécurité. Notamment, on se penche davantage sur la propriété d'opacité libérale.

Pour modéliser les systèmes informatiques ainsi que de tels requis non-fonctionnels, on a besoin de méthodes quantitatives. Ainsi, nous choisissons comme cadre théorique le modèle de la Chaîne de Markov discrète à Intervalles (IDTMC). Ce modèle a pour intérêt d'avoir un aspect non-déterministe. En réalité, c'est une extension du modèle de Système de Transitions Probabilistes (PTS) : en ce sens, on considère qu'une IDTMC représente une spécification, que l'on peut implémenter par un PTS. Les PTS eux-mêmes sont des modèles probabilistes, qui permettent la mesure de propriétés quantitatives. Le second avantage de ce type de modèle est l'existence de trois types de raffinement : fort, faible et complet.

La problématique principale liée au raffinement de systèmes sécurisés est la suivante : le fait qu'une spécification vérifie une propriété de sécurité donnée n'est pas une condition nécessaire au fait que son raffinement la vérifie également. Le but est donc de trouver, dans notre cadre théorique, une notion de raffinement qui préserve la propriété de sécurité que l'on étudie.

L'opacité est une propriété de sécurité introduite avec le modèle du Système de Transitions Étiquetées (LTS), puis étendue aux PTS : elle traduit la capacité d'un observateur extérieur à déduire l'état d'un prédicat secret en observant uniquement la partie publique des exécutions du programme. Sa première définition est une définition binaire ; en étendant la notion aux PTS, on introduit un aspect probabiliste en définissant l'opacité libérale, qui mesure la non-opacité du système, et l'opacité restrictive, qui mesure son opacité effective. Il est alors possible d'étendre à nouveau ces notions aux IDTMC : il suffit de calculer l'opacité dans le pire des cas pour l'ensemble des implémentations des IDTMC.

Ainsi, nous prouvons les résultats suivants.

Tout d'abord, on prouve que l'opacité libérale dans une IDTMC non-modale, c'est-à-dire complètement définie, se calcule en un temps fini, doublement exponentiel. Nous proposons un algorithme de calcul.

De plus, on prouve qu'il est possible d'approcher l'opacité libérale dans une IDTMC dans le cas général, en un temps doublement exponentiel également. Nous proposons comme contribution originale une extension de l'algorithme de calcul du cas non-modal, et nous prouvons sa correction.

Enfin, on prouve que l'opacité libérale dans une spécification est préservée après raffinement faible, ce qui généralise un résultat similaire mais qui ne considérait que le raffinement fort.

En définitive, nous réalisons une preuve de concept destinée à être reproduite pour d'autres modèles et propriétés de sécurité similaires, telles que les Propriétés Rationnelles de Flux d'Information (RIFP) dont est issue l'opacité.

ABSTRACT

Formal methods can help to design any computer system – softwares, protocols, architectures, *etc.* Indeed, developping a system usually consists in refining it. The refined system is then a more precise one, with some more features. Thus, all these stages lead to a final product which is a working implementation of the initial specification. The key issue is as follows: each refined system must at least verify all the properties verified by the previous one. This must be the case for behaviour properties – like the absence of any deadlock – and for security properties.

This issue is relatively easily resolved when it is about usual behaviour properties, but security is trickier to model. Therefore, one cannot ensure the fact that a refined system verifies the same security properties as the previous system.

This essay aims to highlight a particular security property, opacity, for which we prove that it is preserved when a system is refined. Opacity is linked to the probability for a passive external observer to know the content of a secret, only by observing the public outputs of the system.

The framework is as follows. In order to modelize our specifications, we define the Interval Discrete-Time Markov Chain (IDTMC), which is a generalisation of the Probabilistic Transition System (PTS). The probabilistic aspect is a way to introduce quantitative measurement on our models. Since IDTMC are non-deterministic, they carry a higher layer of abstraction than the PTS model. On this framework, one can define three types of refinement: strong, weak and thorough.

Since opacity is already defined on PTSs, we define its extension to IDTMC. Particularly, one can differentiate liberal opacity – the measure of non-opacity – from restrictive opacity – the measure of effective opacity. The extension is directly defined by stating the fact that the opacity of a secret in a IDTMC is the worst case among all the PTSs that implement this specification.

Then we prove the following theorems.

First, if we consider a non-modal IDTMC, *i.e.* a specification for which each transition has a non-zero probability, then the liberal opacity of any secret is computable in 2EXP-time. We provide an algorithm to compute this value.

Then, for the general case, we prove that the liberal opacity can be approximate in 2EXP-time. This original contribution comes with an extension of the previous algorithm, for which

we prove its correctness.

Finally, we solve the main issue of this essay: liberal opacity in a specification is preserved when the system is weakly refined. This contribution expands a similar result, which only considered strong refinement.

These results lead to a proof of concept for the fact that secured systems can be refined and keep their security properties, for a certain type of properties. This can be especially generalised to all Rational Information Flow Properties (RIFP).

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

BFS	Solution Basique Réalisable
BSP	Prédicat Basique de Sécurité
CMC	Chaîne de Markov à Contraintes
DPA	Automate de Parité Déterministe
IDTMC	Chaîne de Markov discrète à Intervalles
IMDP	Processus de Décision Markovien à Intervalles de probabilité
LTS	Système de Transitions Étiquetées
MC	Chaîne de Markov
MDP	Processus de Décision Markovien
PTS	Système de Transitions Probabilistes
RIFP	Propriété Rationnelle de Flux d'Information

CHAPITRE 1 INTRODUCTION

Que ce soit pour comprendre le fonctionnement d'un système ou pour concevoir un produit industriel, il est courant de faire appel à des méthodes de modélisation formelle. Cela consiste à dégager dans un premier temps les caractéristiques principales, en écartant les concepts secondaires, *a priori* inutiles pour comprendre le fonctionnement du système. On obtient alors un modèle théorique qui décrit le comportement du système, mais qui ne le représente pas exactement. Afin de minimiser l'écart entre le système et le modèle, on a recours, lors de la gestion de projet, au processus de raffinement : le modèle est constamment amélioré, et les caractéristiques fonctionnelles initialement écartées sont progressivement étudiées et ajoutées. Ceci est une manière de passer d'une idée théorique à un produit réel implémentant cette idée.

1.1 Éléments de la problématique

En génie logiciel et informatique, les méthodes formelles sont généralement utilisées pour modéliser, concevoir et vérifier des systèmes critiques, notamment dans des domaines tels que le ferroviaire ou l'avionique, où les risques humains demandent une résistance parfaite aux failles. Le raffinement est une part importante de ces méthodes, puisque la conception formelle d'un système passe par une succession d'étapes ajoutant régulièrement des fonctionnalités. Il est alors nécessaire de s'assurer qu'aucune étape du raffinement ne brise le fonctionnement du système. Autrement dit, il est nécessaire de s'assurer, pour chaque étape, que le nouveau modèle vérifie au moins les mêmes propriétés que l'ancien. On parle notamment de raffinement de requis fonctionnels – par exemple, l'absence de blocage, ou la garantie que le train s'arrête lors d'un feu rouge.

Ces requis fonctionnels ne sont cependant pas les seuls requis que peut comporter le cahier des charges d'un système. On distingue également les requis non-fonctionnels. Par exemple, si l'on considère un fournisseur de services d'infonuage, l'un des requis non-fonctionnels classiques est l'obligation de fournir une qualité de service quasiment parfaite aux clients. Il est courant alors de quantifier le temps maximal autorisé de baisse de service durant une période donnée, ou bien la probabilité qu'une panne se déroule. De façon naturelle, la modélisation de ces requis se réalise donc grâce à des méthodes quantitatives.

Une problématique particulière se pose alors : est-il possible de raffiner les modèles quantitatifs, nécessaires à la modélisation de ces requis non-fonctionnels, tels que la sécurité ?

Habituellement, la sécurité est considérée à travers trois axes principaux : la confidentialité, l'intégrité et la disponibilité. Garantir ces trois notions revient à garantir la sécurité du système considéré. Un utilisateur malhonnête désirant attaquer l'un de ses trois aspects est communément appelé adversaire ou attaquant.

- La *confidentialité* englobe tout ce qui est lié à l'aspect privé d'un ensemble de données. Le but est de garantir que seuls les utilisateurs autorisés peuvent intéragir avec les données que l'on veut garder confidentielles.
- L'*intégrité* concerne la modification des données. Le but est de garantir que les données sont à tout moment complètes et lisibles. Notamment, une modification peut être réalisée uniquement par un utilisateur autorisé.
- La *disponibilité* concerne l'accès aux données ou aux services. Le but est de garantir l'accès aux données à tout utilisateur autorisé. Cette troisième propriété est généralement vue comme orthogonale aux deux précédentes, puisqu'elle doit garantir un accès, alors que les autres doivent restreindre l'accès.

Ces trois axes permettent de positionner toute problématique de sécurité dans l'industrie – par exemple, une tentative de déni de service s'attaque à la disponibilité du système, tandis qu'un adversaire essayant d'accéder à une base de données privées s'attaque à la confidentialité du système, voir à son intégrité.

L'expansion des échanges commerciaux réalisés par l'entremise des internets, les données confidentielles hébergées sur des serveurs infonuagiques, ou encore la quantité de services nécessitant un accès fiable et continu aux réseaux internets, posent quantité de problèmes concrets de sécurité : comment être certain de l'identité et de l'honnêteté du serveur qui reçoit nos coordonnées bancaires ? L'accès à nos données confidentielles est-il réellement contrôlé et inviolable ? Jusqu'à quel point pouvons-nous assurer la disponibilité de nos services ?

La modélisation de la sécurité s'avère généralement complexe. Alors que les propriétés de comportement – telles que l'absence de blocage ou l'absence de famine – sont des propriétés portant sur des ensembles de traces d'exécution du système, la sécurité d'un système se modélise par des hyperpropriétés, c'est-à-dire des propriétés portant sur des ensembles d'ensembles de traces : il est nécessaire d'ajouter un niveau d'abstraction supplémentaire par rapport aux propriétés modélisant les requis fonctionnels (Clarkson et Schneider, 2010). Cet ajout de complexité induit notamment que ces propriétés ne sont généralement pas conservées lorsque le système est raffiné. Autrement dit, si un modèle vérifie une propriété de sécurité, il n'est pas nécessaire que les modèles qui le raffinent vérifient la même propriété de sécurité. Ceci pose un problème majeur qui remet en cause la méthodologie de conception évoquée jusqu'alors, puisqu'il n'est plus possible d'assurer la sécurité du produit final en vérifiant uniquement les propriétés de sécurité sur le modèle initial.

Il s'avère que certaines propriétés de sécurité ont tout de même la capacité d'être préservées par raffinement. Prouvons dans ce mémoire que l'extension de l'opacité libérale (Bérard *et al.*, 2015c) à notre cadre de recherche fait partie de ces propriétés particulières.

1.2 Exemple de motivation

Illustrons ces questionnements avec un exemple de motivation simple. Considérons un protocole simplifié de transmission sécurisée de messages entre deux parties, dont le but est de garantir la confidentialité des messages échangés. Le système est constitué d'un canal de transmission et d'un système de chiffrement. On abstrait les détails techniques de ces deux sous-protocoles, ainsi que les systèmes de vérification éventuels qu'un tel protocole peut avoir – attente d'un accusé de réception, par exemple.

Puisqu'aucun système n'est parfait, on considère la possibilité d'une faille dans le système de chiffrement, qui laisserait passer des messages en clair dans le canal. Puisque l'on ne connaît pas la probabilité de cette faille de sécurité, on fait apparaître un intervalle de probabilité, qui contient *a priori* la valeur réelle.

Finalement, on modélise le protocole par l'objet $\mathcal{S}_1$ représenté sur la figure 1.1a. Les différents états du modèle (attente, transmission en clair, transmission chiffrée) traduisent les différentes étapes possibles lors de la transmission d'un message. Les flèches représentent les transitions entre les différents états, c'est-à-dire la manière dont peut s'exécuter le modèle ; elles contiennent les intervalles de probabilité. Notamment, on distingue ici deux boucles d'exécution : lorsque le système est en *attente*, il a la possibilité de transmettre le message *en clair*, ce qui modélise la faille précédemment décrite, ou bien de fonctionner correctement en transmettant le message de façon *chiffrée* ; puis il revient à son état initial dès la transmission réalisée, dans l'attente d'un nouveau message à transmettre.

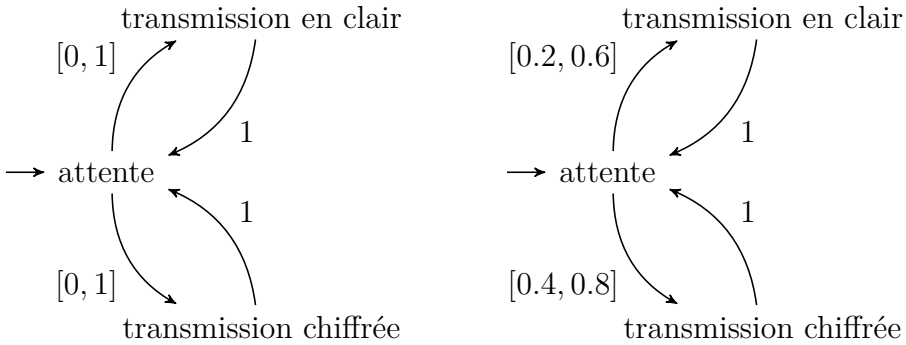
Ce modèle, que l'on appelle une spécification, est l'abstraction de toutes les implémentations du protocole décrit précédemment : il contient toutes les probabilités d'avoir une faille à chaque boucle de transmission, sans considérer les détails derrière la méthode de chiffrement ou de transmission.

Un tel modèle permet alors une étude plus poussée du système, notamment du point de vue de la sécurité. Par exemple, il paraît évident que le but est d'éviter de passer par l'état *transmission en clair* afin d'assurer le bon fonctionnement du protocole. Ce type de modèle permet l'utilisation des méthodes quantitatives : il est possible de mesurer la probabilité de passer par cet état, à la première boucle d'exécution, à la seconde, ou n'importe quand, suivant la propriété que l'on veut mesurer. On parlera dans la suite de probabilité de briser

la sécurité.

Plus précisément, puisque l'on considère une spécification non-déterministe – les probabilités sont uniquement restreintes – on mesure le pire cas possible, c'est-à-dire le maximum des probabilités de briser la sécurité. Par exemple, considérons la propriété "À la première boucle de transmission, le message est transmis en clair.", qui est un comportement qui brise la sécurité. Dans la spécification $\mathcal{S}_1$, la probabilité maximale que cette propriété soit vraie est de 1, dûe à l'intervalle de probabilité $[0, 1]$ de la transition entre les états *attente* et *transmission en clair*.

Dans une logique de conception de protocole, le but est alors de réduire ce risque de faille de sécurité. Pour cela, on utilise la notion de raffinement de spécification : le modèle initial est raffiné, c'est-à-dire qu'une nouvelle spécification est trouvée, telle que toutes ses implémentations implémentent également la spécification de départ. Autrement dit, le raffinement permet de trier parmi toutes les instances du protocole de départ afin de supprimer les pires cas. Cela est réalisé ici en réduisant les intervalles de probabilités des transitions : on obtient la spécification $\mathcal{S}_2$, représentée figure 1.1b, qui raffine de manière évidente la spécification $\mathcal{S}_1$.



(a) Modèle $\mathcal{S}_1$ (b) Modèle $\mathcal{S}_2$

Figure 1.1 Exemple de motivation

Il est alors possible de vérifier si le raffinement a effectivement amélioré la mesure de risque réalisée sur la spécification $\mathcal{S}_1$. Avec le même raisonnement, la propriété "À la première boucle de transmission, le message est transmis en clair." a cette fois-ci une probabilité d'être vraie de 0.6 dans le pire des cas.

1.3 Objectifs de recherche

Nos objectifs de recherche sont liés au problème de raffinement destructif. Le but de nos travaux consiste à :

1. définir un modèle de spécification où les requis sont non-fonctionnels et sur lequel on peut définir un raffinement ;
2. définir et calculer la mesure de sécurité d'une spécification comme étant le pire cas de ses implémentations ;
3. prouver que la propriété de sécurité est préservée par le raffinement de spécification, afin de contourner cette problématique et de montrer que la méthodologie de conception de systèmes sécurisés est justifiable.

1.4 Méthodologie

Afin de remplir ces objectifs, la méthodologie utilisée est la suivante.

Le cadre théorique consiste en l'étude du modèle de la Chaîne de Markov discrète à Intervalles (IDTMC). Ce type de spécification, dont deux exemples sont représentés sur la figure 1.1, est un modèle probabiliste non-déterministe permettant d'abstraire le système étudié. L'aspect probabiliste justifie le choix de cette méthode dans le cadre de propriétés quantitatives ; l'aspect non-déterministe permet d'étudier en un seul modèle l'ensemble des instances possibles du système. En ce sens, les IDTMC héritent des fonctionnalités de modèles quantitatifs connus que sont les Chaînes de Markov – méthode classique pour modéliser le comportement de nombreux systèmes – les Systèmes de Transitions Probabilistes (PTS), ou encore les Processus de Décision Markoviens (MDP) – modèle probabiliste non-déterministe. En pratique, une IDTMC est un PTS dont chaque probabilité de transition est remplacée par un intervalle de probabilités, laissant ainsi un choix indénombrable de distributions pour chaque état du modèle.

Sur ce choix de spécification sont définis des raffinements : afin d'améliorer le modèle d'un système, on le raffine, c'est-à-dire que l'on précise son comportement. En pratique, après raffinement, l'ensemble des implémentations possibles du modèle est réduit et est inclus dans l'ensemble des implémentations du modèle initial. Réitérer ce processus conduit fatalement à une unique implémentation, un modèle totalement déterministe qui spécifie complètement le comportement du système. Un tel modèle est alors un PTS.

On définit également une méthode d'implémentation d'une spécification, l'ordonnancement, similaire à l'ordonnancement des MDP : l'ordonnanceur exécute la spécification, et choisit, à chaque état rencontré, une distribution de probabilités parmi les distributions autorisées par la spécification. Ce moyen d'implémenter permet de simuler l'action d'un adversaire omniscient capable de manipuler le système comme bon lui semble. Étudier ceci donne ainsi les failles éventuelles laissées par la spécification aux utilisateurs.

Parmi ces failles, on se penche davantage sur des propriétés de sécurité que l'on désire mesurer. Notamment, nous étudions plus particulièrement la notion de l'opacité : de façon informelle, l'opacité mesure la probabilité pour un observateur extérieur passif de distinguer un secret d'un message quelconque – si cette probabilité est suffisamment faible, conformément aux requis non-fonctionnels, cela signifie que le secret est opaque dans le modèle. Dans une spécification, la mesure de l'opacité est sa mesure dans la pire de ses implémentations, ce qui constitue une extension de la définition d'opacité au cadre des spécifications. Plus particulièrement, on définit également la mesure dans la pire de ses implémentations ordonnancées.

Toutes ces méthodes aboutissent alors à l'étude de l'effet du raffinement de spécifications sur la mesure de l'opacité.

1.5 Plan du mémoire

Le mémoire se divise en sept chapitres. À la suite de cette introduction, une revue de littérature présente l'état de l'art en ce qui concerne les méthodes formelles de spécification et de raffinement de systèmes sécurisés.

Le chapitre 3 rappelle des résultats préliminaires et introduit les définitions nécessaires pour la suite. Notamment, les notions de système de transition et d'automate sont expliquées, ainsi que la propriété d'opacité, propriété de sécurité choisie pour nos résultats. Les notions sont expliquées en suivant la logique suivante : du modèle le plus simple au plus complet. Ainsi, les premiers systèmes de transition décrivent des systèmes hautement déterministes et très limités. Puis les modèles gagnent en généralité, en ajoutant notamment l'aspect probabiliste.

Le chapitre 4 présente le cadre théorique du mémoire. Les modèles de spécifications particuliers à notre étude, le processus de raffinement utilisé, ainsi que la généralisation du concept d'opacité à ce type de modèle sont définis.

Le chapitre 5 prouve les résultats principaux de cette étude : la décidabilité de l'opacité libérale dans le cas des spécifications non-modales, le calcul d'une approximation de l'opacité libérale dans le cas général, et la préservation de l'opacité par raffinement faible de spécification.

Le chapitre 6 consiste en un exemple pratique d'application des aspects théoriques mis en œuvre dans le chapitre 5. Un système de contrôle d'accès à une base de données est modélisé puis raffiné. Le but est de vérifier sur un exemple simple les résultats précédents, tout en montrant comment appliquer les méthodes de calcul d'opacité.

Le chapitre 7 résume les résultats et contributions, puis décrit les travaux futurs à déployer.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans ce chapitre, nous définissons les fondements du cadre formel utilisé dans ce mémoire, en nous penchant plus particulièrement sur l'étude des systèmes sécurisés. Nous listons également le comportement de différentes propriétés usuelles de sécurité relativement au raffinement de systèmes.

2.1 Raffinement et spécification

Abadi et Lamport (1991) définissent un premier cadre de modèle de spécification sur lequel on peut établir une relation de raffinement. Leurs modèles sont des machines à états potentiellement infinies, qui spécifient des requis fonctionnels, de sûreté ou de vivacité. Le principe de l'article est de prouver que le raffinement d'une spécification permet de trouver une nouvelle spécification qui implémente la première. En ce sens, le modèle raffiné spécifie de façon plus précise les requis par rapport au modèle initial. Ici, les modèles probabilistes ne sont pas utilisés : il n'y a pas de quantification de la précision de spécification des requis. En pratique, une spécification $\mathcal{S}_1$ implémente une spécification $\mathcal{S}_2$ si le comportement extérieur du modèle $\mathcal{S}_1$ fait partie du comportement extérieur du modèle $\mathcal{S}_2$. Cet article pose finalement les fondements du raffinement de machines à états.

Les résultats sont alors étendus par Jonsson et Larsen (1991), qui définissent un cadre probabiliste. Ils utilisent le modèle probabiliste du Système de Transitions Probabilistes (PTS), et l'étendent pour abstraire davantage le modèle. Ils définissent alors la Chaîne de Markov discrète à Intervalles (IDTMC), PTS dont les probabilités de transition sont remplacées par des intervalles de probabilité. Cela permet de spécifier plus généralement des requis non-fonctionnels, qui ont besoin de la quantification probabiliste. Les auteurs définissent alors la notion d'implémentation au sens utilisé dans ce mémoire, c'est-à-dire le fait que les spécifications IDTMC peuvent être vues comme des ensembles de PTS qui les implémentent. Par la suite, la relation de raffinement entre spécification est définie, et telle que si $\mathcal{S}_1$ raffine $\mathcal{S}_2$, alors l'ensemble des implémentations de $\mathcal{S}_1$ est inclus dans celui de $\mathcal{S}_2$. Les auteurs définissent également la notion de modalité de transition – et par extension, la notion de spécification modale : c'est un IDTMC qui possède au moins un arc dont la présence n'est pas obligatoire lors de l'implémentation, c'est-à-dire une transition qui peut être affectée d'une probabilité nulle.

Delahaye *et al.* (2012) continuent le travail précédent, dans le cadre des IDTMC, en définissant

plus précisément la notion de raffinement. Notamment, ils différencient les raffinements fort, faible et complet, ce qui induit une hiérarchie des spécifications plus complexe. Ils prouvent notamment que le raffinement complet est plus faible que le raffinement faible, lui-même plus faible que le raffinement fort. Ces inégalités sont strictes en règle générale. Ainsi, cet article décrit le cadre formel général de raffinement réutilisé par la suite dans ce mémoire.

Il reste à l'appliquer dans le cadre de requis non-fonctionnels.

2.2 Le cas des systèmes sécurisés

La sécurité est un exemple de requis non-fonctionnel. Par conséquent, sa modélisation en est plus complexe qu'un simple requis de sûreté ou de vivacité. Clarkson et Schneider (2010) illustrent ce fait en proposant une classification de l'ensemble des propriétés de sécurité en introduisant la notion d'hyperpropriété. Son raisonnement est le suivant. Les requis fonctionnels d'un système peuvent être représentés facilement à partir de propriétés portant sur les traces infinies d'exécution de celui-ci – formellement, une propriété de traces est l'ensemble des traces qui vérifient un comportement donné. Ce formalisme s'avère insuffisant lorsque l'on décide de décrire des requis de sécurité. Il est nécessaire d'introduire un nouveau formalisme, celui des hyperpropriétés : une hyperpropriété est un ensemble de propriétés, autrement dit un ensemble d'ensembles de traces. Ainsi, en s'appuyant sur les notions de vivacité – on peut toujours rencontrer un bon comportement dans le futur – et de sûreté – il est possible d'atteindre un comportement rédhibitoire en un temps fini – déjà connues dans le domaine des propriétés de traces, Clarkson et Schneider définissent les notions d'hypervivacité et d'hypersûreté. Une hyperpropriété est alors l'intersection d'une hypervivacité et d'une hypersûreté, ce qui introduit un parallèle formel avec les propriétés de traces.

En plus de ces définitions, Clarkson et Schneider donnent le contre-exemple suivant à la problématique qui nous intéresse, selon laquelle le raffinement ne conserve pas nécessairement la sécurité. Considérons la propriété de sécurité "les valeurs possibles de sortie du système sont indépendantes de la valeur du bit secret h". Le système π qui affecte 0, 1 ou h de façon non-déterministe à la sortie, vérifie évidemment la propriété. En revanche, le système π' qui affecte systématiquement la valeur de h à la sortie ne vérifie pas la propriété. Pourtant, toute exécution de π' est une exécution de π, donc π' raffine π. On en déduit donc que raffiner un système peut mener à la création d'une faille de sécurité conformément à une propriété validée précédemment dans le processus de raffinement.

En parallèle de ces concepts, diverses propriétés de sécurité existent afin de décrire les différentes problématiques rencontrées en pratique.

La première définition formelle de la sécurité informatique approche le problème à travers les concepts de non-interférence et de flux d'information (Goguen et Meseguer, 1982). Un système est non-interférent si aucune de ses actions n'induit de changement dans l'observation d'un tiers extérieur. Dans le cas d'interférence, en revanche, on parle de flux ou de fuite d'information : suivant ses observations, le tiers peut déduire des informations potentiellement critiques sur l'état ou les paramètres du système. On peut ainsi formaliser une politique de sécurité en énonçant tout ce qui ne doit pas interférer par l'intermédiaire du système. L'attention est dans un premier temps porté sur les systèmes déterministes, modélisés par des machines de Mealy dont les entrées sont confidentielles et les sorties publiques. Dans le même temps, la non-interférence intransitive est définie : cette propriété autorise la déclassification de l'action confidentielle durant l'exécution, de sorte qu'après déclassification, l'action en question n'est plus critique (Rushby, 1992). Des généralisations aux systèmes non-déterministes sont développées par la suite (Sutherland, 1986). Mantel (2000, 2001) crée alors un moyen de définir toute propriété de non-interférence et de flux d'information à partir d'une liste de briques élémentaires : il introduit la notion de Prédicat Basique de Sécurité (BSP).

L'étape suivante de l'étude des propriétés de flux d'information est leur quantification (Smith, 2009). Le but est de déterminer quelle quantité d'information de l'entrée est détectée par observation de la sortie, et ce après une unique exécution. Pour cette première approche, on utilise le formalisme de l'entropie de Shannon (Shannon, 1948). On définit notamment les notions d'information mutuelle entre deux variables aléatoires – de façon intuitive, la quantité d'information partagée entre les deux variables – ou la min-entropie. Ces définitions établies impliquent l'étude de problèmes supplémentaires. Notamment, un système non-déterministe peut être déterminisé par un adversaire en garantissant un flux d'information borné par une constante donnée, ce en un temps exponentiel (Cerný *et al.*, 2011).

Dans le sillage des propriétés de flux d'information, on peut distinguer l'opacité. C'est une propriété particulière permettant de traduire diverses problématiques liées à la sécurité des systèmes, telles que l'anonymat (Lin, 2011). Elle est imaginée initialement par Mazaré (2004), dont la motivation est d'étendre la vérification de systèmes sécurisés pour des problèmes non-couverts par les propriétés existantes. Ainsi, l'auteur définit une propriété opaque dans un système s'il existe deux messages échangés par le système dont uniquement l'un des deux vérifie la propriété et tel qu'un observateur extérieur au système ne peut les différencier. Il applique ses définitions à des protocoles cryptographiques. Il introduit la notion de similitude de messages du point de vue de l'environnement, afin de traduire l'incapacité pour l'observateur extérieur de différencier les messages échangés dans leur intégralité.

Ces notions sont généralisées à l'ensemble des systèmes et utilisées dans le cadre des réseaux

de Petri (Bryans *et al.*, 2004, 2005). Enfin, cette même notion est formalisée dans le cadre des systèmes de transition (Bryans *et al.*, 2008). L'attaquant passif provenant de l'environnement extérieur est alors représenté par une fonction définie sur l'ensemble des exécutions du système de transition – la notion d'observateur Orwellien est notamment définie. L'article conclut sur la non-décidabilité de l'opacité sous observation orwellienne. Plus tard, Mullins et Yeddes (2014) se limitent à une certaine catégorie de fonctions d'observation, les projections Orwelliennes, et un certain type de secrets, les secrets réguliers, pour lesquelles l'opacité garde un intérêt pratique et devient vérifiable. Il est alors prouvé que l'opacité d'un secret régulier relativement à un projecteur Orwellien est équivalent à la non-interférence intransitive pour un système fini.

L'opacité se présente comme une inclusion d'ensembles réguliers. Bérard et Mullins (2014) ont alors l'idée de définir la notion de Propriété Rationnelle de Flux d'Information (RIFP), dont l'opacité est un exemple. Ce formalisme permet d'étendre les résultats connus pour cette notion à un nombre accru de nouvelles propriétés de sécurité. Le but sous-jacent est de proposer une nouvelle classification, dans laquelle les RIFP joueraient un rôle particulier.

L'aspect quantitatif de l'opacité permet de contrer sa valeur binaire jusqu'alors. Il devient possible de calculer différentes grandeurs d'opacité d'un secret régulier dans un PTS. On définit l'opacité libérale, qui est la probabilité que le secret ne soit pas opaque, et l'opacité restrictive, qui est la probabilité que le secret soit effectivement opaque (*cf.* chapitre 3 ; Sassolas, 2011; Bérard *et al.*, 2015c).

Toutes ces problématiques sont avant tout étudiées à travers des modèles probabilistes très précis, où l'abstraction est limitée, comme dans le cas des PTS. Bérard *et al.* (2015b) étudient alors à étendre l'opacité libérale dans le cadre formel décrit dans la section 2.1 et réutilisé dans la suite du mémoire. Notamment, considérons une IDTMC $\mathcal{S}$ et un secret φ.

La question de l'opacité de φ dans un modèle est la suivante : quelle est la probabilité pour un observateur extérieur de reconnaître avec certitude qu'une exécution du système dont il est témoin est secrète ? Ce problème est calculable dans un PTS, mais n'est pas réglé dans le cadre des spécifications IDTMC. Les auteurs définissent donc l'extension de ce problème en considérant que l'opacité d'un secret dans une spécification IDTMC est la valeur maximale de cette opacité calculée pour chacune des implémentations ordonnancées de la spécification. Autrement dit, ils désirent calculer le pire cas. Les auteurs prouvent que cette probabilité est mesurable pour un certain type de spécifications : les IDTMC non-modales, c'est-à-dire celles qui ne possèdent aucun arc modal. Il est alors prouvé que le calcul est réalisable en un temps doublement exponentiel. Cependant, la question reste ouverte pour le cas général : on se propose de la traiter dans la suite du mémoire.

La seconde question que pose l'article est la suivante : est-il possible de raffiner l'opacité, c'est-à-dire améliorer la mesure de l'opacité dans un modèle en le raffinant ? Les auteurs répondent en partie à cette question, en montrant que le raffinement fort préserve effectivement l'opacité. L'étude des raffinements faibles et complets reste ouverte, et on se propose de traiter le cas du raffinement faible dans la suite du mémoire.

CHAPITRE 3 PRÉLIMINAIRES

Dans ce chapitre, nous introduisons les définitions usuelles de modélisation de systèmes, afin de construire les fondations du formalisme choisi par la suite. Nous introduisons tout d'abord les notions générales de modélisation de systèmes probabilistes, puis nous présentons une propriété de sécurité particulière, l'opacité.

3.1 Modélisation

L'étape préliminaire à toute vérification formelle d'un système sécurisé est sa modélisation. Cette partie a pour but de définir les modèles usuels qui constituent les bases du cadre de notre étude.

3.1.1 Langages et automates

À partir d'un alphabet Σ, on peut construire des mots de longueur finie quelconque ou bien des mots de longueur infinie. On appelle *langage* un ensemble de mots issus d'un alphabet. On note Σ^k le langage contenant tous les mots de longueur k. On note Σ^* le langage contenant tous les mots de longueur finie – formellement, $\Sigma^* = \bigcup_{k\in\mathbb{N}} \Sigma^k$; le langage contenant tous les mots de longueur infinie est noté Σ^ω. L'union de ces deux langages est noté $\Sigma^\infty = \Sigma^\omega \cup \Sigma^*$.

Un premier outil mathématique lié à la théorie des langages est la notion d'automate (Bérard *et al.*, 2015b; Piterman, 2007).

Définition 1 (Automate)**.** *Un automate est un tuple*

$$\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_f)$$

tel que

- Q *est un ensemble fini d'états, dont* q_0 *qui est l'état initial ;*
- Σ *est un alphabet ;*
- $\delta : Q \times \Sigma \to Q$ *est une fonction de transition ;*
- Q_f *est un sous-ensemble de* Q*, composé des états dits acceptants.*

C'est une machine à états finis dont le but est de reconnaître si un mot appartient ou non à un langage donné, lié à l'automate. Pour cela, la machine lit le mot caractère par caractère. Chaque fois qu'elle lit un caractère, elle change d'état en concordance avec sa fonction de

transition. Cette procédure s'appelle l'*exécution* du mot par l'automate. Dans le cas d'un mot fini, on regarde dans quel état se trouve l'automate à la fin de la lecture : le mot appartient au langage si, et seulement si, l'automate à la fin de l'exécution est dans un état acceptant.

Cette méthode doit cependant être modifiée dans le cadre des mots infinis. Pour cela, l'automate généralement utilisé est l'automate de Büchi. La définition d'un automate de Büchi est exactement la définition 1, la différence étant dans la sémantique de ce modèle. Dans un automate de Büchi, un mot est reconnu si, et seulement si, durant sa lecture, l'automate passe infiniment souvent par des états acceptants. Ainsi, les automates de Büchi sont des extensions directes des automates sur les mots finis. Cependant, alors que dans le cas fini, les automates non-déterministes se déterminisent sans aucune perte d'information, cela n'est plus vrai pour les automates de Büchi (Büchi, 1962; Piterman, 2007). Afin de s'affranchir de cette difficulté, on fait appel à un autre type d'automate sur mot infini.

Définition 2 (Automate de Parité Déterministe (DPA))**.** *Un Automate de Parité Déterministe (DPA) est un tuple*

$$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$$

tel que

- Q *est un ensemble fini d'états, dont* q_0 *qui est l'état initial ;*
- Σ *est un alphabet ;*
- $\delta : Q \times \Sigma \to Q$ *est une fonction de transition ;*
- F *est une fonction qui associe à chaque état une couleur parmi un ensemble fini* $\{1, \ldots, k\}$.

Un mot accepté par l'automate est un mot de Σ^ω telle que, lors de la lecture, la couleur minimale des états rencontrés infiniment souvent est paire. Autrement dit, si on note ρ le mot considéré, et $Inf(\rho)$ l'ensemble des états rencontrés infiniment souvent durant la lecture, il reste à calculer $\min\{F(q) \,|\, q \in Inf(\rho)\}$.

On parle de langage *acceptant* ou *reconnu* par un DPA si l'ensemble des mots du langage constitue l'ensemble des mots reconnus par l'automate. Le langage est alors *ω-régulier*.

Exemple. *Sur l'exemple de la figure 3.1, les états sont notés $q|F(q)$, avec les notations de la définition précédente. En analysant cet automate, on note que toute lecture d'un mot contenant un b ou un c envoie l'automate dans l'état s_2 indéfiniment. Ainsi, si ρ est un mot qui n'appartient pas au langage $L = a^\omega$, c'est-à-dire si ρ contient un b ou un c, alors l'ensemble $Inf(\rho)$ est réduit à $\{s_2\}$, donc $\min\{F(q) \,|\, q \in Inf(\rho)\} = 3$ donc est impair : tout mot appartenant au complément du langage L n'est pas reconnu par le DPA. Inversement, la lecture de l'unique mot a^ω du langage L reste indéfiniment dans l'état s_1. Ainsi,* $\min\{F(q) \,|\, q \in$

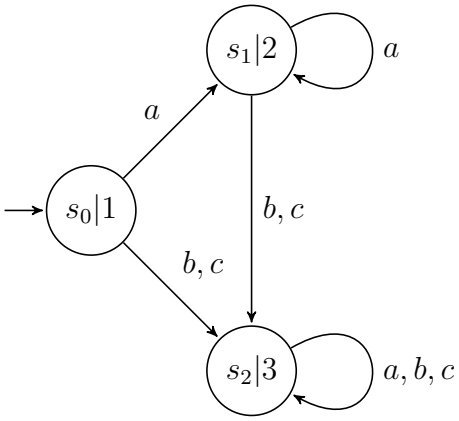


Figure 3.1 Exemple de DPA vérifiant le langage ω-régulier $L = a^\omega$, sur l'alphabet $\Sigma = \{a, b, c\}$

$Inf(a^\omega)\} = 2$ *donc est pair :* a^ω *est reconnu par le DPA. Ces deux affirmations permettent de conclure que le langage reconnu par le DPA est le langage* $L = a^\omega$*.*

3.1.2 Les systèmes de transitions

Alors que la partie précédente introduit davantage un outil mathématique, le but de cette partie est de définir les moyens possibles pour modéliser un système réel (Baier et Katoen, 2008).

La modélisation d'un système passe tout d'abord par l'abstraction de ses composantes inutiles pour l'étude – par exemple, lors de l'étude d'un protocole de contrôle d'accès, on suppose généralement que la cryptographie est parfaite et incassable. Ainsi, on ne s'intéresse qu'aux aspects fonctionnels intéressants pour l'étude. Un système est alors considéré suivant deux aspects particuliers : l'état de ses différentes propriétés atomiques, et son comportement futur. Ces deux aspects sont représentés dans un système d'états-transitions.

Définition 3 (Système de Transitions Étiquetées (LTS))**.** *Un Système de Transitions Étiquetées (LTS) est un tuple* $\mathcal{A} = (Q, q_0, T, \Sigma, \lambda)$ *tel que :*

- Q *est un ensemble dénombrable (fini ou non) d'états, avec* q_0 *l'état initial ;*
- $T \subseteq Q \times Q$ *est une relation de transition ;*
- Σ *est un alphabet ;*
- $\lambda : Q \to \Sigma$ *est une fonction d'étiquetage des états.*

Ce modèle est appelé à temps discret. À chaque unité de temps, le système réalise une transition issue de son état courant, ce qui met alors à jour son état. On appelle exécution du

modèle à partir d'un état $q \in Q$ la suite d'états $\rho = q_0 q_1 q_2 \ldots$, avec $q_0 = q$ et $\forall i$, $(q_i, q_{i+1}) \in T$ – autrement dit, la transition entre q_i et q_{i+1} existe. La *trace* de l'exécution ρ est alors la suite $tr(\rho) = \lambda(q_0)\lambda(q_1)\lambda(q_2)\cdots \in \Sigma^\infty$.

Cette représentation s'apparente à un arbre d'exécutions non-déterministe, en ce sens qu'elle liste toutes les exécutions possibles du système, sans considérer si l'une d'elles est plus ou moins probable.

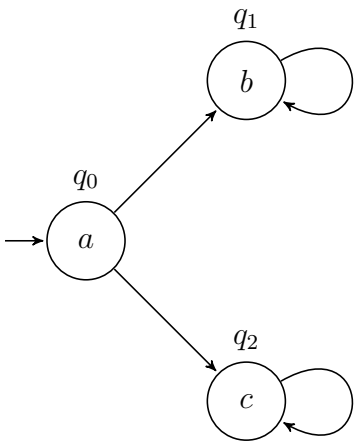


Figure 3.2 Exemple de LTS

Exemple. *L'exemple de la figure 3.2 modélise un système possédant deux exécutions infinies possibles :* $\rho_1 = q_0 q_1^\omega$ *ou* $\rho_2 = q_0 q_2^\omega$. *Les traces sont respectivement* $tr(\rho_1) = ab^\omega$ *et* $tr(\rho_2) = ac^\omega$.

Probabilisation L'aspect probabiliste a pour but d'affiner la modélisation du système en indiquant quelles exécutions sont les plus probables. De plus, ceci ajoute un outil de mesure. La probabilisation d'un LTS s'effectue en attribuant des probabilités aux scénarios futurs suivant le passé du système.

Formellement, prenons un LTS $\mathcal{A}$ dont les états sont indicés par q_i ($i \in \mathbb{N}$). Notons X_k la variable aléatoire qui désigne l'état du système à l'instant discret k ($k \in \mathbb{N}$). Probabiliser $\mathcal{A}$, c'est définir les probabilités $\mathbf{P}(X_{k+1} = q_i \mid \bigwedge_{j=0}^{j=k} X_j = q_{n_j})$, pour tout $k \in \mathbb{N}$ un instant, $i \in \mathbb{N}$ l'indice d'un état, et $\rho = q_{n_0} \ldots q_{n_k}$ une exécution.

Généralement, on considère que les systèmes probabilistes sont *sans-mémoire*, c'est-à-dire que la probabilité de rencontrer un certain état à l'instant $k+1$ ne dépend que de l'état du système à l'instant présent k. On parle également d'hypothèse *markovienne*. Formellement, définir les probabilités d'un système markovien, c'est définir les probabilités $\mathbf{P}(X_{k+1} = q_i \mid X_k = q_j)$,

pour tout instant k et tout couple d'états $q_i, q_j \in Q$; autrement dit, c'est exactement définir les probabilités des transitions entre les états q_i et q_j.

Définition 4 (Chaîne de Markov (MC))**.** *Une Chaîne de Markov (MC) est un tuple* $\mathcal{A} = (Q, q_0, \Delta)$ *tel que :*

- Q *est un ensemble dénombrable (fini ou non) d'états, avec* q_0 *l'état initial ;*
- $\Delta : Q \to \mathcal{D}ist(Q)$ *est une fonction qui associe à chaque état* $q \in Q$ *une distribution* $\Delta(q)$ *sur* Q*.*

Le formalisme des MC mis en relation avec nos réflexions sur les propriétés atomiques que l'on retrouve dans les LTS permet de définir une extension à ces deux notions.

Définition 5 (Système de Transitions Probabilistes (PTS))**.** *Un Système de Transitions Probabilistes (PTS) est un tuple* $\mathcal{A} = (Q, q_0, \Sigma, \Delta, L)$ *tel que :*

- Q *est un ensemble dénombrable (fini ou non) d'états, avec* q_0 *l'état initial ;*
- $\Delta : Q \to \mathcal{D}ist(Q)$ *est une fonction qui associe à chaque état* $q \in Q$ *une distribution* $\Delta(q)$ *sur* Q *;*
- Σ *est un alphabet ;*
- $L : Q \to \Sigma$ *est une fonction d'étiquetage des états.*

Ainsi, avec la notion de PTS, nous possédons le formalisme nécessaire pour modéliser des systèmes probabilistes. On considère qu'un système représenté par un PTS est entièrement spécifié – en oubliant les abstractions initiales.

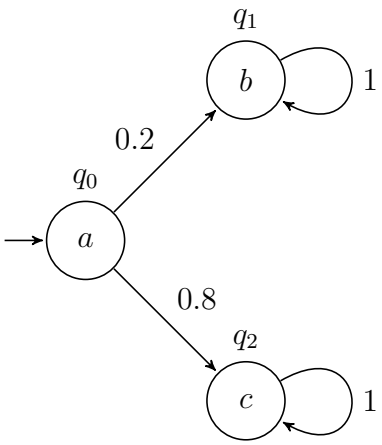


Figure 3.3 Exemple de PTS

Exemple. *L'exemple de la figure 3.3 modélise le même système que l'exemple de la figure 3.2, que l'on a probabilisé. Dans l'état initial, le système a désormais 80% de chance d'aller*

dans l'état q_2 ou bien 20% de chance d'aller dans l'état q_1. On note que cette figure représente bien un PTS puisque chaque état induit une distribution sur Q.

3.1.3 Un modèle de spécification

La section précédente introduit les outils nécessaires pour représenter complètement un système réel probabiliste. Cependant, une telle approche peut mener à une modélisation trop précise, qui ne laisse pas de place à l'ajustement. Par exemple, on peut vouloir ajuster les distributions dans certains états afin d'affiner le comportement ou le rapprocher du comportement désiré. Pour cela, on introduit des outils permettant une modélisation avec un niveau d'abstraction supplémentaire. On appelle ces nouveaux modèles des *spécifications*, car le but est de spécifier le champ des possibles pour le système que l'on veut étudier. Les PTS auxquels on aboutit après étude des spécifications sont des *implémentations* de celles-ci.

Une première idée à envisager lorsqu'il s'agit d'élargir le champ des possibles est de proposer plusieurs choix de distributions pour chaque état. Implémenter un tel objet consiste alors à donner un poids plus ou moins important à chaque distribution au choix.

Définition 6 (Processus de Décision Markovien (MDP))**.** *Un Processus de Décision Markovien (MDP) est un tuple $\mathcal{M} = (Q, q_0, A, \Delta, \Sigma, L)$ tel que :*

- *Q est un ensemble dénombrable (fini ou non) d'états, avec q_0 l'état initial ;*
- *A est un ensemble de choix de distributions sur Q – on l'appelle aussi l'ensemble des distributions de base ;*
- *$\Delta : Q \times A \to \mathcal{D}ist(Q)$ est une fonction qui associe à chaque état $q \in Q$ et à chaque choix $\mu \in A$ une distribution $\Delta(q, \mu)$ sur Q ;*
- *Σ est un alphabet ;*
- *$L : Q \to \Sigma$ est une fonction d'étiquetage des états.*

Un MDP est donc un PTS pour lequel on propose un choix entre plusieurs distributions dans chaque état. C'est un premier niveau d'abstraction qui permet de s'affranchir des probabilités fixes des PTS.

Exemple. *La figure 3.4 représente un modèle similaire à celui représenté par le PTS de la figure 3.3. La différence réside dans le fait que l'on a désormais le choix entre deux systèmes distincts, représentés par les deux distributions μ_1 et μ_2. L'implémentation consiste à choisir deux réels α et β, positifs et tels que leur somme vaut 1, afin de créer la distribution $\mu = \alpha \cdot \mu_1 + \beta \cdot \mu_2$. On note qu'en choisissant $\alpha = 1$ et $\beta = 0$, on obtient le PTS présenté figure 3.3.*

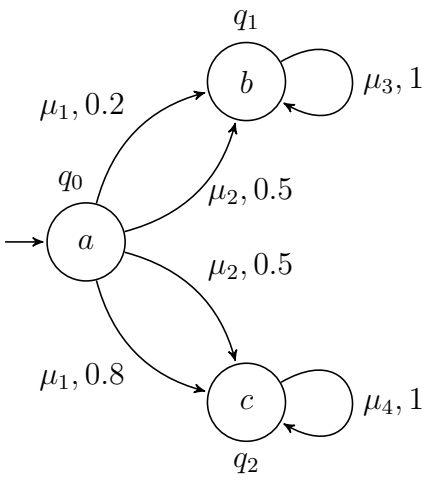


Figure 3.4 Exemple de MDP

Le chapitre 4 a pour but d'introduire un nouveau modèle de spécification, inspiré en partie des MDP, à partir duquel nous définirons plus précisément le concept de raffinement.

Avant cela, il est nécessaire d'introduire les notions de sécurité que nous utilisons pour décrire les systèmes dans cette étude.

3.2 Vérification de l'opacité

La formalisation de la section précédente permet la modélisation, entre autres, de systèmes sécurisés – autrement dit, on désire vérifier un certain nombre de propriétés de sécurité à l'étape de modélisation du système. L'idée est ainsi de détecter les potentielles failles de sécurité avant même leur implémentation. Dans cette étude, l'accent sécuritaire est porté par la propriété d'opacité.

Le scénario général que l'on étudie ici est le suivant. Soit un système $\mathcal{A}$ dont l'ensemble des propriétés atomiques constitue l'alphabet Σ. On considère un langage ω-régulier $\varphi \in \Sigma^\omega$ que l'on désire garder secret. On dit qu'une exécution du système satisfait le secret si sa trace appartient au langage φ. On définit également une fonction d'observation, $\mathcal{O} : \Sigma^\infty \to \Sigma_{ob}^\infty$, avec Σ_{ob} la partie observable de l'alphabet Σ. On se limite aux projections naturelles de Σ dans Σ_{ob}, généralisées aux langages ω-réguliers (Mullins et Yeddes, 2014; Bérard *et al.*, 2015b), c'est-à-dire les fonctions qui associent le mot vide à chaque élément non-observable de Σ, qui laissent inchangé tout élément observable de l'alphabet, et telles que $\mathcal{O}(\varepsilon) = \varepsilon$:

$$\begin{aligned} \forall \sigma \in \Sigma^\infty, \forall a \in \Sigma \setminus \Sigma_{ob}, \quad \mathcal{O}(\sigma a) &= \mathcal{O}(\sigma) \\ \forall \sigma \in \Sigma^\infty, \forall b \in \Sigma_{ob}, \qquad \mathcal{O}(\sigma b) &= \mathcal{O}(\sigma) b. \end{aligned}$$

On note $[\sigma]_{\mathcal{O}}$ – ou $[\sigma]$ quand l'observateur est sous-entendu – la classe d'observation du mot σ, c'est-à-dire l'ensemble des mots dont l'observation par $\mathcal{O}$ est la même que celle du mot σ :

$$[\sigma]_{\mathcal{O}} = \mathcal{O}^{-1}(\mathcal{O}(\sigma)).$$

On note *Obs* l'ensemble des classes d'observations de la fonction d'observation.

Notons que si L est un langage ω-régulier et si $\mathcal{O}$ est un observateur rationnel tel que présenté ci-dessus, alors le langage $\mathcal{O}(L)$ est un langage ω-régulier (Bérard *et al.*, 2015c).

Le but est de calculer la capacité pour le système $\mathcal{A}$ de cacher l'exécution du secret à l'observateur rationnel extérieur.

3.2.1 Une opacité binaire

La première définition d'opacité s'applique dans le contexte des LTS. Formellement, on dit d'un secret qu'il est opaque si, pour toute exécution du secret, il existe une exécution non-secrète qui est observée de la même manière par l'observateur rationnel extérieur – on dit que le secret est couvert par une exécution non-secrète. Autrement dit, pour que le secret soit opaque, il faut et il suffit que chaque classe d'observation contenant une exécution secrète contienne également au moins une exécution non-secrète. On en déduit la définition suivante.

Définition 7 (Opacité)**.** *Soient un LTS $\mathcal{A}$ dont le langage des exécutions est L, un secret ω-régulier $\varphi \subseteq L$ et un observateur rationnel $\mathcal{O}$. Le secret φ est* opaque dans $\mathcal{A}$ relativement à $\mathcal{O}$ *si, et seulement si,*

$$\mathcal{O}(\varphi) \subseteq \mathcal{O}(L \setminus \varphi).$$

Notons que l'on peut également définir l'opacité symétrique, en affirmant qu'un secret φ est symétriquement opaque dans un système au langage L relativement à un observateur si, et seulement si, φ et $L \setminus \varphi$ sont opaques dans le système relativement à l'observateur. Autrement dit, il faut et il suffit que chaque classe d'observation possède à la fois des exécutions secrètes et non-secrètes.

Cette première définition d'opacité induit une classification simple de problèmes de sécurité, puisque l'opacité ici est une grandeur binaire : soit le secret est opaque, soit il ne l'est pas. Cependant, dire que le secret n'est pas opaque signifie qu'il existe une exécution appartenant à l'ensemble φ telle qu'aucune exécution non-secrète ne possède la même observation. Par conséquent, cela ne signifie pas que le secret est systématiquement trahi. Cela signifie uniquement que, si l'exécution en question est observée, alors l'observateur a pleine connaissance du fait qu'il est en présence d'un secret. La question naturelle que l'on peut se poser alors

est la suivante : quelle est la probabilité que cette exécution soit réalisée par le système ?

3.2.2 L'opacité probabiliste

L'opacité probabiliste libérale

Afin de répondre à cette question, il faut pouvoir inclure l'aspect probabiliste au modèle. Pour cela, on adapte la notion d'opacité au contexte des PTS.

On considère l'ensemble des exécutions du système pour lesquelles un observateur est certain d'être en présence du secret. Cet ensemble est donc l'ensemble des exécutions du secret qui n'appartiennent à aucune classe d'observation d'une exécution non-secrète : autrement dit, avec les notations précédentes, on considère l'ensemble $\mathcal{V}(\mathcal{A}, \varphi, \mathcal{O}) = \varphi \cap \overline{\mathcal{O}^{-1}(\mathcal{O}(L \setminus \varphi))}$ (Bérard *et al.*, 2015c; Sassolas, 2011).

Définition 8 (Opacité libérale)**.** *Soient un PTS $\mathcal{A}$ dont le langage des exécutions est L, un secret ω-régulier $\varphi \subseteq L$ et un observateur rationnel $\mathcal{O}$. L'*opacité libérale de φ dans $\mathcal{A}$ relativement à $\mathcal{O}$ *est la grandeur :*

$$PO_l(\mathcal{A}, \varphi, \mathcal{O}) = \mathbf{P}\Big(\varphi \cap \overline{\mathcal{O}^{-1}(\mathcal{O}(L \setminus \varphi))}\Big).$$

Cette définition probabiliste nuance la définition binaire d'opacité, en ce sens qu'elle permet de classer différents problèmes de sécurité suivant la valeur obtenue.

Proposition 1. *Soient un PTS $\mathcal{A}$ dont le langage des exécutions est L, un secret ω-régulier $\varphi \subseteq L$ et un observateur rationnel $\mathcal{O}$.*

- $0 \leq PO_l(\mathcal{A}, \varphi, \mathcal{O}) \leq 1$ *;*
- $PO_l(\mathcal{A}, \varphi, \mathcal{O}) = 1$ *si, et seulement si, $\varphi = L$;*
- $PO_l(\mathcal{A}, \varphi, \mathcal{O}) = 0$ *si, et seulement si, φ est opaque dans $\mathcal{A}$ relativement à $\mathcal{O}$.*

Démonstration. Le premier point découle du fait que l'opacité libérale est par définition une probabilité.

Pour montrer le deuxième point, remarquons que si X est un ensemble, $\mathbf{P}(X) = 1$ si, et seulement si, $X = U$, avec U l'univers. Par conséquent,

$$\begin{aligned}
PO_l(\mathcal{A}, \varphi, \mathcal{O}) = 1 &\Leftrightarrow \mathcal{V}(\mathcal{A}, \varphi, \mathcal{O}) = L \\
&\Leftrightarrow \varphi \cap \overline{\mathcal{O}^{-1}(\mathcal{O}(L \setminus \varphi))} = L \\
&\Leftrightarrow L \subseteq \varphi \ \wedge \ L \subseteq \overline{\mathcal{O}^{-1}(\mathcal{O}(L \setminus \varphi))} \\
&\Leftrightarrow L = \varphi \quad (\text{car } \varphi \subseteq L)
\end{aligned}$$

ce qui prouve le deuxième point.

Pour montrer le troisième point, remarquons que si X est un ensemble, $\mathbf{P}(X) = 0$ si, et seulement si, $X = \emptyset$. Par conséquent,

$$\begin{array}{rcl} PO_l(\mathcal{A}, \varphi, \mathcal{O}) = 0 & \Leftrightarrow & \mathcal{V}(\mathcal{A}, \varphi, \mathcal{O}) = \emptyset \\ & \Leftrightarrow & \mathcal{O}(\varphi) \cap \overline{\mathcal{O}(L \setminus \varphi)} = \emptyset \\ & \Leftrightarrow & \mathcal{O}(\varphi) \subseteq \mathcal{O}(L \setminus \varphi) \end{array}$$

ce qui prouve la propriété. □

Cette valeur permet de discriminer les systèmes pour lequel les secrets sont non-opaques entre eux. On peut alors affirmer quels sont les systèmes les moins sécurisés, c'est-à-dire ceux pour lesquels la probabilité de trahir tout le secret est la plus grande.

Notons qu'à l'inverse des propriétés de flux d'information fondées sur la notion d'entropie de Shannon (Smith, 2009), on ne calcule pas la proportion du secret qui n'est plus protégée, mais bien la probabilité que tout le secret soit rendu public.

Théorème 1. *Soient un PTS $\mathcal{A}$ dont le langage des exécutions est L, un secret ω-régulier $\varphi \subseteq L$ et un observateur rationnel $\mathcal{O}$. Alors la grandeur $PO_l(\mathcal{A}, \varphi, \mathcal{O})$ est mesurable.*

Démonstration. D'après la proposition 9 appliquée au cas particulier d'un PTS, le langage L est ω-régulier. De même, φ est ω-régulier. Enfin, d'après la remarque sur les observateurs rationnels, et d'après les propriétés de fermeture des langages ω-réguliers, on en déduit que le langage dont on veut connaître la probabilité est ω-régulier. Ainsi, le problème de calcul d'opacité libérale revient à un problème de calcul de probabilité d'un langage ω-régulier dans un PTS : on sait que ce problème est décidable. □

D'autres formes d'opacité

Opacité restrictive À l'instar de l'opacité libérale qui permet de discriminer les systèmes pour lesquels les secrets sont non-opaques, on peut définir une notion duale afin de discriminer les systèmes pour lesquels les secrets sont opaques.

Intuitivement, le fait que le secret soit opaque ne le rend pas totalement immune à toute fuite. En effet, l'observation donne un certain nombre d'informations à l'attaquant extérieur. Celui-ci connaît la répartition du secret sur les classes : ainsi, s'il observe une classe couverte en grande majorité par des exécutions secrètes, sa probabilité de pouvoir affirmer qu'il est en présence du secret est plus importante que s'il est en présence d'une classe majoritairement couverte par des exécutions non-critiques (Bérard *et al.*, 2015c; Sassolas, 2011).

Définition 9 (Opacité restrictive)**.** *Soient un PTS $\mathcal{A}$ dont le langage des exécutions est L, un secret ω-régulier $\varphi \subseteq L$ et un observateur rationnel $\mathcal{O}$. L'*opacité restrictive de φ dans $\mathcal{A}$ relativement à $\mathcal{O}$ *est la grandeur définie par :*

$$\frac{1}{PO_r(\mathcal{A}, \varphi, \mathcal{O})} = \sum_{o \in Obs} \mathbf{P}(o) \cdot \frac{1}{\mathbf{P}(L \setminus \varphi \,|\, o)}.$$

On utilise la moyenne harmonique pondérée par les probabilités sur les classes d'observation afin de donner davantage de poids aux classes qui ont le plus de chance de trahir le secret. De plus, cette définition vérifie les résultats de la proposition 2, qui sont les résultats naturels d'une définition d'opacité restrictive.

Proposition 2. *Soient un PTS $\mathcal{A}$ dont le langage des exécutions est L, un secret ω-régulier $\varphi \subseteq L$ et un observateur rationnel $\mathcal{O}$.*

- $0 \leq PO_r(\mathcal{A}, \varphi, \mathcal{O}) \leq 1$ *;*
- $PO_r(\mathcal{A}, \varphi, \mathcal{O}) = 0$ *si, et seulement si, φ n'est pas opaque, c'est-à-dire qu'il existe une classe d'observation qui est uniquement composée d'exécutions de φ ;*
- $PO_r(\mathcal{A}, \varphi, \mathcal{O}) = 1$ *si, et seulement si, $\varphi = \emptyset$.*

Démonstration. Le premier point se déduit immédiatement du fait que l'opacité restrictive est par définition une moyenne harmonique de plusieurs probabilités, c'est donc une probabilité également.

Pour montrer le second point, remarquons que φ n'est pas opaque si, et seulement si, il existe une classe d'observation $o \in Obs$ composée uniquement d'exécutions de φ, *i.e.* telle que $\mathbf{P}(L \setminus \varphi \,|\, o) = 0$. Cela est équivalent au fait que $\frac{1}{PO_r(\mathcal{A},\varphi,\mathcal{O})}$ tend vers l'infini, car alors au moins un des termes de la somme tend vers l'infini. Or, l'inverse de l'opacité tend vers l'infini si, et seulement si, l'opacité restrictive tend vers 0, ce qui termine la preuve.

Montrons le dernier point.

$$\begin{array}{rcll} PO_r(\mathcal{A}, \varphi, \mathcal{O}) = 1 & \Leftrightarrow & \frac{1}{PO_r(\mathcal{A},\varphi,\mathcal{O})} = 1 \\ & \Leftrightarrow & \forall o \in Obs, \ \mathbf{P}(L \setminus \varphi \,|\, o) = \mathbf{P}(o) \\ & \Leftrightarrow & \mathbf{P}(L \setminus \varphi) = 1 \\ & \Leftrightarrow & \varphi = \emptyset. \end{array}$$

□

Un modèle alternatif On présente dans la suite de cette section une mesure d'opacité alternative (Saboori et Hadjicostis, 2014). Dans leurs travaux, les auteurs considèrent des

systèmes finis – dont les langages sont inclus dans Σ^*. On propose de transférer leurs notions dans la sémantique des langages de Σ^ω. Pour cela, on définit la restriction suivante, inspirée des notations de l'article en question.

Définition 10. *Soit L un langage de Σ^*.*

- *L'extension de L à Σ^ω est le cône noté $L_\omega = L\Sigma^\omega$;*
- *Si σ est un mot de L, on appelle* longueur utile *du cône résultant σ_ω la longueur de la chaîne σ ; σ est alors appelé* partie utile.
- *Si $\mathcal{A}$ est un PTS dont le langage des exécutions est L, on construit $\mathcal{A}_\omega$ le PTS dont le langage des exécutions est L_ω.*

L'objet de cette partie concerne donc un ensemble très restreint de PTS.

Considérons que l'attaquant n'est capable d'observer l'exécution que durant un temps fini, qu'il peut choisir aussi long qu'il le désire. Ainsi, il n'est capable d'observer que les exécutions dont la longueur utile est fixée à un entier k, correspondant au temps d'observation. Pour vérifier qu'un système est bien sécurisé contre ce type d'observation, il est alors nécessaire de vérifier que c'est le cas pour chaque longueur potentiellement choisie par l'attaquant. En substance, on introduit ici la notion de quasi-opacité : on autorise le fait que le secret ne soit pas opaque, mais on veut s'assurer que l'opacité libérale ne dépasse jamais un seuil critique, noté θ.

Définition 11 (Quasi-opacité uniforme)**.** *Soit L un langage de Σ^*. Soient un PTS $\mathcal{A}$ dont le langage des exécutions est L, un secret régulier $\varphi \subseteq L$ et un observateur rationnel $\mathcal{O}$.*

Notons $L_C = \varphi \setminus \mathcal{O}^{-1}(\mathcal{O}(L \setminus \varphi))$, le langage de Σ^* composé des exécutions de $\mathcal{A}$ qui brisent l'opacité de φ.

Alors, pour un $\theta > 0$, le cône du secret φ_ω est uniformément quasi-opaque, ou θ-opaque, dans $\mathcal{A}_\omega$ relativement à $\mathcal{O}$ *si, et seulement si,*

$$\forall k \in \mathbb{N},\ \mathbf{P}\Big((L_C \cap \Sigma^k)_\omega\Big) < \theta.$$

Il est important de noter que le langage L_C est construit à partir de la définition de l'opacité dans la sémantique des langages réguliers sur Σ^* (Mullins et Yeddes, 2014). Formellement, les notions sont analogues, et L_C est exactement le langage $\mathcal{V}(\mathcal{A}, \varphi, \mathcal{O})$ dont on calcule la probabilité dans le cadre de l'opacité libérale. La notion de quasi-opacité uniforme revient donc à dire que l'opacité libérale du secret est uniformément répartie suivant la longueur utile des mots du langage $(L_C)_\omega$.

En résumé Ce chapitre a permis d'introduire les notions préliminaires nécessaires à la construction des théorèmes et contributions du chapitre suivant. Nous avons introduit des notions liées principalement au concept d'implémentation : les systèmes décrits par les PTS sont des systèmes très précis, notamment au niveau des probabilités de transitions. Le but du chapitre suivant est de s'affranchir de cette précision, afin de pouvoir utiliser le concept de raffinement de systèmes. Il est alors nécessaire d'analyser comment se comportent les propriétés de sécurité énoncées dans le cadre plus général des spécifications de systèmes.

CHAPITRE 4 THÉORIE DES SPÉCIFICATIONS

Ce chapitre constitue la présentation du cadre théorique de l'étude. Nous commençons par définir le modèle de spécification utilisé, l'IDTMC, dans la section 4.1. À partir de ce modèle de spécification, nous définissons deux types de raffinements – faible et fort – dans la section 4.2, que l'on compare. Cela permet d'aboutir, dans la section 4.3, à la notion d'implémentation d'une spécification, qui est un PTS qui raffine l'IDTMC, puis à la notion de raffinement complet, nouveau raffinement plus faible encore que le raffinement faible. Quelques propriétés sur les langages dans les spécifications sont énoncées dans la section 4.4, afin de compléter les définitions nécessaires au cadre théorique. La section 4.5 définit l'ordonnancement, qui est un moyen de construire une implémentation à partir d'une spécification. Cette méthode permet de construire les implémentations qu'un adversaire est capable de créer, ce qui justifie l'étude de ces implémentations en particulier. La section 4.6 conclut ce chapitre en introduisant une extension de l'opacité aux spécifications IDTMC. La mesure est prise dans le pire cas sur l'ensemble des implémentations ordonnancées, car ce sont celles qui sont implémentées par un adversaire malveillant.

4.1 IDTMC

La première brique du cadre théorique formé ici consiste en le choix d'un modèle de spécification. Un tel modèle doit être probabiliste et non-déterministe, afin de jouer le rôle de spécification en représentant l'ensemble des implémentations possibles du système étudié. Pour cela, on décide de s'inspirer du modèle des PTS, déjà utilisé pour modéliser habituellement des systèmes complètement déterminés. On utilise donc la notion d'IDTMC, extension du PTS.

Dans la suite, on note $\mathcal{I}$ l'ensemble des intervalles de $[0, 1]$.

Définition 12 (Chaîne de Markov discrète à Intervalles (IDTMC))**.** *Une Chaîne de Markov discrète à Intervalles (IDTMC) est un tuple $\mathcal{S} = (S, s_0, T, \Sigma, \lambda)$ tel que :*

- *S est un ensemble dénombrable (fini ou non) d'états, avec s_0 l'état initial ;*
- *$T : S \to (S \to \mathcal{I})$ associe à chaque état $s \in S$ une fonction $T(s)$ de S dans $\mathcal{I}$, telle que*
$$\exists f : S \to [0, 1] \ : \ \sum_{s' \in S} f(s') = 1 \ \textit{et} \ \forall s' \in S, \ f(s') \in T(s)(s');$$
- *Σ est un alphabet ;*
- *$\lambda : S \to \Sigma$ est une fonction d'étiquetage des états.*

Ici, le choix de la distribution dans chaque état s s'effectue grâce à la fonction $T(s)$. Puisque l'on considère des intervalles de $[0, 1]$, l'ensemble des choix est indénombrable. La condition sur la fonction $T(s)$ permet d'assurer que l'on peut définir une distribution à partir de s. Une telle distribution est une fonction $f : S \to [0, 1]$ telle que $\forall s' \in S$, $f(s') \in T(s)(s')$ et $\sum_{s' \in S} f(s') = 1$. Par la suite, on notera $f \in T(s)$ une telle distribution.

Il est à noter qu'à l'origine, les IDTMC sont des extensions des MC (Jonsson et Larsen, 1991). Notre représentation ajoute la fonction d'étiquetage, afin de se rapprocher des PTS.

On peut distinguer les IDTMC suivant si elles sont *modales* ou non. Une IDTMC modale est une spécification qui possède au moins une transition modale, c'est-à-dire une transition reliant deux états s et s' telle qu'il existe une distribution depuis l'état s qui annule sa probabilité.

La sémantique d'exécution d'une IDTMC est héritée de celle d'un LTS, décrite dans le chapitre 3.

Exemple. *L'exemple de la figure 4.1 reprend à nouveau l'exemple de la figure 3.3, en proposant une spécification plus large concernant les probabilités des transitions. On note que cette IDTMC est modale car la transition entre les états q_0 et q_1 peut être annulée. Dans l'implémentation résultante, l'état q_1 n'est plus accessible.*

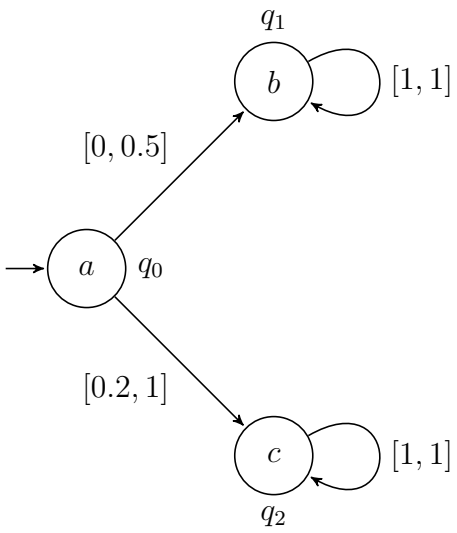


Figure 4.1 Exemple d'IDTMC

4.2 Raffinement fort et faible

On peut définir le raffinement dans le cadre formel des spécifications IDTMC, de trois manières différentes : on parle de raffinement *fort*, *faible*, et *complet*. Nous définissons dans cette section les deux premières notions, le raffinement complet étant défini dans la section 4.3.

Définition 13 (Raffinement fort)**.** *Soient deux spécifications* $\mathcal{S}_1 = (S_1, s_{1,0}, T_1, \Sigma, \lambda_1)$ *et* $\mathcal{S}_2 = (S_2, s_{2,0}, T_2, \Sigma, \lambda_2)$*.*

$\mathcal{S}_1$ raffine fortement $\mathcal{S}_2$ *(on note* $\mathcal{S}_1 \preceq_F \mathcal{S}_2$*) si, et seulement si, il existe une relation* $\mathcal{R} \subseteq S_1 \times S_2$ *telle que*

- $s_{1,0} \mathcal{R} s_{2,0}$ *;*
- *si* $s_1 \mathcal{R} s_2$*, alors* $\lambda_1(s_1) = \lambda_2(s_2)$ *et* $\exists\, \delta : S_1 \to Dist(S_2) : \forall\, f \in T_1(s_1)$*,*

$$\forall\, s'_2 \in S_2,\ \left(\sum_{s'_1 \in S_1} f(s'_1) \cdot \delta(s'_1)(s'_2) \right) \in T_2(s_2)(s'_2)$$
$$\text{et si } \delta(s'_1)(s'_2) > 0,\ s'_1 \mathcal{R} s'_2.$$

Définition 14 (Raffinement faible)**.** *Soient deux spécifications* $\mathcal{S}_1 = (S_1, s_{1,0}, T_1, \Sigma, \lambda_1)$ *et* $\mathcal{S}_2 = (S_2, s_{2,0}, T_2, \Sigma, \lambda_2)$*.*

$\mathcal{S}_1$ raffine faiblement $\mathcal{S}_2$ *(on note* $\mathcal{S}_1 \preceq_f \mathcal{S}_2$*) si, et seulement si, il existe une relation* $\mathcal{R} \subseteq S_1 \times S_2$ *tel que :*

- $s_{1,0} \mathcal{R} s_{2,0}$ *;*
- *si* $s_1 \mathcal{R} s_2$*, alors* $\lambda_1(s_1) = \lambda_2(s_2)$ *et* $\forall\, f \in T_1(s_1)$*,* $\exists\, \delta : S_1 \to Dist(S_2)$ *:*

$$\forall\, s'_2 \in S_2,\ \left(\sum_{s'_1 \in S_1} f(s'_1) \cdot \delta(s'_1)(s'_2) \right) \in T_2(s_2)(s'_2)$$
$$\text{et si } \delta(s'_1)(s'_2) > 0,\ s'_1 \mathcal{R} s'_2.$$

Ces deux définitions de raffinement sont très proches l'une de l'autre. Formellement, la différence réside dans la nature de la fonction δ de correspondance des distributions. Dans le cadre du raffinement fort, celle-ci doit être uniforme : une seule fonction doit suffire pour assurer la correspondance des intervalles de probabilité. En revanche, δ n'est pas nécessairement uniforme dans le cadre du raffinement faible et peut dépendre du choix des distributions.

Cette remarque prouve le résultat suivant.

Proposition 3. *Si* $\mathcal{S}_1$ *raffine fortement* $\mathcal{S}_2$*, alors* $\mathcal{S}_1$ *raffine faiblement* $\mathcal{S}_2$*.*

Remarque. *En revanche, la réciproque est fausse : pour le prouver, étudions le contre-exemple tiré de Delahaye* et al. *(2011) et représenté sur la figure 4.2. Il y a bien raffinement*

faible de $\mathcal{S}_2$ par $\mathcal{S}_1$. En effet, soit $f \in T_1(q)$. Notons $x = f(s)$. Posons alors $\delta : S_1 \to Dist(S_2)$ telle que $\delta(s)(s_1) = p \cdot x$ et $\delta(s)(s_2) = (1-p) \cdot x$, avec

$$\begin{array}{ll} p = 0 & si\ x \in [0.2, 0.4] \\ p = \frac{x-0.3}{x} & si\ x \in [0.4, 0.6] \\ p = 0.6 & sinon. \end{array}$$

On obtient donc une fonction δ non-uniforme suivant la distribution choisie, aucune fonction uniforme ne fonctionnant : cela démontre que le raffinement fort est impossible, au contraire du raffinement faible.

(a) $\mathcal{S}_1$ (b) $\mathcal{S}_2$

Figure 4.2 $\mathcal{S}_1$ raffine faiblement mais non fortement $\mathcal{S}_2$

4.3 Implémentation et raffinement complet

Comme énoncé précédemment, les PTS sont des implémentations de spécifications IDTMC. Pour définir cette notion de façon plus formelle, remarquons qu'un PTS n'est rien de moins qu'une IDTMC pour laquelle les intervalles de probabilité sont réduits à des singletons. Autrement dit, pour chaque paire d'états (q, q') du PTS, on transforme la probabilité $\Delta(q)(q')$ en son singleton correspondant $T(q)(q') = \{\Delta(q)(q')\}$. De cette manière, on peut étendre directement la notion de raffinement au formalisme des PTS.

Définition 15 (Implémentation)**.** *Soient $\mathcal{A}$ un PTS et $\mathcal{S}$ une spécification IDTMC. On dit que $\mathcal{A}$* implémente *$\mathcal{S}$ si, et seulement si, $\mathcal{A}$ raffine faiblement (fortement) $\mathcal{S}$.*

Si on note $\mathfrak{P}$ l'ensemble des PTS, et $imp(\mathcal{S})$ l'ensemble des implémentations d'une IDTMC $\mathcal{S}$,

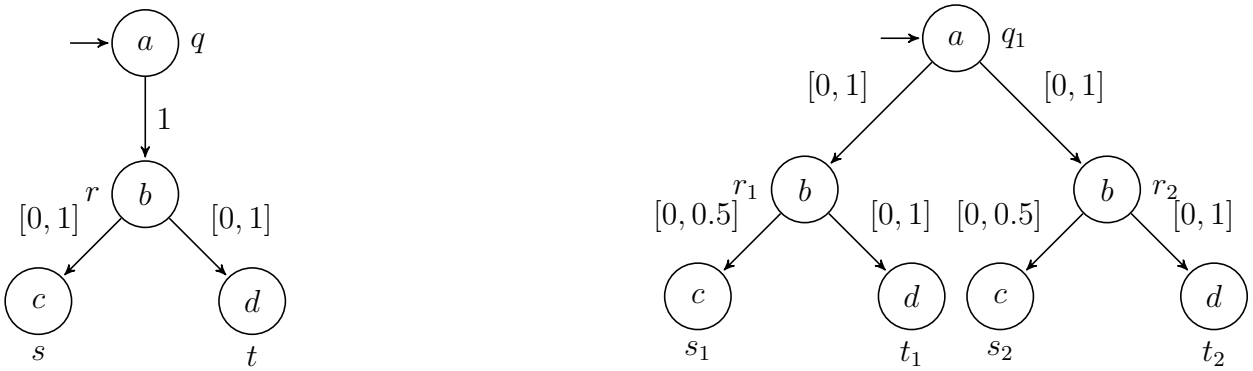
$$imp(\mathcal{S}) = \{\mathcal{A} \in \mathfrak{P} : \mathcal{A} \preceq_i \mathcal{S}\} \text{ où } \preceq_i \in \{\preceq_f, \preceq_F\}.$$

La relation d'implémentation entre un PTS et sa spécification IDTMC est donc une simple relation de raffinement.

En parallèle de cette définition formelle d'implémentation, on peut remarquer que le raffinement est une relation transitive sur l'ensemble des spécifications. Par conséquent, si une IDTMC $\mathcal{S}_1$ raffine (fortement ou faiblement) une IDTMC $\mathcal{S}_2$, cela implique que tout élément de $imp(\mathcal{S}_1)$ raffine $\mathcal{S}_2$. Ainsi, tout élément de $imp(\mathcal{S}_1)$ est un élément de $imp(\mathcal{S}_2)$. On déduit la proposition suivante.

Proposition 4. *Soient deux IDTMC $\mathcal{S}_1$ et $\mathcal{S}_2$ telles que $\mathcal{S}_1$ raffine faiblement ou fortement $\mathcal{S}_2$. Alors $imp(\mathcal{S}_1) \subseteq imp(\mathcal{S}_2)$.*

Remarque. *La réciproque est cependant fausse : pour le prouver, citons ce contre-exemple tiré de Delahaye* et al. *(2011) et représenté figure 4.3. En effet, $imp(\mathcal{S}_1) \subseteq imp(\mathcal{S}_2)$ mais il n'y a pas de raffinement faible car l'état r ne raffine ni r_1 ni r_2. Par l'absurde, supposons qu'il existe une relation de raffinement faible $\mathcal{R}$ telle que, par exemple, $r\mathcal{R}r_1$. Posons $f \in T_1(r)$ telle que $f(s) = 1$. Soit $\delta : S_1 \to Dist(S_2)$ vérifiant la définition du raffinement faible. Puisque s a pour étiquette c, seuls s_1 et s_2 peuvent être raffinés par s. Donc seuls $\delta(s)(s_1)$ et $\delta(s)(s_2)$ sont éventuellement non-nuls. Ainsi, par exemple, $f(s) \cdot \delta(s)(s_1) \in T_2(r_1)(s_1)$. Or $T_2(r_1)(s_1) = [0, 0.5]$ et $f(s) = 1$: il y a contradiction.*



(a) $\mathcal{S}_1$ (b) $\mathcal{S}_2$

Figure 4.3 $imp(\mathcal{S}_1) \subseteq imp(\mathcal{S}_2)$ mais $\mathcal{S}_1$ ne raffine pas faiblement $\mathcal{S}_2$

Ce résultat permet de définir un raffinement plus complet, lui-même plus faible que les deux notions de raffinement précédemment introduites.

Définition 16 (Raffinement complet)**.** *Soient deux IDTMC $\mathcal{S}_1$ et $\mathcal{S}_2$.*

$\mathcal{S}_1$ raffine complètement $\mathcal{S}_2$ *(on note $\mathcal{S}_1 \preceq_c \mathcal{S}_2$) si, et seulement si, $imp(\mathcal{S}_1) \subseteq imp(\mathcal{S}_2)$.*

Finalement, on a la propriété suivante, qui résume les propositions 3 et 4.

Proposition 5. *Soient deux spécifications IDTMC $\mathcal{S}_1$ et $\mathcal{S}_2$.*

Si $\mathcal{S}_1$ raffine fortement $\mathcal{S}_2$, alors elle la raffine faiblement.

Si $\mathcal{S}_1$ raffine faiblement $\mathcal{S}_2$, alors elle la raffine complètement.

Notons que le raffinement d'une IDTMC par un PTS est un cas particulier, résumé par la proposition suivante.

Proposition 6 (Raffinement par un PTS)**.** *Soient $\mathcal{A}$ un PTS et $\mathcal{S}$ une IDTMC quelconque. Les énoncés suivants sont équivalents.*

1. *$\mathcal{A}$ raffine fortement $\mathcal{S}$;*
2. *$\mathcal{A}$ raffine faiblement $\mathcal{S}$;*
3. *$\mathcal{A}$ raffine complètement $\mathcal{S}$;*
4. *$\mathcal{A}$ implémente $\mathcal{S}$.*

Démonstration. On sait déjà que (1) implique (2) et que (2) implique (3) pour tout couple d'IDTMC d'après la proposition 5 : puisque tous les PTS sont des IDTMC particulières, la proposition tient toujours.

De plus, on sait par définition de l'implémentation que (1) est équivalent à (4).

Montrons que (2) implique (1). Si $\mathcal{A} = (Q, q_0, \Sigma, \Delta, L)$ raffine faiblement $\mathcal{S} = (S, s_0, T, \Sigma, \lambda)$, alors, avec les notations habituelles :

$$\forall\, f \in \{\Delta(q)\},\, \exists\, \delta : Q \to Dist(S)\ :\ \forall\, s' \in S,\, \left(\sum_{q' \in Q} f(q') \cdot \delta(q')(s') \right) \in T(s)(s').$$

Or, $\{\Delta(q)\}$ est déjà réduit à un singleton, donc la proposition se réduit à :

$$\exists\, \delta : Q \to Dist(S)\ :\ \forall\, s' \in S,\, \left(\sum_{q' \in Q} \Delta(q)(q') \cdot \delta(q')(s') \right) \in T(s)(s').$$

Cet énoncé est exactement la traduction du raffinement fort d'une IDTMC par un PTS, obtenue en réalisant le même raisonnement : on en déduit que $\mathcal{A}$ raffine fortement $\mathcal{S}$, ce qui prouve l'implication.

Il reste à montrer que (3) implique (1) ou (2). Supposons donc que $\mathcal{A}$ raffine complètement $\mathcal{S}$, c'est-à-dire que $imp(\mathcal{A}) \subseteq \mathcal{S}$. Par définition, $imp(\mathcal{A})$ décrit l'ensemble des PTS qui raffinent $\mathcal{A}$. Notamment, $\mathcal{A}$ lui-même est un PTS qui raffine $\mathcal{A}$. Donc, $\mathcal{A} \in imp(\mathcal{A})$, et par conséquent, $\mathcal{A} \in imp(\mathcal{S})$. Donc par définition, $\mathcal{A}$ raffine faiblement ou fortement $\mathcal{S}$, ce qui prouve l'implication. □

Cette proposition permet donc d'éluder le type de raffinement considéré lorsque c'est un PTS qui raffine un IDTMC : dans un tel cas, on notera alors $\mathcal{A} \preceq \mathcal{S}$ et on dira de manière équivalent $\mathcal{A}$ raffine ou implémente $\mathcal{S}$.

De plus, d'après Baier *et al.* (2005), on a la proposition suivante.

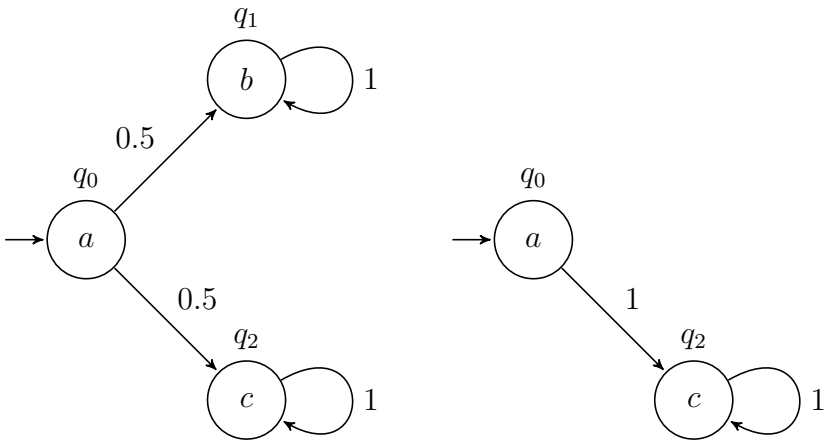
Proposition 7. *Soient $\mathcal{A}_1$ et $\mathcal{A}_2$ deux PTS.*

$$\mathcal{A}_1 \preceq \mathcal{A}_2 \Leftrightarrow \mathcal{A}_2 \preceq \mathcal{A}_1.$$

4.4 Langages dans les IDTMC

Ces définitions énoncées, on peut donner quelques résultats sur les langages dans les spécifications.

Le formalisme des IDTMC modales apporte une complexité dans la détermination d'un unique langage, qui serait commun à l'ensemble des implémentations de l'IDTMC. Pour illustrer ceci, implémentons par exemple de deux manières l'IDTMC de la figure 4.1, et représentons les PTS obtenus sur la figure 4.4. On note que le langage de la première implémentation est $L_1 = a(b^\omega + c^\omega)$ alors que le langage de la seconde implémentation est $L_2 = ac^\omega$. La notion de langage d'IDTMC pose donc problème.



(a) Première implémentation (b) Seconde implémentation

Figure 4.4 Deux implémentations de l'exemple de la figure 4.1

Cependant, on peut affirmer la proposition suivante.

Proposition 8. *Soient $\mathcal{S}_1$ et $\mathcal{S}_2$ deux IDTMC non-modaux telles que $\mathcal{S}_1 \preceq_f \mathcal{S}_2$.*

Alors, en notant L_1 (respectivement L_2) le langage de $\mathcal{S}_1$ (respectivement $\mathcal{S}_2$),

$$L_1 \subseteq L_2.$$

Afin de prouver cette proposition, définissons la notion suivante.

Définition 17 (Exécutions similaires)**.** *Soient $\mathcal{S}_1$ et $\mathcal{S}_2$ deux IDTMC telles que $\mathcal{S}_1$ raffine faiblement $\mathcal{S}_2$. Soit ρ_1 (respectivement ρ_2) une exécution finie de $\mathcal{S}_1$ (respectivement $\mathcal{S}_2$).*

Les exécutions ρ_1 et ρ_2 sont similaires *si, et seulement si,*

- *leurs longueurs sont égales ;*
- *si $\rho_1 = s_1 s_2 \dots s_k$ et $\rho_2 = p_1 p_2 \dots p_k$, alors tous les couples d'états $(s_i, p_i), i \in [1, k]$ sont en relation d'après le raffinement.*

Démonstration. Soient deux spécifications $\mathcal{S}_1 = (S_1, s_{0,1}, T_1, \Sigma, \lambda_1)$ et $\mathcal{S}_2 = (S_2, s_{0,2}, T_2, \Sigma, \lambda_2)$ telles que $\mathcal{S}_1$ raffine faiblement $\mathcal{S}_2$.

La preuve est réalisée par induction sur la propriété plus forte suivante (Bérard *et al.*, 2015b) : pour chaque exécution finie de $\mathcal{S}_1$, il existe une exécution similaire de $\mathcal{S}_2$. On suppose par induction que l'énoncé est vrai pour tout mot de longueur n, et on montre qu'il reste vrai pour un mot de longueur $n+1$. Soit w de longueur $n+1$, dans le langage des mots finis de $\mathcal{S}_1$. On note $w = w_0 a$ avec $a \in \Sigma$. Une exécution qui produit le mot w est de la forme $\rho = \rho_0 s_1$, avec $\lambda_1(s_1) = a$. L'exécution ρ_0 est une exécution de $\mathcal{S}_1$ de longueur n, donc par hypothèse d'induction, il existe une exécution similaire ρ_0' de $\mathcal{S}_2$. Par définition du raffinement, il existe une fonction $\delta : S_1 \to \mathcal{D}ist(S_2)$ telle que, pour $s_2 \in S_2$, si $\delta(s_1)(s_2) > 0$, alors $\lambda_1(s_1) = \lambda_2(s)$. Or $\delta(s_1)$ induit une distribution sur l'ensemble des états de $\mathcal{S}_2$, donc il existe au moins un $s_2 \in S_2$ tel que $\delta(s_1)(s_2) > 0$. Ainsi, ρ et $\rho' = \rho_0' s_2$ sont similaires, ce qui termine la preuve par induction.

Toute exécution finie de $\mathcal{S}_1$ possède donc une exécution similaire dans $\mathcal{S}_2$, donc par définition de la similitude d'exécutions, on en déduit que tout mot fini du langage de $\mathcal{S}_1$ est un mot fini du langage de $\mathcal{S}_2$. Or un mot infini est la limite de la suite de ses préfixes finis, donc par passage à la limite, tout mot infini du langage de $\mathcal{S}_1$ est un mot infini du langage de $\mathcal{S}_2$, ce qui termine la preuve. □

Pour le cas particulier des PTS, si on considère deux PTS $\mathcal{A}_1$ et $\mathcal{A}_2$ tels que $\mathcal{A}_1 \preceq \mathcal{A}_2$, alors d'après la proposition 7, $\mathcal{A}_2 \preceq \mathcal{A}_1$, et ainsi $L_1 = L_2$ en appliquant la proposition 8.

De plus, puisqu'une implémentation d'un IDTMC $\mathcal{S}$ est un PTS $\mathcal{A}$ tel que $\mathcal{A} \preceq \mathcal{S}$, cela signifie que $L_\mathcal{A} \subseteq L_\mathcal{S}$. On peut donc poser la définition suivante, étendue aux IDTMC modales.

Définition 18. *Le langage L d'une IDTMC $\mathcal{S}$ est :*

$$L = \bigcup_{\mathcal{A} \in imp(\mathcal{S})} L_{\mathcal{A}}.$$

Enfin, on établit la régularité du langage d'une spécification.

Proposition 9. *Le langage d'une IDTMC est ω-régulier.*

Démonstration. Le langage d'une IDTMC à nombre fini d'états ne dépend pas des probabilités de transitions, on peut donc la transformer immédiatement en LTS à nombre fini d'états. Un LTS à nombre fini d'états $\mathcal{A} = (Q, q_0, T, \Sigma, \lambda)$ peut être transformé en automate de Büchi non-déterministe : il suffit de poser la fonction de transition $\delta : Q \times \Sigma \to Q$ qui, à tout état $q \in Q$, associe les états $\delta(q, \lambda(q)) = T(q)$. Cela revient à transférer les étiquettes des états sur les transitions issues de ces états. De plus, tout état accessible du LTS devient un état acceptant de l'automate. Cette transformation induit donc que le langage reconnu par l'automate est exactement le langage du LTS. Ainsi, puisque tout langage reconnu par un automate est un langage ω-régulier, le langage de tout LTS à nombre fini d'états est ω-régulier – de même pour le langage de l'IDTMC initiale. □

4.5 Ordonnancement

À l'instar d'un MDP, une IDTMC est un modèle de spécification non-déterministe, qui abstrait de nombreuses implémentations, et qui possède plusieurs composantes probabilistes. Implémenter un tel système consiste à faire un choix parmi toutes les distributions existantes, afin d'aboutir à un unique système probabiliste, un PTS. Recourir à un ordonnanceur est un des moyens de faire ces choix de distributions, de façon successive en exécutant la spécification.

Dans une optique sécuritaire, l'ordonnanceur a une signification supplémentaire : on l'interprète comme un adversaire du système, dont le but est d'implémenter la spécification de façon à briser la sécurité. Ainsi, le meilleur adversaire est celui produisant l'implémentation qui possède le plus de chance d'être nuisible. L'étude des ordonnanceurs est donc naturelle dans le cadre du présent mémoire, et c'est la raison pour laquelle on utilisera par la suite les termes adversaire et ordonnanceur en tant que synonymes.

Définition 19 (Ordonnanceur)**.** *Un* ordonnanceur *pour une IDTMC $\mathcal{S} = (S, s_0, T, \Sigma, \lambda)$ est une application $A : FExec(\mathcal{S}) \to \mathcal{D}ist(S)$ telle que pour tout préfixe fini d'exécution*

$\rho \in FExec(\mathcal{S})$ se terminant dans un état s :

$$\forall s' \in S,\ A(\rho)(s') \in T(s)(s').$$

Exemple. *Considérons l'IDTMC représentée sur la figure 4.5a. Celle-ci est exactement la spécification de l'exemple de motivation de la figure 1.1, à laquelle on a remplacé les états par un alphabet plus générique, pour des soucis d'écriture. Pour rappel, le modèle représente un protocole d'échange sécurisé de message ; si le système est dans l'état q_1, il est en attente d'un message à envoyer ; s'il est dans l'état q_2, il transmet le message en clair, ce qui aboutit à une faille de sécurité ; enfin, s'il est dans l'état q_3, il transmet le message chiffré : c'est son comportement voulu. La spécification laisse indéterminée la distribution entre un comportement normal et un comportement défaillant pour chaque boucle d'exécution.*

Considérons alors par exemple l'ordonnanceur suivant, dont le rôle est de choisir les distributions de probabilité à chaque tour de boucle.

- *À la première utilisation du système, l'ordonnanceur choisit une distribution équiprobable.*
- *Deux exécutions peuvent alors se dérouler. Si le système rencontre une faille (donc si la trace commence par $q_1q_2q_1$), l'ordonnanceur choisit la distribution qui annule la probabilité de transmettre le message de manière chiffrée et rend la faille inévitable ; dans le cas contraire (donc si la trace commence par $q_1q_3q_1$), l'ordonnanceur choisit une distribution* $0.8/0.2$*, privilégiant à nouveau le comportement défaillant pour le second tour de boucle.*
- *Enfin, après le deuxième tour de boucle, l'ordonnanceur oublie les traces précédentes et considère que le système est de retour à son état initial.*

Un tel ordonnanceur implémente la spécification de manière à obtenir le PTS représenté sur la figure 4.5b.

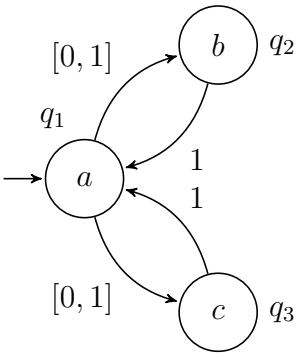
Notons que la définition est ici celle d'un ordonnanceur pour une IDTMC. On peut définir de façon très similaire un ordonnanceur pour un MDP.

L'ensemble des ordonnanceurs d'une spécification $\mathcal{S}$ est noté $Sched(\mathcal{S})$. L'ordonnancement d'une telle spécification par un ordonnanceur A induit la construction du PTS

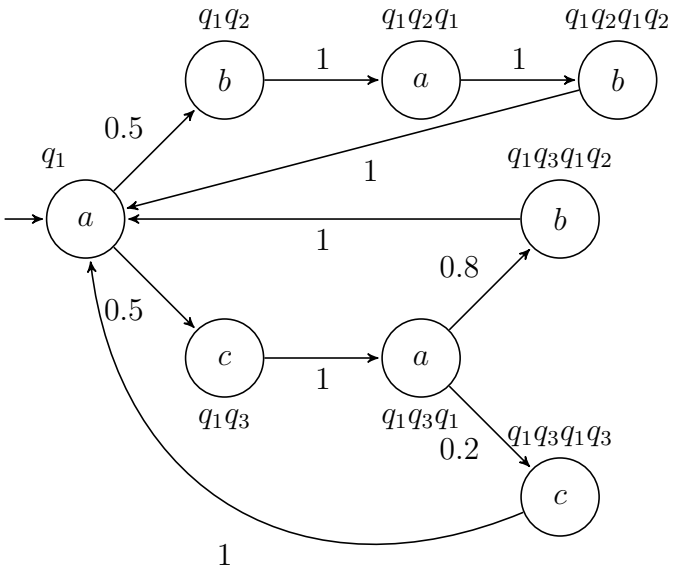
$$\mathcal{S}(A) = (Q, q_0, \Sigma, \Delta, L)$$

tel que :

- $Q \subseteq FExec(\mathcal{S})$ est l'ensemble des états, tel que chaque état est une exécution finie des états de $\mathcal{S}$;



(a) Spécification à ordonnancer



(b) Implémentation ordonnancée

Figure 4.5 Exemple d'ordonnancement

— q_0 est l'exécution ne contenant que l'état initial s_0 ;
— $\forall \rho \in Q$, $\forall s' \in S$, $\Delta(\rho)(\rho s') = A(\rho)(s')$;
— $\forall \rho \in Q$, $L(\rho) = \lambda(s)$ si s est le dernier état de S visité par l'exécution ρ.

Par construction, $\mathcal{S}(A) \in imp(\mathcal{S})$. De plus, on note $S\_imp(\mathcal{S}) = \{\mathcal{S}(A) \,|\, A \in Sched(\mathcal{S})\}$ l'ensemble des implémentations ordonnancées. La remarque précédente prouve alors que

$$S\_imp(\mathcal{S}) \subseteq imp(\mathcal{S}).$$

Remarque. *Cependant, l'égalité n'est pas assurée.*

En effet, considérons l'exemple de la figure 4.6. On affirme que le PTS représenté sur la

figure 4.6b est une implémentation de l'IDTMC $\mathcal{S}$ mais n'a pas été ordonnancé.

On montre que c'est une implémentation en posant $\mathcal{R}$ la relation de $Q \times S$ telle que $\mathcal{R} = \{(q_0, s_0), (q_1, s_1), (q_1, s_2), (q_2, s_1), (q_2, s_2), (q_3, s_1), (q_3, s_2)\}$.

Puisque $q_0 \mathcal{R} s_0$, posons $\delta : Q \to \mathcal{D}ist(S)$ tel que :

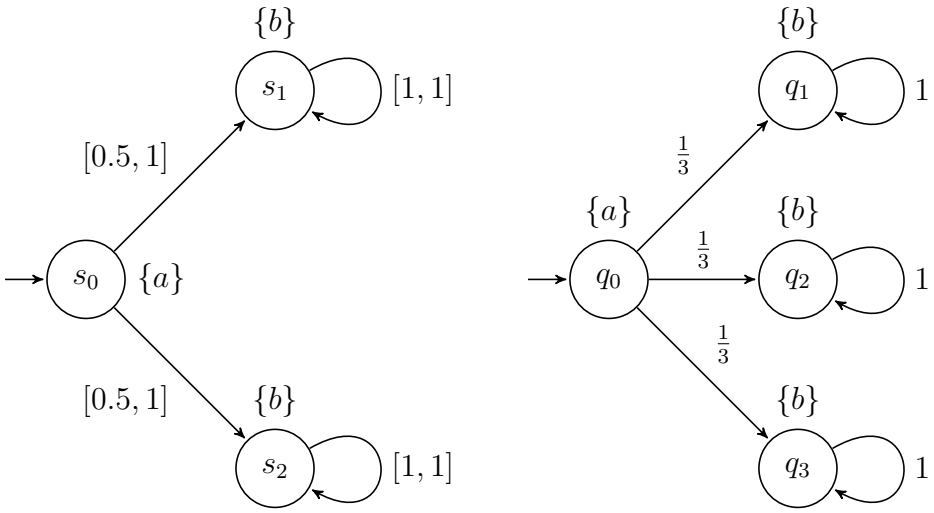
$$\begin{cases} \delta(q_0)(s_0) = 1; \\ \delta(q_1)(s_1) = 0.5 \ et \quad \delta(q_1)(s_2) = 0.5; \\ \delta(q_2)(s_1) = 0.5 \ et \quad \delta(q_2)(s_2) = 0.5; \\ \delta(q_3)(s_1) = 0.5 \ et \quad \delta(q_3)(s_2) = 0.5. \end{cases}$$

Alors, pour s_1,

$$\sum_{q' \in Q} \Delta(q_0)(q') \cdot \delta(q')(s_1) = \frac{1}{3} \times 0.5 + \frac{1}{3} \times 0.5 + \frac{1}{3} \times 0.5 = 0.5 \in T(s_0)(s_1).$$

On obtient le même résultat pour s_2, et pour s_0, le résultat est trivial. Dans tous les cas, on retrouve donc la condition de raffinement d'une spécification par un PTS.

Donc, le PTS $\mathcal{A}$ est bien une implémentation de $\mathcal{S}$.



(a) Exemple d'IDTMC $\mathcal{S}$ (b) PTS non-ordonnancé $\mathcal{A}$ de $\mathcal{S}$

Figure 4.6 Exemple d'implémentation non-ordonnancée d'une IDTMC

Cependant, ce n'est pas un ordonnancement.

Par l'absurde, supposons qu'il existe $A : FExec(\mathcal{S}) \to \mathcal{D}ist(\mathcal{S})$ tel que $\mathcal{S}(A) = \mathcal{A}$, c'est-à-dire que $\forall \rho \in Q$, $\forall s' \in S$, $A(\rho)(s') = \Delta(\rho)(\rho s')$. Alors, plus particulièrement, $A(s_0)(s_1)$ doit être

égal à $\Delta(q_0)(q_1)$, $\Delta(q_0)(q_2)$ *ou* $\Delta(q_0)(q_3)$. *Donc nécessairement,* $A(s_0)(s_1) = \frac{1}{3}$ *; mais dans ce cas,* $A(s_0)(s_1) \notin T(s_0)(s_1)$, *ce qui est absurde.*

On a montré de cette manière que l'égalité entre $S\_imp(\mathcal{S})$ *et* $imp(\mathcal{S})$ *n'est pas assurée en règle générale.*

Un ordonnanceur général est un objet dont la mémoire est infinie : en effet, un ordonnanceur résout l'indéterminisme étant donné le préfixe fini qu'il lit, parmi un ensemble infini de préfixes finis. Dans un cadre plus pratique, tout système réel possède une mémoire finie : on peut donc définir un sous-ensemble d'ordonnanceurs.

Définition 20 (Ordonnanceur à mémoire finie)**.** *Notons* $[n] = \{1, \ldots, n\}$, *avec* n *un entier. Un* ordonnanceur à mémoire n *pour une IDTMC* $\mathcal{S} = (S, s_0, T, \lambda)$ *est un tuple* $A = ([n], i_0, \theta, \gamma)$ *tel que*

- $[n]$ *est un ensemble fini de modes ;*
- i_0 *est le mode initial ;*
- $\theta : [n] \times S \to [n]$ *est une fonction de transition de modes ;*
- $\gamma : [n] \times S \to \mathcal{D}ist(S)$ *est une fonction de choix vérifiant*

$$\forall (i, s) \in [n] \times S,\ \forall s' \in S,\ \gamma(i, s)(s') \in T(s)(s').$$

Concrètement, l'ensemble des modes représente l'ensemble des états de la mémoire de l'ordonnanceur ; celle-ci est modifiée par la fonction de transition de modes δ. En parallèle, la fonction γ est appelée et choisit la distribution parmi toutes celles qui sont disponibles.

L'ensemble des ordonnanceurs à mémoire n d'une spécification $\mathcal{S}$ est noté $Sched_n(\mathcal{S})$. On note également $S\_imp_n(\mathcal{S}) = \{\mathcal{S}(A) \mid A \in Sched_n(\mathcal{S})\}$ l'ensemble des implémentations ordonnancées par un ordonnanceur à mémoire n.

Un cas particulier d'ordonnanceur à mémoire finie est celui de l'ordonnanceur sans-mémoire – élément de $Sched_1(\mathcal{S})$. Le PTS obtenu alors est exactement de la même forme que l'IDTMC de départ – hormis les éventuelles transitions modales, dont l'ordonnanceur peut annuler les probabilités.

4.6 Extension de l'opacité libérale aux IDTMC

Passons à présent à la formalisation de la mesure d'opacité dans le cadre des modèles de spécification. Comme expliqué dans la section 3.1, une spécification IDTMC $\mathcal{S}$ peut se caractériser par l'ensemble de ses implémentations, noté $imp(\mathcal{S})$, ou bien par l'ensemble de ses

implémentations ordonnancées, $S\_imp(\mathcal{S})$. Or, on sait calculer, d'après le chapitre 3, les différentes valeurs d'opacité pour des implémentations. Ainsi, intuitivement, il s'agit de calculer ces valeurs pour toutes les implémentations de la spécification, afin d'obtenir un encadrement des valeurs d'opacité que l'on peut obtenir en implémentant $\mathcal{S}$. La mesure de l'opacité pour la spécification est alors le pire des cas sur l'ensemble de ses implémentations. Dans le cas de l'opacité libérale, d'après la propriété 1, plus la valeur de l'opacité libérale d'une implémentation est grande, plus cette implémentation a de chance de divulguer le secret. Ainsi, l'implémentation la moins sécurisée d'une spécification est celle qui possède l'opacité libérale la plus élevée.

Définition 21 (Opacité libérale d'une spécification)**.** *Soient une spécification IDTMC $\mathcal{S}$, un secret ω-régulier φ et un observateur rationnel $\mathcal{O}$. L'*opacité libérale de φ dans $\mathcal{S}$ relativement à $\mathcal{O}$ *est la grandeur :*

$$PO_l(\mathcal{S}, \varphi, \mathcal{O}) = \sup_{\mathcal{A} \in imp(\mathcal{S})} PO_l(\mathcal{A}, \varphi, \mathcal{O}).$$

Si on restreint le calcul à l'ensemble des implémentations ordonnancées, on définit :

$$S\_PO_l(\mathcal{S}, \varphi, \mathcal{O}) = \sup_{\mathcal{A} \in S\_imp(\mathcal{S})} PO_l(\mathcal{A}, \varphi, \mathcal{O}) = \sup_{A \in Sched(\mathcal{S})} PO_l(\mathcal{S}(A), \varphi, \mathcal{O}).$$

Avec ce type de propriété, la sécurité est brisée lorsque le secret n'est pas opaque dans le système, c'est-à-dire lorsque l'observateur extérieur est capable de distinguer entre les exécutions secrètes et non-secrètes. L'adversaire a pour but de faciliter ce bris de sécurité, dans le but de divulguer toute l'information à l'observateur. En somme, dans cette étude, la sécurité est brisée dès lors qu'il y a un canal de communication entre l'adversaire et l'observateur passif. Ceci justifie le fait que l'on s'intéresse uniquement à l'étude des implémentations ordonnancées dans la suite du mémoire.

Ce chapitre a permis de poser les fondations du cadre théorique de l'étude. Après avoir défini le modèle de spécification de l'IDTMC, des moyens de raffiner ces spécifications, de les implémenter et enfin de les ordonnancer, nous avons étendu la notion d'opacité libérale déjà définie sur les PTS. À partir de ces définitions, il est possible de modéliser des systèmes sécurisés opaques ou non-opaques.

Le prochain chapitre s'applique à étudier plus précisément l'opacité libérale dans les spécifications. Notamment, on prouve que la mesure de l'opacité libérale est décidable dans le cas de spécifications non-modales ; on prouve également qu'elle peut être approchée en un temps fini dans le cas général ; enfin, on prouve que l'opacité est préservée quand la spécification est raffinée.

CHAPITRE 5 VÉRIFICATION DE L'OPACITÉ

Ce chapitre a pour but de prouver les principaux résultats du mémoire, en étudiant le comportement de l'opacité libérale dans les spécifications. Les deux principaux objectifs de ce chapitre sont de prouver que la mesure de l'opacité libérale dans une spécification IDTMC est calculable en un temps fini d'une part, et que l'opacité libérale est préservée lors du raffinement de spécification d'autre part.

Le premier résultat pose un problème car l'on remarque que le calcul est plus difficile dans le cas des IDTMC modales, dûes aux modifications du langage causées par la modalité des transitions. Ainsi, on se penche avant tout sur le cas des spécifications non-modales dans la section 5.2, puis on propose un algorithme permettant d'approcher le calcul dans le cas général dans la section 5.3. La section 5.1 présente les notions préliminaires à la construction des algorithmes de calcul.

Le second résultat est prouvé dans la section 5.4 : notamment, on prouve que l'opacité est préservée par passage au raffinement faible. Aucun résultat n'est précisé dans le cas du raffinement complet.

Enfin, la section 5.5 généralise les notions de quasi-opacité uniforme et d'opacité restrictive aux spécifications IDTMC. De plus, on constate que la quasi-opacité uniforme se comporte de manière très similaire à l'opacité libérale, puisqu'elles sont toutes deux issues de probabilités d'ensembles ω-réguliers.

5.1 Notions préliminaires

Cette section introduit les notions nécessaires à l'exécution de l'algorithme de calcul de l'opacité libérale dans une spécification, particulièrement dans le cas non-modal, détaillé dans la section 5.2. Notamment, on définit le processus de synchronisation entre un DPA et une IDTMC, ainsi que le processus de construction d'un MDP à partir d'une IDTMC. Cette seconde procédure demande au préalable l'introduction de la notion de Solution Basique Réalisable (BFS), dans la section 5.1.2.

5.1.1 Synchronisation entre un DPA et une IDTMC

L'une des étapes principales de l'algorithme de calcul consiste en la synchronisation d'un DPA, représentant un langage régulier, avec l'IDTMC étudiée. On construit ainsi une nouvelle IDTMC, contenant l'information du langage régulier à vérifier.

Cette procédure est définie de la façon suivante.

Définition 22. *Soient un DPA* $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ *et une IDTMC* $\mathcal{S} = (S, s_0, T, \Sigma, \lambda)$*. La* synchronisation *entre* $\mathcal{A}$ *et* $\mathcal{S}$ *est l'IDTMC colorée* $\mathcal{S} \otimes \mathcal{A} = (S \times Q, (s_0, q_0), T', \Sigma, \lambda', F')$ *telle que, pour tout état* $(s, q) \in S \times Q$*,*

— $\forall\, s' \in S,\ T'\Big((s,q)\Big)\Big(\Big(s', \delta(q, \lambda(s))\Big)\Big) = T(s)(s')$ *;*
— $\lambda'\Big((s,q)\Big) = \lambda(s)$ *;*
— $F'\Big((s,q)\Big) = F(q)$*.*

Cette définition se traduit de la façon suivante : si le DPA est dans un état q et l'IDTMC dans un état s simultanément, alors l'automate lit l'étiquette $\alpha = \lambda(s)$: d'après sa fonction de transition, son prochain état est nécessairement $\delta(q, \alpha)$. L'IDTMC, quant à elle, choisit la distribution comme à son habitude.

Il est nécessaire de conserver dans l'IDTMC finale les couleurs appliquées aux différents états afin de détecter les exécutions qui appartiennent au langage reconnu par le DPA : celles-ci vérifient la condition d'acceptation des automates de parité (*cf.* définition 2). Illustrons ceci sur un exemple.

Exemple. *Pour illustrer ce processus, considérons le DPA représenté sur la figure 3.1 ainsi que l'IDTMC représentée sur la figure 4.1, et réalisons la synchronisation entre ces deux modèles. Le résultat est représenté sur la figure 5.1.*

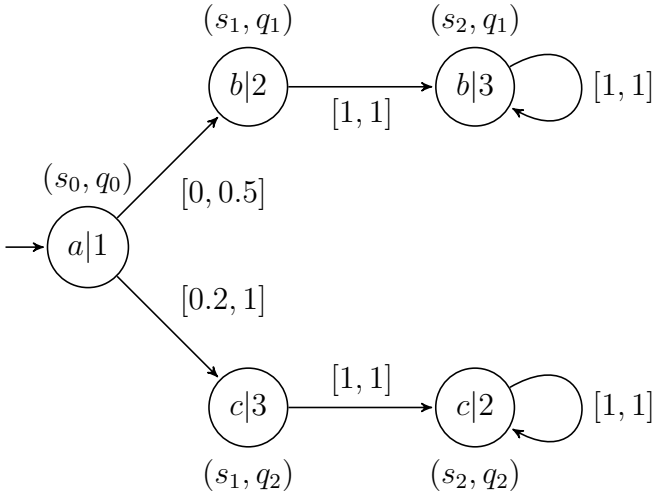


Figure 5.1 Exemple de synchronisation entre un DPA et une IDTMC

5.1.2 Solution Basique Réalisable (BFS)

Cette section permet d'introduire la notion de Solution Basique Réalisable (BFS) que l'on utilise par la suite. C'est une notion mathématique, issue du domaine de l'optimisation linéaire. Elle s'applique donc naturellement à notre cadre théorique, doté de modèles linéaires et de grandeurs à optimiser.

Définition 23 (Solution Basique Réalisable (BFS))**.** *Soit n un entier naturel, soient $(a_i)_{i\in[1,n]}$ et $(b_i)_{i\in[1,n]}$ dans $\mathbb{R}^n$, et soit le problème linéaire contraint $\mathcal{P}$ suivant.*

$$\mathcal{P} = \begin{cases} & \sum_{i=1}^{n} x_i = 1 \\ \forall i \in [1,n], & x_i \in [a_i, b_i]. \end{cases}$$

Le vecteur $\alpha = (\alpha_i, i \in [1,n])$ est une Solution Basique Réalisable (BFS) de ce problème si, et seulement si, il existe au plus *un indice $k \in [1,n]$ tel que $\alpha_k \neq a_k$ et $\alpha_k \neq b_k$.*

L'ensemble des BFS constitue une base de l'ensemble vectoriel des solutions, de sorte que toute solution générale du problème s'exprime comme une combinaison linéaire des BFS.

L'intérêt de cette notion réside dans le fait qu'il suffit de se restreindre à l'étude de la frontière du polyèdre. Plus précisément, il y a un nombre fini de BFS pour un problème donné.

Proposition 10. *Il y a au maximum $n \cdot 2^{n-1}$ points de l'espace qui peuvent être les BFS du problème linéaire $\mathcal{P}$.*

Démonstration. Soit $\alpha = (\alpha_i, i \in [1,n])$ une BFS du problème $\mathcal{P}$. C'est par définition une solution du problème, donc si l'on connaît $n-1$ de ses composantes, on peut calculer la n-ième. On sait également qu'au moins $n-1$ de ses composantes sont des frontières d'intervalles. D'après les coefficients binomiaux, il y a $\binom{n}{n-1} = n$ façons de choisir quelles sont ces composantes. La n-ième composante est alors connue.

Pour chacune des composantes frontières, il reste alors à choisir si elle est égale à la frontière haute (b_i) ou basse (a_i) : il y a donc deux choix pour chacune des $n-1$ composantes, c'est-à-dire, 2^{n-1} choix en tout.

Finalement, il y a au maximum $n \cdot 2^{n-1}$ choix possibles pour déterminer les BFS du problème.

□

Illustrons ceci par un exemple simple.

Exemple. *Soit le problème suivant.*

$$\begin{cases} x + y = 1 \\ x \in [0.5, 1] \\ y \in [0, 1] \end{cases}$$

Notons $C = [0.5, 1] \times [0, 1]$ *l'ensemble des contraintes ; c'est un polyèdre sur l'espace vectoriel* $\mathbb{R}^2$*. Il est représenté par le rectangle jaune sur la figure 5.2. La droite d'équation* $x + y = 1$ *est représentée en bleu.*

Les points de $\mathbb{R}^2$ *qui peuvent être des BFS du problème sont* $(0.5, 0.5)$*,* $(1, 0)$ *et* $(0, 1)$*. Or le point* $(0, 1)$ *n'est pas dans le domaine des contraintes, il est donc écarté. Ce processus est illustré par l'intersection entre la droite et la frontière du polyèdre. Les BFS sont donc :*

$$\begin{cases} x = 0.5 \ \textit{ et } y = 0.5; \\ x = 1 \quad \textit{ et } y = 0. \end{cases}$$

Une solution générale au problème s'écrit alors :

$$(x_0, y_0) = k_1 \cdot (0.5, 0.5) + k_2 \cdot (1, 0)$$

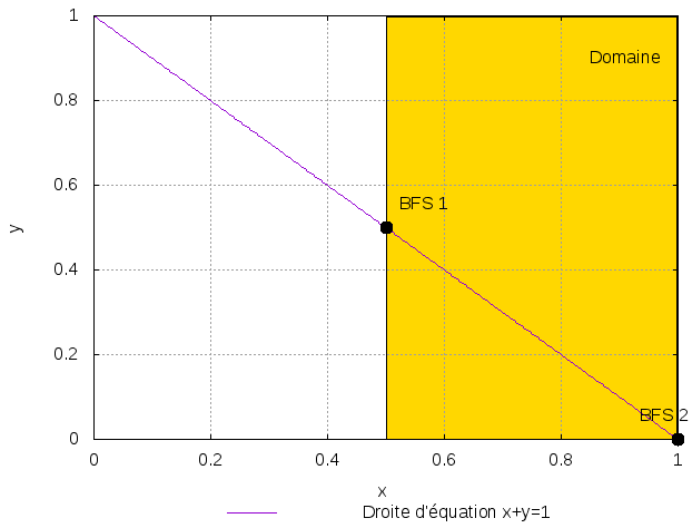


Figure 5.2 Illustration de la résolution du problème de l'exemple

5.1.3 Calcul d'un MDP à partir d'une IDTMC

Une seconde étape importante de l'algorithme de calcul consiste en la transformation d'une IDTMC en un MDP. En effet, l'algorithme de mesure de probabilité maximale d'un langage ω-régulier dans un MDP est déjà connu (Bérard *et al.*, 2015a), alors que le même calcul dans une spécification IDTMC ne l'est pas.

La procédure est représentée sur l'algorithme 1 et est fondée sur la notion de BFS (Chatterjee *et al.*, 2008).

Pour un état s donné d'une spécification $\mathcal{S} = (S, s_0, T, \Sigma, \lambda)$, $T(s)$ décrit l'ensemble des distributions acceptées par la spécification. Mathématiquement, c'est une équation linéaire en les probabilités de transitions, dont les inconnues sont restreintes dans un ensemble fermé borné. On note ce problème dans toute la suite $\mathcal{P}(s)$.

$$\mathcal{P}(s) = \begin{cases} \sum_{s' \in S} \mathbf{P}(s_{i+1} = s' \,|\, s_i = s) & = & 1 \\ \forall\, s' \in S,\, \mathbf{P}(s_{i+1} = s' \,|\, s_i = s) & \in & T(s)(s') \end{cases}$$

Il respecte toutes les conditions de la définition 23. Ainsi, en appliquant la procédure décrite dans la section 5.1.2, on peut lister les différentes BFS de ce problème au sommet $\mathcal{P}(s)$. Notons que dans ce cas particulier, chaque BFS est une distribution sur les états de $\mathcal{S}$.

On définit alors le MDP $\mathcal{M}$ tel que l'ensemble de ses états est exactement l'ensemble des états de l'IDTMC $\mathcal{S}$ et tel que son ensemble de distributions de base correspond exactement à l'ensemble des BFS des problèmes exprimés précédemment aux états correspondants – par exemple, pour l'état initial, le MDP propose comme distributions toutes les BFS du problème $\mathcal{P}(s_0)$ lié à l'état initial.

input : IDTMC $\mathcal{S} = (S, s_0, T, \Sigma, \lambda)$
output: MDP $\mathcal{M} = (S, s_0, A, \Delta, \Sigma, \lambda)$

```
1  A = ∅;
2  Δ = null;
3  foreach s ∈ S do
4  |   Trouver les BFS μ du problème P(s);
5  |   A = A ∪ {μ | μ est une BFS de P(s)};
6  |   foreach μ BFS du problème P(s) do
7  |   |   Δ((s), μ) = μ;
8  |   end
9  end
10 M = (S, s_0, A, Δ, Σ, λ);
```

Algorithme 1: Algorithme de construction d'un MDP à partir d'une IDTMC

Un tel MDP traduit exactement la même spécification que l'IDTMC initial. En effet, on remarque qu'ordonnancer une spécification IDTMC revient exactement à trouver une solution à chaque problème au sommet : pour chaque $s \in S$, il s'agit de trouver une distribution dans $T(s)$. On remarque alors qu'on peut résoudre ces problèmes de deux manières : soit par la méthode directe, qui revient à l'ordonnancement direct de l'IDTMC ; soit par la méthode expliquée ci-dessus, en exprimant une solution générale en tant que distribution de BFS. En clair, un tel ordonnanceur choisit les poids qu'il donne aux différentes distributions : c'est exactement le principe d'un ordonnanceur de MDP (*cf.* définition 6).

La proposition 10 ainsi que le fait que l'on calcule les BFS pour chaque état de l'IDTMC donne le résultat suivant.

Corollaire 1. *Le MDP issu de l'algorithme 1 est de taille exponentielle par rapport à la taille de l'IDTMC initial.*

Ces notions préliminaires définies, il est temps de les utiliser dans l'algorithme de calcul de l'opacité libérale. Commençons par le cas plus simple des spécifications non-modales.

5.2 Calcul de l'opacité libérale dans le cas des IDTMC non-modales

La modalité d'une IDTMC peut poser problème. En effet, comme indiqué dans la section 4.4, lorsqu'on implémente une spécification modale, on peut modifier le langage. Ainsi, si $\mathcal{S}$ est une IDTMC modale, $S\_imp(\mathcal{S})$ est probablement composé de plusieurs groupes de PTS, différenciés selon leur langages. Or l'ensemble des exécutions qui divulguent le secret, dont on veut calculer la probabilité lors de la mesure d'opacité, dépend du langage : le calcul de la borne supérieure sur l'ensemble des implémentations ordonnancées est donc plus complexe.

Pour cette raison, étudions tout d'abord le cas des spécifications non-modales, c'est-à-dire celles qui ont un langage unique, afin de s'affranchir de cette difficulté. Nous énonçons le résultat suivant.

Théorème 2 (Décidabilité de l'opacité libérale)**.** *Soient une spécification IDTMC* $\mathcal{S}$ non-modale*, un secret* ω*-régulier* φ *et un observateur rationnel* $\mathcal{O}$*.*

Alors la grandeur $S\_PO_l(\mathcal{S}, \varphi, \mathcal{O})$ *est mesurable.*

Démonstration. La preuve de ce théorème consiste en une preuve algorithmique. Elle décrit l'algorithme 2.

input : $\mathcal{S} = (S, s_0, T, \Sigma, \lambda)$, φ et $\mathcal{O}$
output: $S\_PO_l(\mathcal{S}, \varphi, \mathcal{O})$

1 Construire le DPA $\mathcal{A}_\mathcal{V} = (Q, \Sigma, \delta, q_0, F)$;
2 // Construction de $\mathcal{S}_\mathcal{V} = \mathcal{S} \otimes \mathcal{A}_\mathcal{V}$
3 **foreach** $(s, q) \in S \times Q$ **do**
4 **foreach** $s' \in S$ **do**
5 $T'(s, q)(s', \delta(q, \lambda(s))) = T(s)(s')$;
6 **end**
7 $\lambda'(s, q) = \lambda(s)$;
8 $F'(s, q) = F(q)$;
9 **end**
10 $\mathcal{S}_\mathcal{V} = (S \times Q, (s_0, q_0), T', \Sigma, \lambda', F')$;
11 // Construction du MDP $\mathcal{M}_\mathcal{V}$
12 $A = \emptyset$;
13 $\Delta = null$;
14 **foreach** $(s, q) \in S \times Q$ **do**
15 Trouver les BFS μ du problème $\mathcal{P}(s, q)$;
16 $A = A \cup \{\mu \,|\, \mu \text{ est une BFS de } \mathcal{P}(s, q)\}$;
17 **foreach** μ *BFS du problème* $\mathcal{P}(s, q)$ **do**
18 $\Delta((s, q), \mu) = \mu$;
19 **end**
20 **end**
21 $\mathcal{M}_\mathcal{V} = (S \times Q, (s_0, q_0), A, \Delta, \Sigma, \lambda', F')$;
22 Calculer la probabilité maximale $\mathbf{P}$ du langage reconnu par $\mathcal{M}_\mathcal{V}$;
23 $S\_PO_l(\mathcal{S}, \varphi, \mathcal{O}) = \mathbf{P}$;

Algorithme 2: Algorithme de calcul de l'opacité libérale dans une IDTMC non-modale

Étape 1. La première étape (lignes 1 à 10) consiste à remarquer que le secret φ est un langage ω-régulier : ainsi puisque l'ensemble des langages ω-réguliers est fermé pour l'intersection et que l'observateur est rationnel, l'ensemble $\mathcal{V}(\mathcal{S}, \varphi, \mathcal{O})$ est lui-même ω-régulier. Le problème se réduit donc au calcul de la probabilité maximale d'un langage régulier dans une IDTMC. Pour cela, on s'inspire des travaux de Bérard *et al.* (2015c) : il s'agit de construire l'IDTMC qui correspond à l'intersection du système et du langage.

On peut construire un DPA, noté $\mathcal{A}_\mathcal{V}$, qui reconnaît le langage $\mathcal{V}(\mathcal{S}, \varphi, \mathcal{O})$. On le construit en déterminisant l'automate de Büchi reconnaissant le même langage (Piterman, 2007), que l'on obtient en réalisant des compositions et intersections d'automates à partir de l'automate reconnaissant le secret φ : cette procédure est de complexité exponentielle.

L'intersection entre le système et le langage se réalise par la synchronisation (*cf.* définition 22) de ce DPA avec l'IDTMC initiale : on obtient une IDTMC colorée synchronisée, notée

$\mathcal{S}_\mathcal{V} = \mathcal{S} \otimes \mathcal{A}_\mathcal{V}$.

Étape 2. La seconde étape (lignes 11 à 21) consiste à construire le MDP correspondant à cette IDTMC $\mathcal{S}_\mathcal{V}$ à partir de la procédure décrite dans la section 5.1.3. On le note $\mathcal{M}_\mathcal{V}$. Ce MDP contient décrit exactement la même spécification que l'IDTMC $\mathcal{S}_\mathcal{V}$.

Étape 3. La dernière étape consiste alors à maximiser la probabilité du langage $\mathcal{V}(\mathcal{S}, \varphi, \mathcal{O})$ dans le MDP $\mathcal{M}_\mathcal{V}$. Cette probabilité est exactement la grandeur $S\_PO_l(\mathcal{S}, \varphi, \mathcal{O})$, l'opacité libérale de φ dans l'IDTMC $\mathcal{S}$. □

D'après (Piterman, 2007), la procédure de construction du DPA reconnaissant le langage $\mathcal{V}(\mathcal{S}, \varphi, \mathcal{O})$ est de complexité exponentielle. De plus, d'après la section 5.1.3, la construction du MDP $\mathcal{M}_\mathcal{V}$ est exponentielle en la taille de l'IDTMC. Ainsi, on en déduit le corollaire suivant.

Corollaire 2 (Complexité de l'algorithme de calcul de l'opacité libérale)**.** *Le calcul de l'opacité libérale dans une spécification non-modale est de complexité doublement exponentielle.*

Ce résultat constitue l'un des principaux résultats du mémoire, en ce sens qu'il justifie l'utilisation de ce cadre et de cette propriété de sécurité : l'opacité est effectivement mesurable dans une spécification non-modale, et nous avons un algorithme permettant de la calculer.

Cependant, ce résultat reste encore insuffisant, puisqu'il ne concerne qu'un nombre restreint de spécifications. La section suivante tente de remédier à ce problème en cherchant à approcher la mesure de l'opacité dans le cas général.

5.3 Une approximation du cas général

Le théorème 2 s'applique uniquement aux spécifications qui ne possèdent aucune transition modale. L'objet de cette partie est de construire un algorithme permettant un calcul approché de l'opacité libérale dans le cas des spécifications modales.

Pour cela, remarquons que l'ordonnancement d'une IDTMC modale consiste en le choix d'une distribution pour chaque préfixe fini d'exécution (*cf.* définition 19). Ainsi, s'il s'avère que l'une des transitions à venir est modale, l'ordonnanceur peut choisir une distribution qui annulera la probabilité de cette transition. On peut alors considérer que ce choix d'annuler telle ou telle transition est réalisé avant le choix des probabilités des autres transitions – l'ordre d'affectation des probabilités importe peu. Ainsi, on peut supposer que, pour chaque

préfixe fini, l'ordonnanceur choisit un sous-ensemble des transitions modales qu'il annule, puis ajuste la distribution avec les transitions restantes.

Cette procédure laisse penser que l'on peut séparer le travail de l'ordonnanceur, entre son travail sur la présence des transitions, puis sur leurs probabilités. Cependant, dans le cas de l'ordonnanceur général, l'application reste limitée puisque la procédure est réalisée à chaque étape, pour chaque préfixe fini, sans aucun moyen de prédire les décisions futures.

Si, en revanche, on se restreint au cas des ordonnanceurs à mémoire finie, il est possible d'aller plus loin dans le raisonnement. En effet, le fait de considérer une mémoire finie implique que l'on connaît tous les comportements de l'ordonnanceur suivant l'état de sa mémoire. Autrement dit, pour un état de mémoire donné, l'ordonnanceur choisira toujours la même distribution – en particulier, il choisira toujours le même sous-ensemble de transitions modales à annuler. Par conséquent, si l'on a une spécification modale, on peut considérer que l'ordonnanceur choisit tout d'abord toutes les transitions qu'il désire annuler suivant les différents états de sa mémoire. Il obtient ainsi une spécification non-modale, qu'il peut ordonnancer comme à son habitude.

En définitive, l'idée qui ressort de cette introduction est la suivante : l'ordonnancement d'une spécification peut être décortiqué et réordonné. Nous formalisons cette idée dans la suite.

5.3.1 Détermination des transitions modales

Cette introduction au problème implique tout d'abord de pouvoir déterminer l'ensemble des transitions modales d'une spécification IDTMC. Ceci est réalisé par l'algorithme 3, inspiré de Bérard *et al.* (2015b).

Cet algorithme parcourt toutes les transitions de l'IDTMC et détermine pour chacune d'elles si elle est modale. On affirme qu'une transition est modale si, et seulement si, on peut lui donner une probabilité nulle ou non-nulle (ligne 4), et :

— soit la somme des bornes supérieures des autres intervalles est strictement supérieure à 1 (ligne 5) ;
— soit cette somme est égale à 1, et les bornes supérieures sont les maxima de leurs intervalles respectifs (ligne 7).

Cet algorithme se termine car les IDTMC considérées sont à ensemble d'états finis, donc les deux boucles sont finies.

```
   input  : Une spécification 𝒮 = (S, s_0, T, λ)
   output: L'ensemble des transitions modales ℰ_m(𝒮)
 1 ℰ_m = ∅;
 2 foreach s ∈ S do
 3 |   foreach s' ∈ S do
 4 |   |   if T(s)(s') ≠ {0} ∧ 0 ∈ T(s)(s') then
 5 |   |   |   if ∑_{t≠s'} sup T(s)(t) > 1 then
 6 |   |   |   |   ℰ_m = ℰ_m ∪ {(s, s')};
 7 |   |   |   else if ∑_{t≠s'} sup T(s)(t) = 1 ∧ ∀_{t≠s'} sup T(s)(t) ∈ T(s)(t) then
 8 |   |   |   |   ℰ_m = ℰ_m ∪ {(s, s')};
 9 |   |   end
10 |   end
11 end
```

Algorithme 3: Algorithme de détermination des transitions modales

5.3.2 Élimination de certaines transitions modales

Outre le besoin de déterminer l'ensemble des transitions modales d'une spécification, l'introduction émet le besoin d'en éliminer certaines afin d'obtenir une nouvelle spécification non-modale, sur laquelle on peut appliquer les théorèmes connus.

Cette procédure est expliquée par l'algorithme 4. L'objet créé possède exactement la même structure que l'IDTMC initiale, seule l'application T est modifiée. Ceci est réalisé de manière à assurer l'annulation des transitions de l'ensemble que l'on désire supprimer (lignes 2 et 3), tout en retirant la modalité des autres transitions modales (lignes 4 et 5).

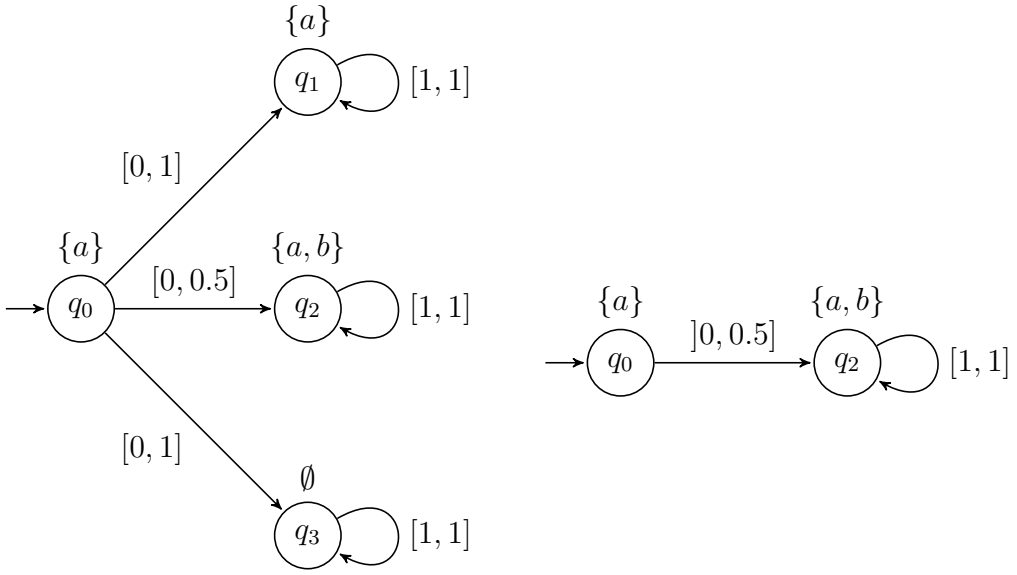
Puisque les spécifications IDTMC considérées sont à nombre fini d'états, l'ensemble des transitions modales est nécessairement fini, ce qui assure la terminaison de l'algorithme.

Il est à noter que l'objet créé par l'algorithme n'est pas nécessairement une IDTMC. En effet, étudions l'exemple de la figure 5.3a. Les trois transitions issues de q_0 sont évidemment modales, puisque si on annule l'une d'entre elles, il existe un moyen d'ajuster les probabilités des deux autres afin d'assurer une distribution issue de q_0. Supposons alors que l'on désire annuler les transitions modales (q_0, q_1) et (q_0, q_3). L'objet obtenu par l'algorithme est représenté sur la figure 5.3b – notons que les états q_1 et q_3 ne sont plus accessibles depuis l'état initial, leur représentation est donc inutile. Cet objet n'est évidemment pas une IDTMC.

input : $\mathcal{S} = (S, s_0, T, \Sigma, \lambda)$, $\mathcal{E} \in \mathscr{P}\Big(\mathcal{E}_m(\mathcal{S})\Big)$
output: $\mathcal{S} \setminus \mathcal{E}$

$T_{\mathcal{E}} = T$;
foreach $(s, s') \in \mathcal{E}$ **do**
  $T_{\mathcal{E}}(s)(s') = \{0\}$;
end
foreach $(s, s') \in \mathcal{E}_m(\mathcal{S}) \setminus \mathcal{E}$ **do**
  $T_{\mathcal{E}}(s)(s') = T(s)(s') \setminus \{0\}$;
end
$\mathcal{S} \setminus \mathcal{E} = (S, s_0, T_{\mathcal{E}}, \Sigma, \lambda)$

Algorithme 4: Algorithme d'élimination d'un ensemble de transitions modales dans une spécification



(a) Exemple d'IDTMC modale (b) Objet issu de l'algorithme

Figure 5.3 Application de l'algorithme 4 sur les transitions modales (q_0, q_1) et (q_0, q_3)

5.3.3 Dépliage de l'ordonnancement

Considérons le système de transitions non-étiquetées fini $\mathcal{A}_{n,\theta} = ([n], i_0, \theta)$ – ici, on prend en compte uniquement l'aspect traitement de mémoire de l'ordonnanceur. Cette première étape d'ordonnancement est réalisée en construisant l'IDTMC $\mathcal{S} \times \mathcal{A}_{n,\theta} = ([n] \times S, (i_0, s_0), T', \Sigma, \lambda')$ telle que :

— $\forall (i,s) \in [n] \times S,\ \forall s' \in S,\ T'(i,s)(\theta(i,s), s') = T(s)(s')$;
— $\forall (i,s) \in [n] \times S,\ \lambda'(i,s) = \lambda(s)$.

Ceci traduit exactement le comportement de l'IDTMC initiale, à laquelle on a ajouté l'ajustement de la mémoire de l'ordonnanceur. À cette étape, l'ordonnanceur n'a aucune action sur le système : en effet, on est en présence d'une spécification non-implémentée, aucun choix n'a été pris à propos des distributions.

Ces choix sont réalisés dans la seconde étape : il suffit alors d'ordonnancer $\mathcal{S} \times \mathcal{A}_{n,\theta}$ par l'ordonnanceur sans-mémoire dont la fonction de choix de distribution est γ' telle que

$$\forall (i,s) \in [n] \times S,\ \gamma'(i,s) \in T'(i,s).$$

On déduit de ce raisonnement que tout ordonnancement à mémoire finie n sur une spécification $\mathcal{S}$ est produit par un ordonnancement sans-mémoire sur la spécification $\mathcal{S} \times \mathcal{A}_{n,\theta}$, pour un certain système de transitions à n états $\mathcal{A}_{n,\theta}$. Notons par ailleurs qu'il n'existe qu'un nombre fini de tels systèmes de transitions, puisque l'on considère des IDTMC à nombre fini d'états. On déduit le résultat suivant.

Lemme 1. *Soit $\mathcal{S}$ une spécification IDTMC quelconque. Alors*

$$S\_imp_n(\mathcal{S}) = \bigcup_{\theta} S\_imp_1(\mathcal{S} \times \mathcal{A}_{n,\theta}).$$

5.3.4 Approximation du calcul de l'opacité libérale dans le cas des IDTMC modales

Rappelons que le problème consiste en le calcul du maximum des opacités libérales des implémentations ordonnancées d'une spécification. Les paragraphes précédents formalisent le dépliage d'un ordonnancement à mémoire finie ainsi que la détermination et l'élimination des transitions modales d'une spécification modale. Tel que discuté précédemment, le calcul sur l'ensemble des ordonnanceurs est *a priori* trop complexe : on se limite donc aux ordonnancements à mémoire finie et fixée – dans la suite du raisonnement, on choisit un entier n, représentant la taille de la mémoire des ordonnanceurs que l'on considère.

On note $S\_PO_{ln}(\mathcal{S}, \varphi, \mathcal{O})$ le maximum des opacités libérales sur $S\_imp_n(\mathcal{S})$.

Puisque $S\_imp_n(\mathcal{S}) \subseteq S\_imp(\mathcal{S})$, de manière évidente, $S\_PO_{ln}(\mathcal{S}, \varphi, \mathcal{O}) \leq S\_PO_l(\mathcal{S}, \varphi, \mathcal{O})$.

Le raisonnement décrit par l'algorithme 5 est le suivant. Puisque l'on sait calculer l'opacité libérale d'une spécification non-modale, on essaie de se ramener à ce cas en retirant les transitions modales. Pour cela, on considère uniquement l'ensemble $S\_imp_n(\mathcal{S})$, et on liste l'ensemble des ordonnanceurs qui aboutissent à ces implémentations.

Plus exactement, on liste l'ensemble des systèmes de transitions (ligne 2) à partir desquels on réalise le dépliage de la spécification (ligne 3). On obtient à ce stade une IDTMC modale dépliée, sur laquelle on peut appliquer successivement les algorithmes de détermination de transitions modales 3, puis d'élimination d'un sous-ensemble de ces transitions 4 : on réalise ce dernier algorithme pour tout sous-ensemble de l'ensemble des transitions modales (lignes 4 et 5). Pour chaque objet ainsi créé, on vérifie si c'est bien une IDTMC – autrement dit, si l'on peut créer une distribution sur l'ensemble des états à partir de chaque état (ligne 6). Dans ce cas, il est alors possible d'appliquer l'algorithme du théorème 2, puisqu'on est en présence d'une spécification non-modale.

La valeur de sortie de cet algorithme est notée $S\_PO_l(n, \mathcal{S}, \varphi, \mathcal{O})$ et est le maximum de toutes les valeurs obtenues à l'intérieur des boucles. On peut mettre à jour cette valeur à chaque tour de boucle (ligne 7).

Théorème 3 (Approximation de l'opacité libérale d'une spécification modale)**.** *Soient une spécification IDTMC $\mathcal{S}$, un secret ω-régulier φ et un observateur rationnel $\mathcal{O}$. Alors on peut approcher par la borne inférieure l'opacité libérale de φ dans $\mathcal{S}$ et approcher par la borne supérieure l'opacité libérale de φ dans $\mathcal{S}$ limitée à ses ordonnanceurs à mémoire finie.*

Preuve de correction de l'algorithme 5. Soient une spécification IDTMC $\mathcal{S}$, un secret ω-régulier φ, un observateur rationnel $\mathcal{O}$ et un entier naturel n.

La sortie de l'algorithme est la valeur :

$$S\_PO_l(n, \mathcal{S}, \varphi, \mathcal{O}) = \sup_{\theta} \sup_{\mathcal{E}} S\_PO_l(\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}, \varphi, \mathcal{O}).$$

On prouve qu'elle vérifie la post-condition

$$S\_PO_{ln}(\mathcal{S}, \varphi, \mathcal{O}) \leq S\_PO_l(n, \mathcal{S}, \varphi, \mathcal{O}) \leq S\_PO_l(\mathcal{S}, \varphi, \mathcal{O}). \tag{5.1}$$

input : $n \in \mathbb{N}$, $\mathcal{S} = (S, s_0, T, \Sigma, \lambda)$, φ et $\mathcal{O}$
output: $S\_PO_l(n, \mathcal{S}, \varphi, \mathcal{O})$ tel que
$S\_PO_{ln}(\mathcal{S}, \varphi, \mathcal{O}) \leq S\_PO_l(n, \mathcal{S}, \varphi, \mathcal{O}) \leq S\_PO_l(\mathcal{S}, \varphi, \mathcal{O})$

$S\_PO_l(n, \mathcal{S}, \varphi, \mathcal{O}) = 0$;
foreach $\theta : [n] \times S \to [n]$ **do**
  construire $\mathcal{S} \times \mathcal{A}_{n,\theta}$;
  foreach $\mathcal{E} \in \mathscr{P}\Big(\mathcal{E}_m(\mathcal{S} \times \mathcal{A}_{n,\theta})\Big)$ **do**
    construire $\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}$;
    if $\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}$ *est une IDTMC* **then**
      $S\_PO_l(n, \mathcal{S}, \varphi, \mathcal{O}) =$
      $\max\Big(S\_PO_l(n, \mathcal{S}, \varphi, \mathcal{O})\,;\, S\_PO_l(\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}, \varphi, \mathcal{O})\Big)$;
    end
  end
end

Algorithme 5: Algorithme de calcul de l'opacité libérale d'une spécification modale

Rappelons les définitions des différentes grandeurs :

$$\begin{cases} S\_PO_{ln}(\mathcal{S}, \varphi, \mathcal{O}) &= \sup_{A \in Sched_n(\mathcal{S})} PO_l(\mathcal{S}(A), \varphi, \mathcal{O}) &= \sup_{\mathcal{A} \in S\_imp_n(\mathcal{S})} PO_l(\mathcal{A}, \varphi, \mathcal{O}) \\ S\_PO_l(\mathcal{S}, \varphi, \mathcal{O}) &= \sup_{A \in Sched(\mathcal{S})} PO_l(\mathcal{S}(A), \varphi, \mathcal{O}) &= \sup_{\mathcal{A} \in S\_imp(\mathcal{S})} PO_l(\mathcal{A}, \varphi, \mathcal{O}) \end{cases}$$

Afin de prouver la première inégalité, on utilise le lemme 1 de dépliage.

$$S\_imp_n(\mathcal{S}) = \bigcup_{\theta} S\_imp_1(\mathcal{S} \times \mathcal{A}_{n,\theta})$$

Cela signifie qu'ordonnancer la spécification avec une mémoire finie est équivalent à ordonnancer sans-mémoire la spécification dépliée.

On calcule alors sur chacun de ces ensembles le maximum de l'opacité libérale du secret, d'après les définitions précédentes :

$$S\_PO_{ln}(\mathcal{S}, \varphi, \mathcal{O}) = \sup_{\theta} S\_PO_{l1}(\mathcal{S} \times \mathcal{A}_{n,\theta}, \varphi, \mathcal{O}) \tag{5.2}$$

De plus, pour θ fixé, on remarque que pour tout ordonnanceur sans-mémoire de $\mathcal{S} \times \mathcal{A}_{n,\theta}$, dont la fonction de choix de distribution est notée γ, il existe un sous-ensemble de transitions modales $\mathcal{E}$ tel que, $\gamma(s)(s') = 0$ si, et seulement si, $(s, s') \in \mathcal{E}$. Ainsi, tout ordonnancement sans-mémoire de $\mathcal{S} \times \mathcal{A}_{n,\theta}$ est équivalent à un ordonnancement sans-mémoire de la même spécification pour laquelle on a retiré les transitions de l'ensemble $\mathcal{E}$ correspondant à la

fonction de choix. On en déduit le résultat suivant.

$$S\_imp_1(\mathcal{S} \times \mathcal{A}_{n,\theta}) = \bigcup_{\mathcal{E}} S\_imp_1(\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E})$$

De plus, puisque toute implémentation ordonnancée sans mémoire est une implémentation ordonnancée, on déduit l'inclusion suivante.

$$\bigcup_{\mathcal{E}} S\_imp_1(\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}) \subseteq \bigcup_{\mathcal{E}} S\_imp(\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E})$$

On calcule alors sur chacun de ces ensembles le maximum de l'opacité libérale du secret, d'après les définitions précédentes :

$$S\_PO_{l1}(\mathcal{S} \times \mathcal{A}_{n,\theta}, \varphi, \mathcal{O}) = \sup_{\mathcal{E}} S\_PO_{l1}(\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}, \varphi, \mathcal{O}) \tag{5.3}$$

et

$$\sup_{\mathcal{E}} S\_PO_{l1}(\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}, \varphi, \mathcal{O}) \leq \sup_{\mathcal{E}} S\_PO_{l}(\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}, \varphi, \mathcal{O}). \tag{5.4}$$

Ainsi, en mettant en relation les équations (5.2) et (5.3) et l'inéquation (5.4), on obtient la première inégalité de la post-condition (5.1) :

$$S\_PO_{ln}(\mathcal{S}, \varphi, \mathcal{O}) \leq \sup_{\theta} \sup_{\mathcal{E}} S\_PO_{l}(\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}, \varphi, \mathcal{O}) = S\_PO_{l}(n, \mathcal{S}, \varphi, \mathcal{O}).$$

Afin de prouver la seconde inégalité, remarquons que toute implémentation ordonnancée de $\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}$ est une implémentation ordonnancée de la spécification initiale $\mathcal{S}$. Ainsi, $\forall \theta$, $\forall \mathcal{E}$, $S\_imp(\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}) \subseteq S\_imp(\mathcal{S})$ donc en calculant les opacités libérales et par passage à la borne supérieure sur θ et $\mathcal{E}$,

$$\sup_{\theta} \sup_{\mathcal{E}} S\_PO_{l}(\mathcal{S} \times \mathcal{A}_{n,\theta} \setminus \mathcal{E}, \varphi, \mathcal{O}) = S\_PO_{l}(n, \mathcal{S}, \varphi, \mathcal{O}) \leq S\_PO_{l}(\mathcal{S}, \varphi, \mathcal{O}).$$

Enfin, on note que l'algorithme se termine puisqu'il y a un nombre fini d'applications θ : $[n] \times S \to S$ et un nombre fini de transitions modales – donc de sous-ensembles de l'ensemble des transitions modales, ce qui termine la preuve de correction totale de l'algorithme. □

Puisqu'on ne réalise qu'un nombre exponentiel de boucles, et puisque le théorème 2, de complexité doublement exponentielle, est appelé à chaque tour de boucle, la complexité de l'algorithme total n'est pas aggravée et on en déduit le corollaire suivant.

Corollaire 3 (Complexité de l'algorithme de calcul d'opacité libérale dans le cas général)**.** *L'approximation du calcul de l'opacité libérale dans les spécifications modales est réalisable en un temps doublement exponentiel.*

L'algorithme 5 et le théorème 3 permettent de s'approcher de la généralisation du problème à toute spécification IDTMC. Cependant, on n'obtient qu'une borne inférieure de l'opacité libérale, puisqu'il est impossible de lister tous les ordonnanceurs de la spécification étudiée. La limitation vient du fait que l'on se limite aux ordonnanceurs à mémoire finie. Il semble judicieux de penser que l'on peut affiner la précision de la valeur approchée en considérant davantage d'ordonnanceurs, notamment en augmentant la valeur de la mémoire limite que l'on se fixe.

Dans la section suivante, on s'attache à prouver que l'opacité libérale est préservée par raffinement : cela prouve alors que l'opacité est une propriété de sécurité mesurable et capable d'être raffinée dans les specifications IDTMC.

5.4 Préservation de l'opacité libérale par raffinement

Les définitions et résultats des sections précédentes permettent la vérification de la sécurité dans les spécifications IDTMC. L'objet de la section suivante consiste à finalement résoudre le problème initial, grâce aux acquis accumulés. Le cœur du problème initial est de spécifier un système probabiliste, puis de l'améliorer continuellement en lui rajoutant des fonctionnalités et en affinant son comportement : c'est le processus de raffinement (*cf.* section 4.1).

Dans les sections précédentes, nous avons considéré plus précisément les systèmes probabilistes sécurisés. Vérifions à présent si le processus de raffinement d'un système probabiliste sécurisé permet effectivement d'améliorer – ou de conserver – le comportement de la spécification, notamment en ce qui concerne les propriétés d'opacité définies précédemment.

Théorème 4 (Préservation de l'opacité libérale par raffinement)**.** *Soient un secret ω-régulier φ et un observateur rationnel $\mathcal{O}$, soient deux spécifications IDTMC $\mathcal{S}_1$ et $\mathcal{S}_2$ telles que $\mathcal{S}_1$ raffine faiblement $\mathcal{S}_2$. Alors $S\_PO_l(\mathcal{S}_1, \varphi, \mathcal{O}) \leq S\_PO_l(\mathcal{S}_2, \varphi, \mathcal{O})$.*

Remarquons que prouver le théorème 4 pour le raffinement faible permet de le prouver également pour le raffinement fort. En effet, si $\mathcal{S}_1$ raffine fortement $\mathcal{S}_2$, alors elle la raffine faiblement d'après la proposition 5, et on peut appliquer le théorème.

La preuve du théorème 4 s'appuie sur le fait que si $\mathcal{S}_1$ raffine $\mathcal{S}_2$, alors $imp(\mathcal{S}_1) \subseteq imp(\mathcal{S}_2)$; cependant, cela n'implique pas que $S\_imp(\mathcal{S}_1) \subseteq S\_imp(\mathcal{S}_2)$ *a priori*. En effet, si A_1 est un

ordonnanceur de $\mathcal{S}_1$ qui produit une implémentation $\mathcal{S}_1(A_1) \in S\_imp(\mathcal{S}_1)$, alors c'est une implémentation de $\mathcal{S}_2$, mais il est possible que ce ne soit pas un ordonnancement. L'idée consiste alors à prouver que pour tout ordonnancement $\mathcal{S}_1(A_1)$, on peut trouver un ordonnancement $\mathcal{S}_2(A_2)$ qui est raffiné par $\mathcal{S}_1(A_1)$ (Bérard *et al.*, 2015b).

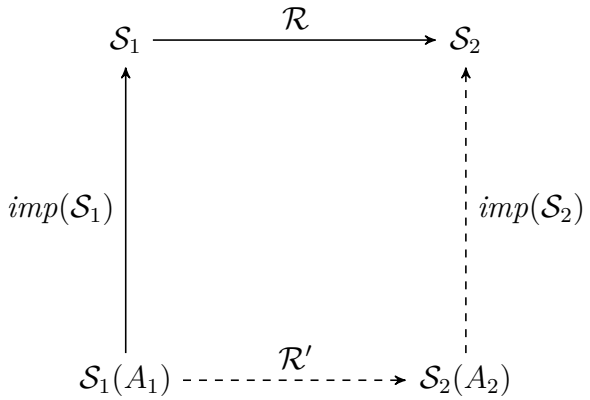


Figure 5.4 Illustration de la preuve du théorème 5

Théorème 5. *Soient deux spécifications IDTMC $\mathcal{S}_1$ et $\mathcal{S}_2$ telles que $\mathcal{S}_1$ raffine faiblement $\mathcal{S}_2$. Pour tout ordonnanceur A_1 de $\mathcal{S}_1$, il existe un ordonnanceur A_2 de $\mathcal{S}_2$ tel que $\mathcal{S}_1(A_1)$ raffine $\mathcal{S}_2(A_2)$.*

Quelques notations à propos des préfixes finis d'exécutions Avant de rédiger la preuve de ce théorème, définissons certaines notations.

Si $\rho \in FExec(\mathcal{S})$, alors $\text{lst}(\rho)$ désigne le dernier état visité par ρ.

Si $\mathcal{S}_1$ raffine faiblement $\mathcal{S}_2$ avec la relation $\mathcal{R}$, alors on définit la relation $\sim \subseteq FExec(\mathcal{S}_1) \times FExec(\mathcal{S}_2)$ telle que : $\rho_1 \sim \rho_2$ si, et seulement si, $|\rho_1| = |\rho_2|$ et pour chaque état intermédiaire, $s_{i,1} \mathcal{R} s_{i,2}$.

Pour tout $\rho_2 \in FExec(\mathcal{S}_2)$, on désigne par $sim(\rho_2) = \{\rho_1 \in FExec(\mathcal{S}_1) : \rho_1 \sim \rho_2\}$, et on définit la mesure μ_{ρ_2} telle que $\forall \rho_1 \in sim(\rho_2)$, $\mu_{\rho_2}(\rho_1) = \frac{\mathbf{P}(\rho_1)}{\mathbf{P}(sim(\rho_2))}$.

Démonstration. Soient deux IDTMC $\mathcal{S}_1 = (S_1, s_{0,1}, T_1, \Sigma, \lambda_1)$ et $\mathcal{S}_2 = (S_2, s_{0,2}, T_2, \Sigma, \lambda_2)$ telles que $\mathcal{S}_1$ raffine faiblement $\mathcal{S}_2$, et soit un ordonnanceur $A_1 \in Sched(\mathcal{S}_1)$. Notons $\mathcal{S}_1(A_1) = (Q_1, q_{0,1}, \Sigma, \Delta_1, L_1)$ le PTS ordonnancé par A_1.

L'idée de la preuve consiste à construire le bon ordonnanceur $A_2 \in Sched(\mathcal{S}_2)$.

Notons tout d'abord $\mathcal{R}$ la relation de raffinement entre $\mathcal{S}_1$ et $\mathcal{S}_2$.

Soit $\rho_2 \in FExec(\mathcal{S}_2)$ avec $\mathrm{lst}(\rho_2) = s_2$. Alors, pour tout $\rho_1 \in sim(\rho_2)$, $A_1(\rho_1) \in T_1(\mathrm{lst}(\rho_1))$ et $\mathrm{lst}(\rho_1)\mathcal{R}s_2$. $\mathcal{S}_1$ raffine faiblement $\mathcal{S}_2$, donc il existe $\delta_{\rho_1} : S_1 \to \mathcal{D}ist(S_2)$ telle que

$$\sum_{s'_1 \in S_1} A_1(\rho_1)\delta_{\rho_1}(s'_1) \in T_2(s_2).$$

On définit A_2 tel que :

$$\forall \rho_2 \in FExec(\mathcal{S}_2), A_2(\rho_2) = \sum_{\rho_1 \in sim(\rho_2)} \mu_{\rho_2}(\rho_1) \sum_{s'_1 \in S_1} A_1(\rho_1) \cdot \delta_{\rho_1}(s'_1) \in T_2(s_2).$$

On en déduit que A_2 est bien un ordonnanceur de $\mathcal{S}_2$. Montrons maintenant que l'ordonnancement obtenu est raffiné par $\mathcal{S}_1(A_1)$ par une relation $\mathcal{R}'$.

Dans la suite, on note $\mathcal{A}_1 = \mathcal{S}_1(A_1)$ et $\mathcal{A}_2 = \mathcal{S}_2(A_2)$.

Posons $\mathcal{R}' = \sim$ où $\sim$ est la relation de similitude entre deux préfixes finis d'exécutions. Puisque les états de $\mathcal{A}_1$ et de $\mathcal{A}_2$ sont des préfixes finis de leurs spécifications respectives, $\mathcal{R}'$ est bien une relation entre les états des PTS considérés. Montrons que c'est bien une relation de raffinement.

Soient $\rho_1 \in Q_1$ et $\rho_2 \in Q_2$ tels que $\rho_1 \mathcal{R}' \rho_2$.

Posons l'application $\delta' : Q_1 \to \mathcal{D}ist(Q_2)$ définie par :

$$\delta'(\nu_1)(\nu_2) = \begin{cases} \mu_{\rho_2}(\rho_1) \cdot \delta_{\rho_1}(\mathrm{lst}(\nu_1))(\mathrm{lst}(\nu_2)) & \text{si } \nu_1 \mathcal{R}' \nu_2, \\ 0 & \text{sinon.} \end{cases}$$

Alors, pour tout $\rho'_2 \in Q_2$, et en notant $\rho'_2 = \rho_2 \xrightarrow{A_2(\rho_2)} s'_2$,

$$\begin{array}{rcl} \sum_{\rho'_1 \in Q_1} \Delta_1(\rho_1)(\rho'_1) \cdot \delta'(\rho'_1)(\rho'_2) & = & \sum_{s'_1 \in S_1} \Delta_1(\rho_1)(\rho_1 s'_1) \cdot \delta'(\rho_1 s'_1)(\rho'_2) \\ & = & \sum_{s'_1 \in S_1} A_1(\rho_1)(s'_1) \cdot \delta'(\rho_1 \xrightarrow{A_1(\rho_1)} s'_1)(\rho'_2) \\ & = & \sum_{s'_1 \in S_1} A_1(\rho_1)(s'_1) \cdot \mu_{\rho_2}(\rho_1) \cdot \delta_{\rho_1}(s'_1)(s'_2) \\ & = & \sum_{\rho_1 \in sim(\rho_2)} \sum_{s'_1 \in S_1} A_1(\rho_1)(s'_1) \cdot \mu_{\rho_2}(\rho_1) \cdot \delta_{\rho_1}(s'_1)(s'_2) \\ & = & A_2(\rho_2)(s'_2) \\ & = & \Delta_2(\rho_2)(\rho'_2). \end{array}$$

□

Cette partie technique prouvée, il reste à utiliser ceci pour prouver le théorème 4.

Démonstration du théorème 4. Soit $\mathcal{A}_1 \in S\_imp(\mathcal{S}_1)$. D'après le théorème 5, il existe un ordonnancement $\mathcal{A}_2 \in S\_imp(\mathcal{S}_2)$ tel que $\mathcal{A}_1 \preceq \mathcal{A}_2$. Puisque ce sont des PTS, d'après la proposition 7, $\mathcal{A}_2 \preceq \mathcal{A}_1$, et comme dans la discussion autour de la proposition 8, cela implique que leurs langages sont égaux. D'après Bérard *et al.* (2015b), on en déduit également que $\mathbf{P}_{\mathcal{A}_1}(w) = \mathbf{P}_{\mathcal{A}_2}(w)$ pour tout préfixe fini appartenant au langage en commun. Donc, pour tout langage ω-régulier L, $\mathbf{P}_{\mathcal{A}_1}(L) = \mathbf{P}_{\mathcal{A}_2}(L)$. C'est notamment le cas pour le langage ω-régulier $\mathcal{V}(\mathcal{A}_1, \varphi, \mathcal{O}) = \mathcal{V}(\mathcal{A}_2, \varphi, \mathcal{O})$.

On déduit de ce raisonnement que, pour tout ordonnancement $\mathcal{A}_1 \in S\_imp\mathcal{S}_1$, il existe un ordonnancement $\mathcal{A}_2 \in S\_imp(\mathcal{S}_2)$ tel que $PO_l(\mathcal{A}_1, \varphi, \mathcal{O}) = PO_l(\mathcal{A}_1, \varphi, \mathcal{O})$. Par conséquent, $S\_PO_l(\mathcal{S}_1, \varphi, \mathcal{O}) \leq S\_PO_l(\mathcal{S}_2, \varphi, \mathcal{O})$, puisqu'il peut rester des ordonnancements de $\mathcal{S}_2$ qui ne sont raffinés par aucun ordonnancement de $\mathcal{S}_1$. □

Cette section a présenté l'un des résultats principaux du mémoire, en montrant que l'opacité libérale dans une spécification IDTMC était préservée par raffinement faible, ce qui étend le résultat déjà prouvé par Bérard *et al.* (2015b), qui se limitaient aux raffinements forts.

Cependant, seule l'opacité libérale a été étudiée dans les sections précédentes. Nous proposons dans la section suivante de définir d'autres formes d'opacité pour les spécifications, et d'essayer d'étendre le théorème 4 à celles-ci.

5.5 Cas des autres opacités

Dans le chapitre 3, plusieurs types d'opacité ont été définies sur les PTS. Nous avons étudié plus en détail l'extension de l'opacité libérale aux IDTMC ; définissons à présent l'extension de la quasi-opacité uniforme ainsi que celle de l'opacité restrictive aux spécifications.

5.5.1 Quasi-opacité uniforme

Dans cette section, on généralise la notion de quasi-opacité uniforme aux spécifications IDTMC. De façon analogue à la généralisation de l'opacité libérale, le but est de déterminer la quasi-opacité du secret sur chacune des implémentations ordonnancées afin de trouver le pire cas pour la spécification. Puisque la quasi-opacité uniforme est une valeur binaire, la définition est légèrement différente.

En utilisant le même raisonnement que pour la définition 11, on restreint l'espace de raisonnement aux langages de la forme L_ω, avec $L \subseteq \Sigma^*$.

Définition 24 (Quasi-opacité uniforme dans une IDTMC)**.** *Soit L un langage de Σ^*. Soient une IDTMC $\mathcal{S}$ dont le langage des exécutions est L, un secret régulier $\varphi \subseteq L$, un observateur rationnel $\mathcal{O}$, et un seuil $\theta \in [0,1]$.*

Alors le secret φ_ω est uniformément quasi-opaque *ou* θ-opaque *dans $\mathcal{S}_\omega$ relativement à $\mathcal{O}$ si, et seulement si, il est uniformément quasi-opaque dans toute implémentation ordonnancée $\mathcal{A} \in S\_imp(\mathcal{S}_\omega)$ relativement à $\mathcal{O}$, c'est-à-dire,*

$$\forall k \in \mathbb{N}, \sup_{\mathcal{A} \in S\_imp(\mathcal{S}_\omega)} \mathbf{P}\Big((L_C \cap \Sigma^k)_\omega\Big) \leq \theta.$$

Le problème consiste ainsi à déterminer, pour tout k entier, la valeur suivante, avec les mêmes notations que dans la définition 11 :

$$\sup_{\mathcal{A} \in S\_imp(\mathcal{S}_\omega)} \mathbf{P}\Big((L_C \cap \Sigma^k)_\omega\Big).$$

Puisque, pour k fixé, le langage $(L_C \cap \Sigma^k)_\omega$ est ω-régulier, ce problème est tout à fait analogue au problème de l'opacité libérale. Il est ainsi possible d'adapter les résultats des théorèmes 2 et 3 afin d'obtenir en des temps doublement exponentiels les valeurs de ces bornes supérieures, pour une valeur k fixée.

Notons que la discussion à propos de la modalité de l'IDTMC considérée tient toujours : le langage L_C peut prendre plusieurs valeurs distinctes si on considère une spécification modale selon l'ordonnancement, ce qui rend plus complexe le calcul.

Le calcul pour un k fixé ne suffit cependant pas à décider de l'uniforme quasi-opacité du secret. En effet, la définition demande le calcul pour tout entier k naturel, ce qui rend le problème *a priori* indécidable en temps fini, à moins de prouver que l'on peut se limiter à un nombre fini d'entiers k (Saboori et Hadjicostis, 2014). En d'autres termes, le problème de décidabilité de la quasi-opacité uniforme reste ouvert.

En revanche, on remarque que la preuve du théorème 4 tient pour toute probabilité d'ensemble ω-régulier, et pas uniquement pour le calcul de l'opacité libérale. Par exemple, pour le cas de la quasi-opacité uniforme, si l'on suppose que l'on est capable de calculer une telle valeur, on peut traduire le théorème 4 comme suit.

Théorème 6. *Soient un secret ω-régulier φ et un observateur rationnel $\mathcal{O}$, soient deux spécifications IDTMC $\mathcal{S}_1$ et $\mathcal{S}_2$ telles que $\mathcal{S}_1$ raffine complètement $\mathcal{S}_2$, et soit un seuil $\theta \in [0,1]$. Alors, si φ est uniformément θ-opaque dans $\mathcal{S}_1$ relativement à $\mathcal{O}$, alors il est uniformément θ-opaque dans $\mathcal{S}_2$ relativement à $\mathcal{O}$.*

Autrement dit, la quasi-opacité uniforme est également préservée par raffinement dans les IDTMC.

5.5.2 Opacité restrictive

Valeur duale de l'opacité libérale, l'opacité restrictive a une définition bien plus complexe que la première. Elle s'avère plus compliquée à manipuler, mais l'on peut tout de même définir cette mesure pour les IDTMC.

Dans le même esprit que pour les autres opacités, on généralise la notion d'opacité restrictive aux spécifications IDTMC : on définit l'opacité restrictive d'une spécification comme sa valeur dans le pire des cas. Puisque la fonction d'opacité restrictive décroît en fonction du risque de découvrir le secret, le pire cas est atteint lorsque l'opacité restrictive est minimale (*cf.* proposition 2).

Définition 25 (Opacité restrictive dans une IDTMC)**.** *Soit L un langage de Σ^*. Soient une IDTMC $\mathcal{S}$ dont le langage des exécutions est L, un secret régulier $\varphi \subseteq L$ et un observateur rationnel $\mathcal{O}$. L'*opacité restrictive de φ dans $\mathcal{S}$ relativement à $\mathcal{O}$ *est la grandeur définie par :*

$$S\_PO_r(\mathcal{S}, \varphi, \mathcal{O}) = \inf_{\mathcal{A} \in S\_imp(\mathcal{S})} PO_r(\mathcal{A}, \varphi, \mathcal{O}).$$

Cependant, puisque la définition de l'opacité restrictive dans un PTS n'est pas linéaire, et puisque l'opacité restrictive n'est la probabilité d'aucun langage régulier *a priori*, le calcul de cette borne inférieure est très complexe en pratique.

Ce chapitre a permis de prouver les résultats théoriques importants que sont :

- le calcul de l'opacité libérale dans une spécification non-modale est réalisable en un temps doublement exponentiel ;
- le calcul de l'opacité libérale dans une spécification modale est approximable par la borne inférieure en un temps doublement exponentiel ;
- toute probabilité d'ensemble ω-régulier, telle que l'opacité libérale ou la quasi-opacité uniforme, est préservée par raffinement faible de spécification IDTMC.

Les algorithmes de calcul d'opacité ont été énoncés et prouvés, dans le but notamment de pouvoir les appliquer dans le chapitre 6 : celui-ci consiste en l'étude d'un cas concret de système sécurisé. Nous considérons un exemple de cas qu'un expert responsable sécurité dans une entreprise peut recevoir pour étude, et nous établissons notre méthode de raffinement sur cet exemple.

CHAPITRE 6 ÉTUDE DE CAS

Les sections 4 et 5 ont introduit un ensemble d'aspects théoriques et ont produit d'importants résultats de la théorie du raffinement de systèmes sécurisés. Cette section applique ces résultats et algorithmes sur un système abstrait. On réalise ici l'ensemble du processus de raffinement, en partant d'un système réel que l'on veut sécuriser. On commence par sa modélisation, le calcul de sa sécurité, puis on réalise un raffinement afin d'améliorer le modèle. Ce cas est inspiré des modèles d'authentification de Biondi *et al.* (2014).

6.1 Description de l'étude de cas

Cette section décrit le problème que l'on considère. On explique tout d'abord les caractéristiques du système existant, puis on le modélise sous forme de spécification IDTMC. Suite à cela, on présente les requis de sécurité que l'on veut assurer, en les modélisant sous forme de propriété d'opacité.

6.1.1 Modélisation du système

Considérons une base de données médicales. La confidentialité critique des données considérées implique que seul un panel restreint d'utilisateurs a accès à la base. Cependant, elle doit pouvoir être consultée à tout moment depuis plusieurs endroits distincts (médecins traitants, hôpitaux, cliniques), sans contrainte de temps ou d'espace. Ainsi, le serveur de base de données est relié aux internets : un système de contrôle d'accès est nécessaire afin d'assurer la confidentialité. Un simple système de vérification de couple *utilisateur/mot de passe* est envisagé dans un premier temps. On suppose pour notre étude formelle que les aspects cryptographiques sont parfaits. Ainsi, on peut représenter le système par l'IDTMC de la figure 6.1, sur l'alphabet de propriétés atomiques exclusives $\Sigma = \{a, b, c, d, e\}$.

Dans l'état q_0, le système attend l'entrée du nom d'utilisateur. Le système est dans un état d'attente, hors de la zone privée : cela correspond à la propriété a. Si le nom d'utilisateur n'est pas reconnu, ce qui correspond à la propriété d, la requête est rejetée (propriété e) et le système passe dans l'état déconnecté q_4.

Si en revanche l'utilisateur est connu et possède *a priori* les droits requis à la consultation de la base de données, le système attend l'entrée du mot de passe associé dans l'état q_1. Cette attente correspond à la propriété b, vérifiée tant que le mot de passe est incorrect ou inexistant. Dès que le bon mot de passe est entré, le système garantit l'accès à la base de

données (propriété c). Si le mot de passe est incorrect, l'accès est éventuellement refusé et le système passe dans l'état déconnecté q_4.

La spécification autorise l'implémentation d'un mécanisme permettant d'essayer une nouvelle fois d'entrer un mot de passe en cas d'erreur. Ceci est représenté par la transition qui boucle sur l'état q_1.

La vérification du mot de passe est sous-spécifiée (état q_1), en ce sens que les intervalles de probabilités de transitions issues de cet état n'ajoutent aucune contrainte particulière. L'intérêt ici est uniquement de traduire le flux des événements lors du contrôle d'accès, première étape de la conception de ce système.

On impose uniquement des contraintes en sortie de l'état q_1, lors de la vérification du nom d'utilisateur. Ces probabilités dépendent du nombre d'utilisateurs possédant les droits de consultation. La base est nécessairement utilisée par quelqu'un, donc ce nombre est nécessairement non-nul ; de plus, puisque la base est reliée aux internets, il faut considérer des utilisateurs qui ne sont pas enregistrés, nécessairement présents également. Ces deux raisonnements permettent de justifier la non-modalité des transitions $q_0 \to q_1$ et $q_0 \to q_2$.

En pratique, les contraintes permettent de considérer divers scénarios d'implémentation, suivant le nombre d'utilisateurs autorisés, la robustesse de leur mot de passe, et le degré de connaissance d'éventuels utilisateurs malhonnêtes.

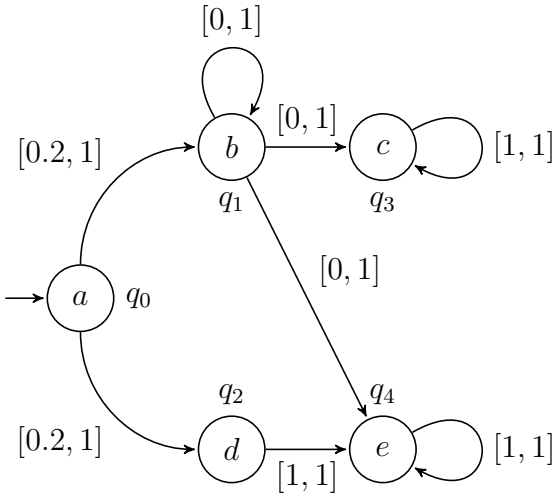


Figure 6.1 Un système de contrôle d'accès à une base de données médicales $\mathcal{S}_2$

6.1.2 Requis de sécurité

Tel qu'indiqué précédemment, on ne modélise pas les aspects cryptographiques : le choix du procédé de chiffrement utilisé par la suite n'est pas le propos ici. La seule hypothèse que l'on réalise concernant ces questions est celle selon laquelle la tentative de mot de passe entrée par l'utilisateur reste secrète et cachée de l'environnement extérieur. On modélise ceci par la fonction d'observation rationnelle suivante.

$$\begin{cases} \mathcal{O}(\sigma) & = & \sigma & \text{si } \sigma \neq b \\ \mathcal{O}(b) & = & \varepsilon \end{cases}$$

On affirme alors que, malgré une infrastructure cryptographique parfaite, un tel système ne garantit pas nécessairement la confidentialité des données médicales. Notamment, ce système ne garantit pas une protection suffisante contre la création de canaux cachés de communication entre un attaquant et les données confidentielles.

À titre d'exemple, supposons qu'un attaquant extérieur, passif, soit capable de distinguer les connexions à la base de données après plusieurs essais de mots de passe (abb^+c^ω), de celles obtenues dès le premier essai (abc^ω). Il représente alors chaque exécution du premier groupe par le bit 0 (par exemple), et chaque exécution du second groupe par le bit 1. De cette manière, un complice possédant les droits de consultation de la base de données peut se servir de ce canal caché pour transmettre des informations à l'attaquant, uniquement en se connectant successivement de l'une ou l'autre des manières, tel que décrit sur la figure 6.2.

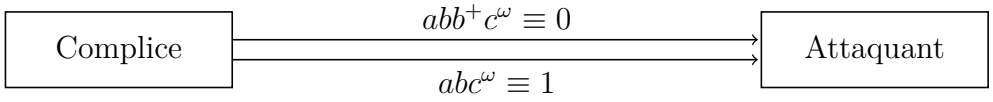


Figure 6.2 Canal caché de communication entre le complice et l'attaquant

Une telle possibilité de fuite d'information est à éviter. Pour cela, le but est d'empêcher la distinction des deux groupes d'exécutions ; autrement dit, on requiert l'opacité du prédicat

$$\varphi = abc^\omega$$

dans la spécification considérée, relativement à l'observateur rationnel $\mathcal{O}$ défini précédemment.

6.2 Calcul de l'opacité binaire

Vérifions dans un premier temps l'opacité au sens de la définition 7 dans la spécification $\mathcal{S}_2$. Pour cela, calculons les différents langages à considérer.

$$\begin{array}{lcllcl} L_2 & = & ab^+c^\omega + ab^+e^\omega + ade^\omega & \mathcal{O}(L_2 \setminus \varphi) & = & ac^\omega + ae^\omega + ade^\omega \\ L_2 \setminus \varphi & = & abb^+c^\omega + ab^+e^\omega + ade^\omega & \mathcal{O}(\varphi) & = & ac^\omega \end{array}$$

On en déduit que $\mathcal{O}(\varphi) \subseteq \mathcal{O}(L_2 \setminus \varphi)$, donc *a priori*, le secret serait opaque dans la spécification – c'est-à-dire qu'il est opaque dans toute implémentation de la spécification. Cependant, le calcul a été réalisé en supposant que le langage L_2 de $\mathcal{S}_2$ était le même que toutes ses implémentations, ce qui est faux puisque l'IDTMC possède des transitions modales (*cf.* discussion autour de la définition 18).

Notamment, le PTS représenté figure 6.3 est une implémentation de la spécification, mais le calcul de l'opacité est bien différent. On note L'_2 son langage.

$$\begin{array}{lcllcl} L'_2 & = & abc^\omega + abe^\omega + ade^\omega & \mathcal{O}(L'_2 \setminus \varphi) & = & ae^\omega + ade^\omega \\ L'_2 \setminus \varphi & = & abe^\omega + ade^\omega & \mathcal{O}(\varphi) & = & ac^\omega \end{array}$$

Dans ce cas-ci, $\mathcal{O}(\varphi) \not\subseteq \mathcal{O}(L'_2 \setminus \varphi)$, donc le secret n'est pas opaque dans l'implémentation. Ainsi, il existe des implémentations de la spécification dans lesquelles le secret n'est pas opaque relativement à $\mathcal{O}$. Puisque la sécurité d'un système se vérifie toujours au pire cas, on en déduit que le secret n'est pas opaque dans la spécification $\mathcal{S}_2$. Le but est désormais d'améliorer cette mesure, en calculant l'opacité libérale du secret dans la spécification.

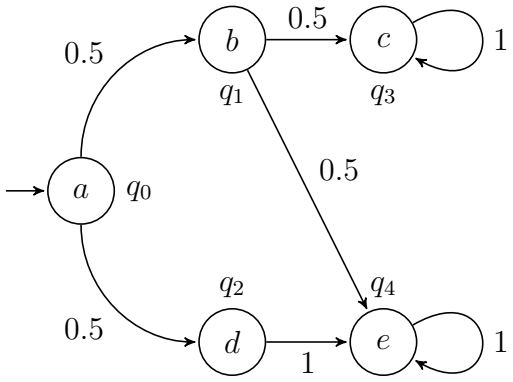


Figure 6.3 Une implémentation de $\mathcal{S}_2$

6.3 Opacité libérale

Afin de calculer l'opacité libérale de φ dans $\mathcal{S}_2$ relativement à $\mathcal{O}$, on applique le théorème 3 puisque l'IDTMC est modale. Par conséquent, on ne pourra pas obtenir la valeur de l'opacité libérale, mais uniquement une approximation par valeur inférieure.

L'algorithme dont le théorème 3 prouve la correction demande comme paramètres d'entrée un entier naturel non-nul n (correspondant à la taille de la mémoire des ordonnanceurs considérés), ainsi que $\mathcal{S}_2$, φ et $\mathcal{O}$. Afin d'illustrer nos calculs, posons par exemple $n = 1$: on considère alors uniquement les ordonnanceurs sans-mémoire.

La première étape de l'algorithme consiste à énumérer tous les systèmes de transitions $\mathcal{A}_{n,\theta}$, avec $\theta : [n] \times S_2 \to [n]$. Puisque $n = 1$, il existe une unique fonction de transition de modes θ, notée θ_1. Ainsi, il existe un unique système de transitions $\mathcal{A}_{1,\theta_1}$, représenté figure 6.4. Il s'ensuit qu'il existe une unique spécification produit $\mathcal{S}_2 \times \mathcal{A}_{1,\theta_1}$, qui est égale à $\mathcal{S}_2$. Autrement dit, un ordonnanceur sans-mémoire laisse inchangée la forme globale de l'IDTMC.

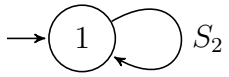


Figure 6.4 Système de transitions $\mathcal{A}_{1,\theta_1}$ – résultat du dépliage d'un ordonnanceur sans-mémoire

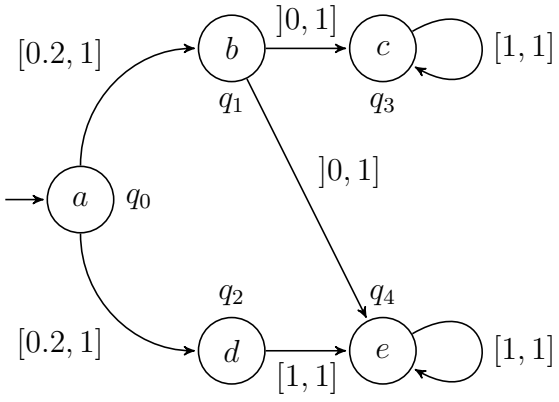


Figure 6.5 Résultat de la seconde étape – $\mathcal{S}_2 \times \mathcal{A}_{1,\theta_1} \setminus \mathcal{E}$

La seconde étape consiste à calculer l'ensemble des transitions modales de la spécification

produit $\mathcal{S}_2 \times \mathcal{A}_{1,\theta_1}$. Ici,

$$\mathcal{E}_m(\mathcal{S}_2 \times \mathcal{A}_{1,\theta_1}) = \{(q_1, q_1), (q_1, q_3), (q_1, q_4)\}.$$

Une fois ceci réalisé, il s'agit d'énumérer les sous-parties de cet ensemble, puis de construire, pour chaque sous-partie $\mathcal{E}$, la spécification non-modale $\mathcal{S}_2 \times \mathcal{A}_{1,\theta_1} \setminus \mathcal{E}$. Par exemple, pour $\mathcal{E} = \{(q_1, q_1)\}$, on construit la spécification représentée sur la figure 6.5.

Cet objet étant effectivement une spécification IDTMC, on peut passer à l'étape suivante et appliquer le théorème 2 sur $\mathcal{S}_2 \times \mathcal{A}_{1,\theta_1} \setminus \mathcal{E}$.

6.3.1 Application du théorème 2

Tout d'abord, il s'agit de construire l'automate de parité déterministe $\mathcal{A}_\mathcal{V}$ correspondant à l'ensemble $\mathcal{V}(\mathcal{S}_2 \times \mathcal{A}_{1,\theta_1} \setminus \mathcal{E}, \varphi, \mathcal{O})$. Le langage de la spécification ici est $L_{2,a} = abc^\omega + abe^\omega + ade^\omega$, de sorte que l'ensemble considéré se calcule de la manière suivante.

$$\begin{aligned}
\mathcal{O}(L_{2,a} \setminus \varphi) &= ae^\omega + ade^\omega \\
\text{donc } \mathcal{O}^{-1}(\mathcal{O}(L_{2,a} \setminus \varphi)) &= abe^\omega + ade^\omega \\
\text{donc } \overline{\mathcal{O}^{-1}(\mathcal{O}(L_{2,a} \setminus \varphi))} &= L_{2,a} \setminus (abe^\omega + ade^\omega) \\
&= abc^\omega \\
\text{donc } \mathcal{V}(\mathcal{S}_2 \times \mathcal{A}_{1,\theta_1} \setminus \mathcal{E}, \varphi, \mathcal{O}) &= \varphi \cap \overline{\mathcal{O}^{-1}(\mathcal{O}(L_{2,a} \setminus \varphi))} \\
&= abc^\omega \cap abc^\omega = abc^\omega = \varphi.
\end{aligned}$$

On en déduit le DPA représenté sur la figure 6.6.

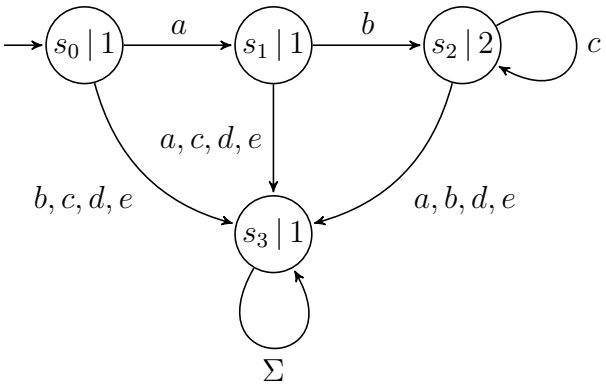


Figure 6.6 Construction du DPA $\mathcal{A}_\mathcal{V}$

La prochaine étape consiste à réaliser le produit synchronisé de l'IDTMC non-modale étudiée

avec le DPA précédent. On obtient l'IDTMC non-modale colorée

$$\mathcal{S}_{\mathcal{V}} = \left(\mathcal{S}_2 \times \mathcal{A}_{1,\theta_1} \setminus \mathcal{E}\right) \otimes \mathcal{A}_{\mathcal{V}}$$

représentée sur la figure 6.7.

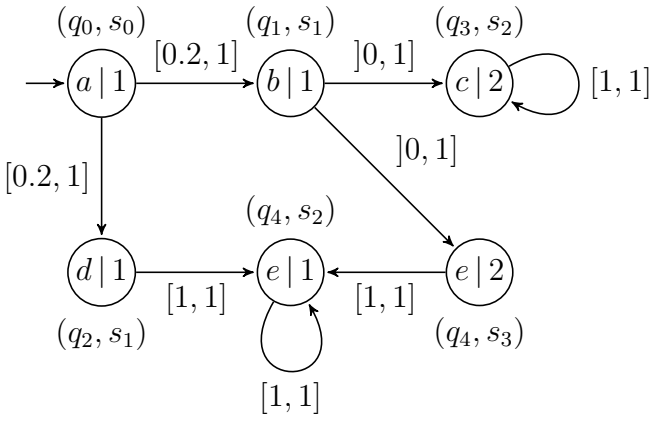


Figure 6.7 Application du théorème – $\mathcal{S}_{\mathcal{V}} = \left(\mathcal{S}_2 \times \mathcal{A}_{1,\theta_1} \setminus \mathcal{E}\right) \otimes \mathcal{A}_{\mathcal{V}}$

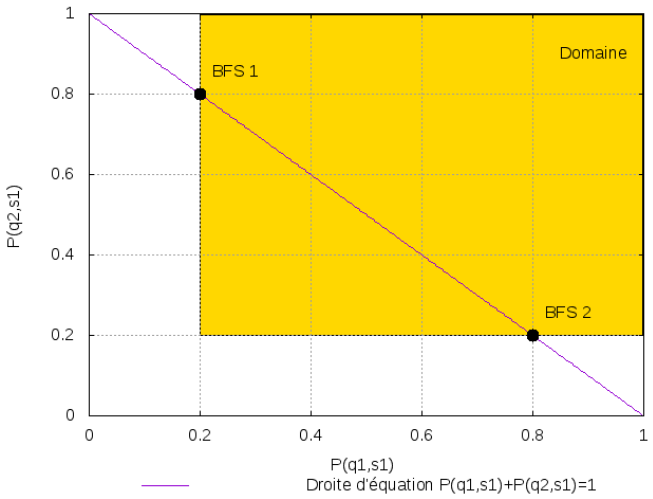


Figure 6.8 Illustration du problème au sommet (q_0, s_0)

Il s'agit à présent de considérer les différents systèmes d'équations linéaires aux différents sommets du graphe, tel qu'expliqué dans la preuve du théorème 2. Par exemple, dans l'état initial (q_0, s_0), le problème linéaire à résoudre est le suivant, représenté graphiquement sur la

figure 6.8.

$$\begin{cases} \mathbf{P}\Big((q_1, s_1)\Big) + \mathbf{P}\Big((q_2, s_1)\Big) &= 1 \\ \mathbf{P}\Big((q_1, s_1)\Big) \in [0.2, 1] \\ \mathbf{P}\Big((q_2, s_1)\Big) \in [0.2, 1] \end{cases}$$

Sur la figure, l'ensemble des solutions du problème est représenté par l'intersection entre la droite d'équation $\mathbf{P}\Big((q_1, s_1)\Big) + \mathbf{P}\Big((q_2, s_1)\Big) = 1$ et le domaine représenté en jaune. En s'inspirant de l'explication du théorème 2, on peut définir les BFS suivantes : $(0.2, 0.8)$ et $(0.8, 0.2)$.

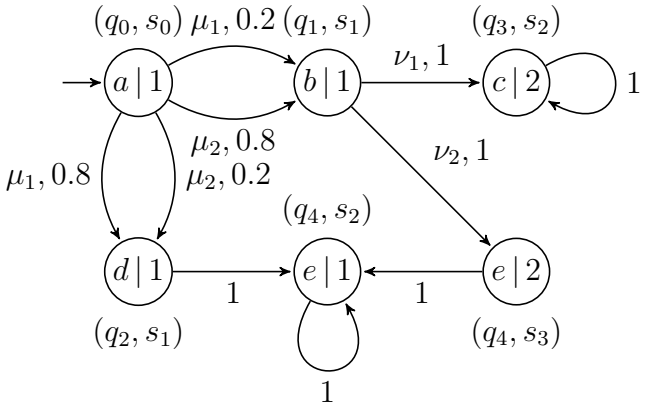


Figure 6.9 Application du théorème – Transformation de $\mathcal{S}_\mathcal{V}$ en son MDP $\mathcal{M}_\mathcal{V}$

Il reste à réitérer cette procédure pour l'ensemble des états de $\mathcal{S}_\mathcal{V}$ afin de construire le MDP correspondant. On rappelle que celui-ci est construit en donnant les différentes BFS en choix de distributions pour chaque état, tel qu'illustré sur la figure 6.9. Notamment, pour l'état initial, on retrouve les distributions μ_1 et μ_2 qui correspondent respectivement aux BFS $(0.2, 0.8)$ et $(0.8, 0.2)$ obtenues précédemment.

Il reste alors à calculer la probabilité maximale du langage abc^ω dans le MDP $\mathcal{M}_\mathcal{V}$, qui est nécessairement atteinte pour une implémentation telle que, pour chaque état, la distribution choisie est l'une des distributions obtenues par BFS. Ainsi, l'implémentation qui donne la probabilité maximale a nécessairement choisi μ_1 ou μ_2 comme distribution à l'état initial – et non une combinaison linéaire quelconque. On obtient finalement un nombre fini d'implémentations à considérer, appelées implémentations extrémales. Il est facile d'extraire la probabilité maximale recherchée.

Ici, on distingue quatre implémentations, suivant si on choisit μ_1 ou μ_2, puis ν_1 ou ν_2. Les résultats sont reportés sur le tableau 6.1 : on obtient que la probabilité maximale de lire abc^ω dans $\mathcal{M}_\mathcal{V}$ est 0.8.

Tableau 6.1 Probabilités du langage abc^ω dans les implémentations extrémales de $\mathcal{M}_\mathcal{V}$

Choix de distribution	μ_1	μ_2
ν_1	0.2	0
ν_2	0.8	0

On en déduit, d'après le théorème 2, que $S\_PO_l(1, \mathcal{S}_\mathcal{V}, \varphi, \mathcal{O}) = 0.8$.

6.3.2 Bilan du calcul

On a réalisé ici un seul passage dans l'algorithme du théorème 3. Pour $n = 1$, on a considéré l'unique système de transitions $\mathcal{A}_{1,\theta_1}$; puis, pour ce système, on a étudié une seule sous-spécification non-modale de l'IDTMC dépliée par $\mathcal{A}_{1,\theta_1}$. Puis, on a calculé la probabilité maximale recherchée dans cette sous-spécification particulière. Afin de terminer l'étude pour $n = 1$, il reste à considérer les autres cas de sous-spécifications non-modales – obtenues à partir des autres sous-ensembles de $\mathcal{E}_m$. Cela terminera l'algorithme, puisque $\mathcal{A}_{1,\theta_1}$ est l'unique système de transitions pour $n = 1$.

Les résultats des probabilités maximales pour les différents sous-ensembles de transitions modales sont reportés sur le tableau 6.2.

Tableau 6.2 Résultats de l'algorithme suivant le sous-ensemble de transitions modales choisi

Choix de $\mathcal{E} \subseteq \mathcal{E}_m$	$\{(q_1, q_1)\}$	$\{(q_1, q_3)\}$	$\{(q_1, q_4)\}$	$\emptyset$
Rés. intermédiaires	0.8	0	0	0
Choix de $\mathcal{E} \subseteq \mathcal{E}_m$	$\{(q_1, q_1), (q_1, q_3)\}$	$\{(q_1, q_1), (q_1, q_4)\}$	$\{(q_1, q_3), (q_1, q_4)\}$	$\mathcal{E}_m$
Rés. intermédiaires	0	0.8	0	0

Ainsi, d'après le théorème 3 et la post-condition de l'algorithme correspondant, on en déduit que

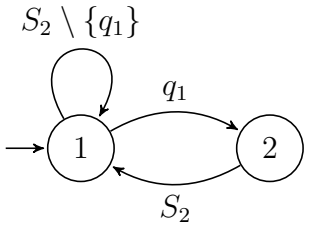
$$S\_PO_{l1}(\mathcal{S}_2, \varphi, \mathcal{O}) \leq S\_PO_l(1, \mathcal{S}_2, \varphi, \mathcal{O}) = 0.8 \leq S\_PO_l(\mathcal{S}_2, \varphi, \mathcal{O}).$$

On déduit notamment le fait que le secret φ n'est pas opaque dans $\mathcal{S}_2$ relativement à $\mathcal{O}$, puisque $S\_PO_l(\mathcal{S}_2, \varphi, \mathcal{O}) > 0$, ce qui est cohérent avec la discussion de la partie 6.2.

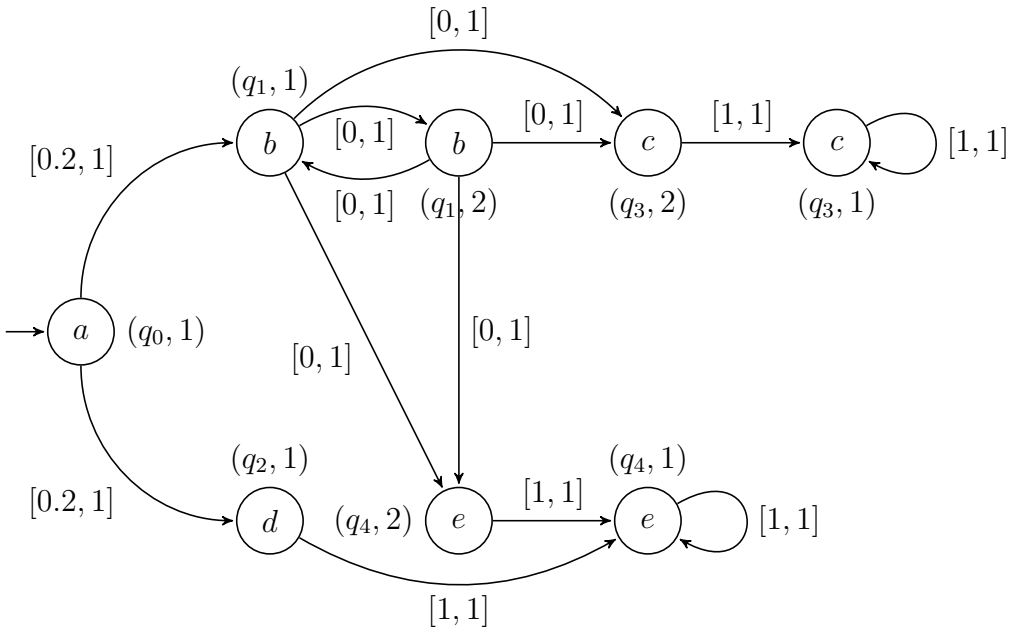
6.3.3 Un autre exemple

Illustrons à présent l'algorithme pour $n = 2$. Dans ce cas, il existe plusieurs systèmes de transitions à deux modes, il faut donc tous les étudier. Pour les besoins de l'illustration,

détaillons l'étude dans le cas de la fonction de transition de mode θ_2, produisant le système de transitions $\mathcal{A}_{2,\theta_2}$ représenté sur la figure 6.10a. La spécification représentée sur la figure 6.10b est issue du dépliage de l'ordonnancement de $\mathcal{S}_2$ par un ordonnanceur dont la transition de modes est régie par $\mathcal{A}_{2,\theta_2}$. En substance, cet adversaire dédouble l'état q_1 vérifiant la propriété atomique b, état central dans notre problème.



(a) Système de transitions de modes d'un ordonnanceur à mémoire finie $\mathcal{A}_{2,\theta_2}$



(b) Première étape de l'ordonnancement – $\mathcal{S}_2 \times \mathcal{A}_{2,\theta_2}$

Figure 6.10 Application des étapes du théorème – cas d'un ordonnanceur de mémoire 2

L'ensemble des transitions modales est

$$\mathcal{E}_m = \begin{array}{l}\Big\{\Big((q_1,1),(q_1,2)\Big),\Big((q_1,1),(q_3,2)\Big),\Big((q_1,1),(q_4,2)\Big),\\ \Big((q_1,2),(q_1,1)\Big),\Big((q_1,2),(q_3,2)\Big),\Big((q_1,2),(q_4,2)\Big)\Big\}.\end{array}$$

Dans le cas $n = 1$, seuls deux sous-ensembles de transitions modales parmi huit donnent des résultats de probabilité maximale non-nuls. Cela s'explique par le fait que, pour les autres

cas, le langage $\mathcal{V}(\mathcal{S}_2 \times \mathcal{A}_{1,\theta_1} \setminus \mathcal{E}, \varphi, \mathcal{O})$ est vide, autrement dit, le prédicat φ est opaque dans la spécification modale correspondante. Ainsi, afin de limiter les calculs, une bonne méthode consiste à éliminer tous les cas qui préservent l'opacité. Dans la suite, on note $\mathcal{E} \subseteq \mathcal{E}_m$ le sous-ensemble de transitions modales que l'on décide d'annuler, et $\mathcal{S}_{nm} = \mathcal{S}_2 \times \mathcal{A}_{2,\theta_2} \setminus \mathcal{E}$ la spécification non-modale obtenue.

— Supposons que $\Big((q_1, 1), (q_3, 2)\Big) \in \mathcal{E}$ – c'est-à-dire que la transition $\Big((q_1, 1), (q_3, 2)\Big)$ est implémentée avec une probabilité nulle. Alors le langage L de la spécification non-modale $\mathcal{S}_{nm}$ ne contient pas abc^ω, donc $\varphi \cap L = \emptyset$. Autrement dit, aucune exécution de $\mathcal{S}_{nm}$ n'appartient au secret, celui-ci est donc évidemment opaque.

— Supposons que $\Big((q_1, 1), (q_1, 2)\Big) \notin \mathcal{E}$ et $\Big((q_1, 2), (q_3, 2)\Big) \notin \mathcal{E}$. Alors le langage L contient $abbc^\omega$ et $abbc^\omega \notin \varphi$, donc

$$\mathcal{O}(abbc^\omega) = ac^\omega \in \mathcal{O}(L \setminus \varphi).$$

Or $\mathcal{O}(\varphi \cap L) \subseteq ac^\omega$, suivant si $\varphi \cap L$ est vide ou non. On en déduit

$$\mathcal{O}(\varphi \cap L) \subseteq \mathcal{O}(L \setminus \varphi).$$

Autrement dit, le secret est opaque dans $\mathcal{S}_{nm}$.

— Supposons enfin que $\mathcal{E}$ ne vérifie pas les deux conditions précédentes. Alors $\varphi \cap L$ est non-vide et L ne contient pas le langage abb^+c^ω. Autrement dit, on obtient à nouveau le cas de non-opacité discuté dans la section 6.2. On peut alors réitérer la procédure décrite pour le cas $n = 1$, et ce pour chaque sous-ensemble $\mathcal{E}$ restant.

Ce raisonnement permet d'éviter de rentrer dans des boucles inutiles. On résume ce paragraphe par le tableau 6.3, contenant également les résultats intermédiaires pour chaque sous-ensemble. Rappelons que ces résultats ne concernent que la fonction de transition de modes θ_2. Il est nécessaire de réitérer les étapes précédentes pour chaque fonction de transition de modes à mémoire finie égale à deux.

Tableau 6.3 Résultats suivant le sous-ensemble de transitions modales choisi, pour $\mathcal{A}_{2,\theta_2}$

Conditions sur $\mathcal{E} \subseteq \mathcal{E}_m$	**Résultats intermédiaires**
Si $\Big((q_1, 1), (q_3, 2)\Big) \in \mathcal{E}$	0
Si $\Big((q_1, 1), (q_1, 2)\Big) \notin \mathcal{E}$ et $\Big((q_1, 2), (q_3, 2)\Big) \notin \mathcal{E}$	0
Sinon	0.8

Les résultats pour les autres fonctions de transitions de modes à mémoire finie égale à deux

donnent comme résultat d'algorithme

$$S\_PO_l(2, \mathcal{S}_2, \varphi, \mathcal{O}) = 0.8.$$

On déduit de ce résultat et de celui obtenu pour $n = 1$ que le requis de sécurité exprimé dans la section 6.1.2 n'est pas vérifié pour la spécification $\mathcal{S}_2$. Plus précisément, on sait que la probabilité maximale pour un adversaire de briser le secret est au moins de 0.8.

Le but de la section suivante est de corriger la spécification grâce au processus de raffinement, afin si possible de valider le requis de sécurité, tout en spécifiant un peu plus précisément le fonctionnement du système de contrôle d'accès modélisé.

6.4 Un raffinement

La spécification $\mathcal{S}_2$ est le résultat d'une première modélisation abstraite du système de contrôle d'accès, tel que discuté dans la section 6.1.1. Il est possible d'appliquer le processus de raffinement afin d'améliorer le système, en modélisant progressivement les différents composants du système.

Précédemment, la spécification envisageait un mécanisme autorisant un nouvel essai à l'utilisateur ayant entré un mot de passe erroné, sans détailler. On propose ici le mécanisme suivant afin d'améliorer le système de vérification du mot de passe. Si le mot de passe entré par l'utilisateur est correct, rien n'est modifié. Si en revanche il est erroné, le programme vérifie si la tentative appartient à une liste noire de mots courants : dans ce cas, afin de contrecarrer une éventuelle attaque de type dictionnaire, l'accès est refusé et le système passe dans l'état déconnecté.

Ceci est représenté par l'IDTMC de la figure 6.11. L'état q_1 de la spécification $\mathcal{S}_2$ originale est séparé en deux états : r_1 représente la vérification effective du mot de passe entré par l'utilisateur, tandis que r_2 représente la lecture de la liste noire et la défense contre les éventuelles attaques de type dictionnaire.

L'IDTMC proposée est non-modale. L'aspect abstrait du système de vérification de mot de passe dans la spécification $\mathcal{S}_2$ justifie les modalités des transitions issues de l'état q_1 ; dans $\mathcal{S}_1$, en revanche, on suppose que le processus ajouté doit nécessairement être implémenté. Par conséquent, on retire toute modalité dans les transitions issues des états r_1 et r_2.

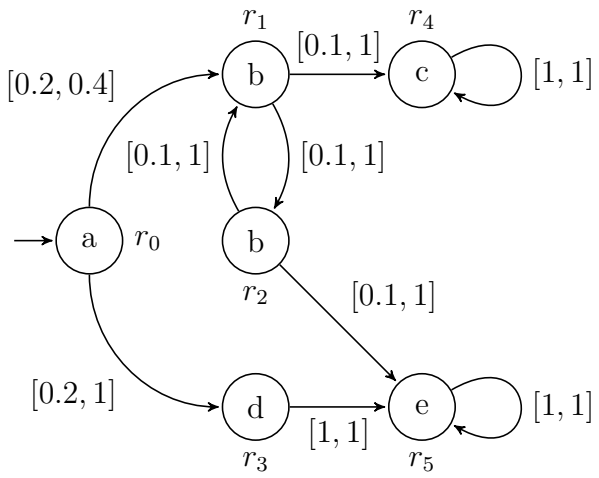


Figure 6.11 Un raffinement $\mathcal{S}_1$ du système précédent

Preuve du raffinement fort de $\mathcal{S}_2$ par $\mathcal{S}_1$. Posons $\mathcal{R} \subseteq S_1 \times S_2$ la relation représentée par la matrice $M_{\mathcal{R}}$.

$$M_{\mathcal{R}} = \begin{array}{c} \\ r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{array} \begin{array}{c} \begin{array}{ccccc} q_0 & q_1 & q_2 & q_3 & q_4 \end{array} \\ \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix} \end{array}$$

Alors, notamment, $r_0 \mathcal{R} q_0$, et, pour $r \in S_1$ et $q \in S_2$, si $r \mathcal{R} q$,

— $\lambda_1(r) = \lambda_2(q)$ de manière évidente ;

— posons $\delta : S_1 \to \mathcal{D}ist(S_2)$ l'application telle que

$$\forall r' \in S_1,\ \forall q' \in S_2,\ \delta(r')(q') = \begin{cases} 1 & \text{si } r' \mathcal{R} q' \\ 0 & \text{sinon.} \end{cases}$$

Alors pour toute distribution $f \in T_1(r)$,

$$\forall q' \in S_2,\ \sum_{r' \in S_1} f(r') \cdot \delta(r')(q') = \sum_{r' \mathcal{R} q'} f(r') \in T_2(q)(q').$$

De plus, par définition de δ, on a $r' \mathcal{R} q'$ si, et seulement si, $\delta(r')(q') > 0$.

Par exemple, pour $r_0 \mathcal{R} q_0$,

$$T_1(r_0) = \{r_0 \xrightarrow{[0.2,0.4]} r_1, r_0 \xrightarrow{[0.2,1]} r_3\}$$

et

$$T_2(q_0) = \{q_0 \xrightarrow{[0.2,1]} q_1, q_0 \xrightarrow{[0.2,1]} q_3\}.$$

Soit une distribution $f \in T_1(r_0)$. Alors,

— pour $q' = q_1$, $\sum_{r' \in S_1} f(r') \cdot \delta(r')(q_1) = f(r_1) \in [0.2, 0.4] \subseteq T_2(q_0)(q_1)$;

— pour $q' = q_3$, $\sum_{r' \in S_1} f(r') \cdot \delta(r')(q_3) = f(r_3) \in [0.2, 1] \subseteq T_2(q_0)(q_3)$;

— dans les autres cas, $\sum_{r' \in S_1} f(r') \cdot \delta(r')(q') = 0 \in \{0\} \subseteq T_2(q_0)(q')$.

On traite les autres cas de manière analogue, ce qui termine la preuve de raffinement fort de $\mathcal{S}_2$ par $\mathcal{S}_1$. □

6.4.1 Calcul d'opacité libérale

Puisque $\mathcal{S}_1$ n'est pas modale, il est possible d'appliquer le théorème 2 afin de calculer l'opacité libérale de φ relativement à $\mathcal{O}$. Pour cela, calculons $\mathcal{V}(\mathcal{S}_2, \varphi, \mathcal{O})$.

Le langage de la spécification $\mathcal{S}_1$ est $L_1 = ab(bb)^* c^\omega + a(bb)^+ e^\omega + ade^\omega$ donc

$$\begin{aligned}
L_1 \setminus \varphi &= ab(bb)^+ c^\omega + a(bb)^+ e^\omega + ade^\omega \\
\mathcal{O}(L_1 \setminus \varphi) &= ac^\omega + ae^\omega + ade^\omega \\
\varphi \cap L_1 &= abc^\omega \\
\mathcal{O}(\varphi \cap L_1) &= ac^\omega \\
\text{donc } \mathcal{O}(\varphi \cap L_1) &\subseteq \mathcal{O}(L_1 \setminus \varphi)
\end{aligned}$$

Par conséquent, φ est opaque dans $\mathcal{S}_1$ relativement à $\mathcal{O}$, c'est-à-dire que

$$S\_PO_l(\mathcal{S}_1, \varphi, \mathcal{O}) = 0.$$

Remarquons que $S\_PO_l(\mathcal{S}_1, \varphi, \mathcal{O}) = 0 < 0.8 \leq S\_PO_l(\mathcal{S}_2, \varphi, \mathcal{O})$, ce qui est cohérent avec le théorème 4, puisque $\mathcal{S}_1$ raffine fortement $\mathcal{S}_2$.

Ici, le système raffiné valide les requis de sécurité, ce qui est une nette amélioration devant

le système précédent. Ceci est dû au raffinement des transitions modales en transitions non-modales. L'ajustement des distributions problématiques permet d'écarter du domaine des possibles les implémentations qui présentent une faille de sécurité.

Ainsi, outre l'ajout d'une fonctionnalité au système de contrôle d'accès, le raffinement permet de conserver voire d'améliorer l'opacité d'un secret régulier.

6.4.2 Calcul d'opacité restrictive

D'après les résultats de la section 6.4.1, la spécification $\mathcal{S}_1$ valide les requis d'opacité du secret. Cependant, tel que cela a été discuté dans la partie 3.2, vérifier si un prédicat est opaque dans une spécification est un résultat peu nuancé. L'étape suivante consiste à donner une approximation de la valeur de l'opacité restrictive – puisque la valeur exacte est très complexe à obtenir.

Rappelons que l'opacité restrictive du secret dans la spécification $\mathcal{S}_1$ est définie par l'expression (*cf.* définition 25) :

$$S\_PO_r(\mathcal{S}_1, \varphi, \mathcal{O}) = \inf_{\mathcal{A} \in S\_imp(\mathcal{S}_1)} PO_r(\mathcal{A}, \varphi, \mathcal{O}).$$

Pour une implémentation ordonnancée fixée $\mathcal{A} \in S\_imp(\mathcal{S}_1)$,

$$\frac{1}{PO_r(\mathcal{A}, \varphi, \mathcal{O})} = \sum_{o \in Obs} \mathbf{P}(o) \cdot \frac{1}{\mathbf{P}(L \setminus \varphi \,|\, o)}.$$

Considérons dans un premier temps uniquement les implémentations ordonnancées par un adversaire sans-mémoire. On peut alors représenter un tel ordonnancement par le PTS $\mathcal{A}$ de la figure 6.12, avec les conditions suivantes sur les variables p_1, p_2, et p_3.

$$p_1 \in [0.2, 0.4]; \quad p_2 \in [0.1, 0.9]; \quad p_3 \in [0.1, 0.9]$$

Calculons alors l'expression générale de l'opacité restrictive de φ dans $\mathcal{A}$.

Les classes d'observation du PTS par l'observateur rationnel $\mathcal{O}$ sont :

- $o_1 = ac^\omega = \mathcal{O}(ab(bb)^* c^\omega)$;
- $o_2 = ae^\omega = \mathcal{O}(a(bb)^+ e^\omega)$;
- $o_3 = ade^\omega = \mathcal{O}(ade^\omega)$.

Pour chaque classe $o \in Obs$, on calcule la probabilité de la classe, $\mathbf{P}(o)$, ainsi que la probabilité de toute exécution de la classe sachant qu'elle ne fait pas partie du secret, $\mathbf{P}(L \setminus \varphi \,|\, o)$.

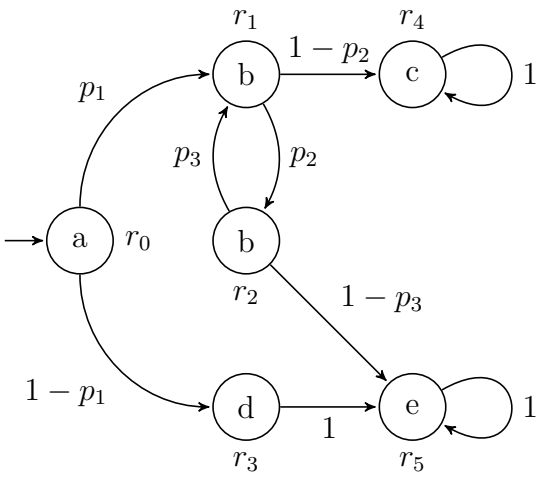


Figure 6.12 Ordonnancement sans-mémoire quelconque de $\mathcal{S}_1$

Pour cela, on suit le raisonnement suivant.

Une exécution du langage $ab(bb)^*c^\omega$ s'écrit $ab(bb)^kc^\omega$, avec $k \in \mathbb{N}$. Ainsi, la probabilité d'une telle exécution est

$$\mathbf{P}(ab(bb)^kc^\omega) = p_1(1-p_2)\cdot(p_2\cdot p_3)^k.$$

Par conséquent, pour calculer la probabilité de tout le langage correspondant, on somme les probabilités de toutes les exécutions appartenant à ce langage. On obtient ainsi la somme de la série infinie suivante.

$$\mathbf{P}(ab(bb)^*c^\omega) = p_1(1-p_2)\sum_{k=0}^{\infty}(p_2\cdot p_3)^k$$

Puisque cette somme est la somme d'une série géométrique de raison $p_2 \cdot p_3 < 1$, la somme de la série est finie et on peut calculer sa valeur.

$$\mathbf{P}(ab(bb)^*c^\omega) = p_1(1-p_2)\cdot\frac{1}{1-p_2\cdot p_3}.$$

On obtient avec le même raisonnement les résultats, reportés dans le tableau 6.4.

Tableau 6.4 Probabilités nécessaires au calcul de l'opacité restrictive

Classe d'observation	$\mathbf{P}(o)$	$\mathbf{P}(L \setminus \varphi \mid o)$
ac^ω	$p_1(1-p_2)\sum_{k=0}^{\infty}(p_2\cdot p_3)^k$	$\frac{1}{\sum_{k=0}^{\infty}(p_2\cdot p_3)^k}$
ae^ω	$p_1p_2(1-p_3)\sum_{k=0}^{\infty}(p_2\cdot p_3)^k$	1
ade^ω	$1-p_1$	1

Finalement, on obtient l'expression suivante d'opacité restrictive.

$$\begin{aligned}\frac{1}{PO_r(\mathcal{A},\varphi,\mathcal{O})} &= 1 - p_1 + \frac{p_1 p_2 (1-p_3)}{1-p_2 \cdot p_3} + \frac{p_1(1-p_2)}{(1-p_2 \cdot p_3)^2} \\ &= 1 + p_1 \left(\frac{p_2(1-p_3)}{1-p_2 \cdot p_3} + \frac{(1-p_2)}{(1-p_2 \cdot p_3)^2} - 1 \right)\end{aligned}$$

Rappelons que le but est d'extraire la valeur minimale de l'opacité restrictive, donc la valeur maximale de cette fonction. Pour cela, il suffit de poser $p_1 = 0.4$ – qui est la valeur maximale de p_1 – puis de trouver la valeur maximale de la fonction

$$f(p_2, p_3) = \frac{p_2(1-p_3)}{1-p_2 \cdot p_3} + \frac{(1-p_2)}{(1-p_2 \cdot p_3)^2} - 1$$

sur le domaine $[0.1, 0.9]^2$.

Cette fonction étant non-linéaire, le problème fait appel à des procédés complexes d'optimisation non-linéaire, ce qui n'est pas le propos de cette étude. Le graphe de la fonction sur le domaine considéré est tracé sur la figure 6.13.

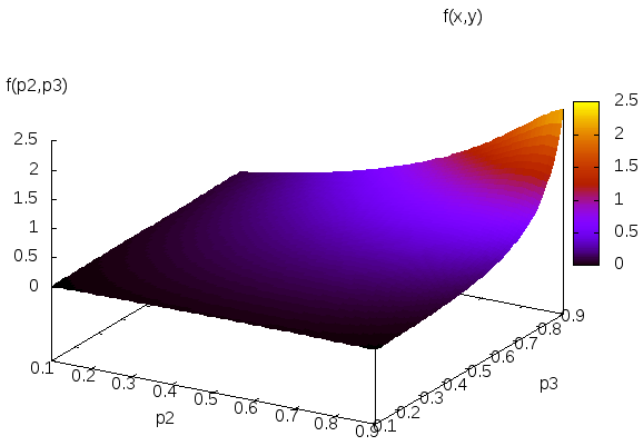


Figure 6.13 Graphe de la fonction f

Finalement, on déduit de cette étude que l'opacité restrictive minimale de la spécification, si on se limite aux implémentations ordonnancées sans mémoire, est atteinte pour l'implémentation de la figure 6.12 telle que $p_1 = 0.4$, $p_2 = 0.9$ et $p_3 = 0.9$. Pour cette implémentation, l'opacité restrictive est $PO_r(\mathcal{A}, \varphi, \mathcal{O}) \approx 0.52$.

On peut donc déduire de ce résultat que l'opacité restrictive minimale de la spécification

vérifie

$$S\_PO_r(\mathcal{S}_1, \varphi, \mathcal{O}) \leq 0.5.$$

Il est possible d'affiner ce résultat, en réitérant l'opération pour l'ensemble des implémentations ordonnancées par un adversaire à mémoire quelconque n, dans le même esprit que lors du calcul approché de l'opacité libérale dans le cas d'une spécification modale.

Ce chapitre a permis l'application d'une étape du processus de raffinement sur un cas concret de contrôle d'accès à une base de données. Les algorithmes énoncés et prouvés dans le chapitre précédent ont permis la vérification des requis de sécurité sur les deux systèmes sécurisés considérés. Notamment, tel qu'il a été prouvé dans le chapitre précédent, raffiner le système a permis d'améliorer la sécurité du système, tout en précisant ses fonctionnalités.

CHAPITRE 7 CONCLUSION

7.1 Synthèse des travaux

Le but de ce mémoire consistait à étudier la formalisation du raffinement de systèmes sécurisés. En règle générale, il est connu que cette problématique aboutit à un résultat négatif : en effet, le raffinement d'un système sécurisé peut briser la sécurité et le rendre vulnérable pour certaines propriétés. Cependant, ce résultat n'est pas absolu, et notre objectif était de représenter un cadre formel dans lequel raffiner un système sécurisé préserverait la sécurité.

Dans cette optique, nous avons tout d'abord défini le cadre formel de recherche, en choisissant comme modèle de spécification l'IDTMC, couplé au PTS en tant que modèle d'implémentation. Ces deux modèles utilisant les hypothèses markoviennes permettent la représentation de nombreux systèmes réels, tout en appliquant des méthodes quantitatives, nécessaire à la modélisation de propriétés de sécurité. En parallèle, nous avons choisi l'opacité comme propriété formelle de sécurité, possédant l'avantage d'être connue dans le cadre des PTS et étant calculable en un temps fini. De plus, l'opacité est liée aux propriétés de flux d'information, qui sont au cœur d'importantes problématiques sur la confidentialité des données et des échanges : elles représentent l'information que peut déduire un agent extérieur en observant un système.

Afin de justifier l'intérêt du choix de ce cadre, nous avons commencé par définir l'extension de l'opacité au domaine des IDTMC, puis nous avons prouvé que cette grandeur était calculable ou approchable en un temps fini. De fait, nous avons expliqué la méthode permettant de calculer l'opacité libérale dans une IDTMC non-modale, c'est-à-dire une spécification dont toutes les transitions sont nécessairement ordonnancées. Ce calcul, revenant à l'optimisation d'un système linéaire dans un ensemble fermé borné, se réalise en un temps doublement exponentiel. De plus, en étudiant le cas général, nous avons construit les algorithmes permettant de nous ramener à des calculs d'opacité dans le cas des spécifications non-modales. Ceux-ci sont fondés sur le fait que nous listons de manière formelle tous les ordonnancements à mémoire finie de l'IDTMC, afin d'en sortir le cas le moins sécurisé. Ainsi, nous avons pu affirmer que la valeur de l'opacité libérale dans le cas général pouvait être approchée en un temps doublement exponentiel, ce qui constitue notre première contribution originale en étendant le résultat précédent, limité à une partie des spécifications.

Consécutivement, nous avons résolu notre problématique en énonçant le théorème suivant : l'opacité libérale d'une spécification IDTMC est, dans le pire des cas, conservée lorsque la

spécification est raffinée faiblement. Ce résultat s'appuie sur l'étude de la relation de raffinement d'une part, et la relation d'ordonnancement d'autre part. Il constitue notre seconde contribution originale, en généralisant le résultat déjà connue de la préservation de l'opacité par raffinement fort uniquement.

Enfin, nous avons illustré tous ces résultats sur un exemple d'application. Nous avons considéré un système de contrôle d'accès à une base de données confidentielles, que nous avons modélisé conformément à notre cadre d'études. Nous avons calculé que la première spécification n'était pas opaque, en appliquant les algorithmes expliqués précédemment. Nous avons alors défini un raffinement du système, et dans la même optique, nous avons assuré l'opacité, ce qui a illustré le résultat de préservation de la sécurité.

7.2 Limitations de la solution proposée

Bien qu'il constitue une avancée dans la théorie, ce travail de recherche comporte certaines limitations que nous n'avons pu franchir.

Tout d'abord, le théorème de calcul de l'opacité libérale dans le cas général n'aboutit pas à la preuve que la valeur exacte est calculable en temps fini. En effet, puisque l'algorithme consiste à énumérer les cas pour les différents ordonnanceurs à mémoire finie, et puisqu'il y a théoriquement une infinité de ces ordonnanceurs – sans compter les ordonnanceurs à mémoire quelconque – il est *a priori* impossible de réaliser le calcul exact en un temps fini : la question reste ouverte.

En second lieu, nous avons prouvé que l'opacité libérale était préservée lors d'un raffinement faible, ce qui implique qu'elle est préservée lors d'un raffinement fort. Pour autant, rien n'est prouvé pour le raffinement complet. Cette question est encore ouverte.

Enfin, la plupart de l'étude a été réalisée en prenant en compte l'opacité libérale. La question quant à sa grandeur duale, l'opacité restrictive, est encore en suspens. Or cette opacité est justement intéressante pour distinguer des systèmes dont le secret est effectivement opaque, à l'inverse de l'opacité libérale, qui détermine le degré de non-opacité du secret dans un système. La difficulté réside ici dans le fait que l'opacité restrictive dans un PTS n'est pas la probabilité d'un ensemble régulier mais le résultat d'une moyenne harmonique de probabilités particulières.

7.3 Améliorations futures

Cette recherche est le fruit de l'étude d'un cadre particulier dans la problématique du raffinement de systèmes sécurisés. Les travaux futurs consistent donc à continuer dans cette voie en essayant de généraliser les résultats obtenus.

La première généralisation consiste à vérifier si les théorèmes sont toujours valables si l'on modifie la nature de la propriété. Notamment, que se passe-t-il lorsque l'on remplace l'opacité libérale par une autre RIFP (Bérard et Mullins, 2014) ? Si on parvient à prouver la généralisation, il sera possible de considérer un grand nombre de propriétés formelles de sécurité. En effet, il est possible de prouver que les Prédicats Basiques de Sécurité (BSP) (Mantel, 2000, 2001) – introduites par Mantel et qui constituent les briques élémentaires de toutes les propriétés de sécurité – peuvent être vues comme des RIFP, en les calculant sous le sens libéral. Ces deux résultats combinés sont autant de poinsts justifiant l'intérêt de ce formalisme.

La seconde généralisation se trouve du côté du choix de modélisation. Le choix de l'IDTMC permet un formalisme contraint qui facilite les résultats. Serait-il possible d'étendre le tout à des objets plus généraux, tels que les Chaînes de Markov Contraintes (CMC) ? Dans ce formalisme, les probabilités de transitions ne sont plus restreintes par des intervalles, mais par des relations de contraintes quelconques. Ainsi, les IDTMC sont des CMC particulières.

Au-delà de ces extensions du domaine d'application des résultats, il serait intéressant d'explorer davantage le calcul de l'opacité restrictive dans les IDTMC. L'aspect non-linéaire de cette grandeur pose problème, mais peut-être existe-t-il un moyen de linéariser les fonctions afin d'avoir une approximation de la valeur. Il faudrait pour cela suivre des pistes d'optimisation de fonctions.

Enfin, nos résultats sont pour le moment dans un cadre très théorique et peu applicatif. Le logiciel de modélisation et vérification PRISM est actuellement capable de calculer l'opacité libérale d'un PTS ; il serait intéressant d'explorer la possibilité de modéliser les IDTMC sur ce logiciel. Si cela s'avère possible, il serait alors envisageable de calculer l'opacité libérale d'une IDTMC, la difficulté étant alors de se limiter aux ordonnancements.

RÉFÉRENCES

Martín Abadi et Leslie Lamport : The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253 – 284, 1991. ISSN 0304-3975. URL http://www.sciencedirect.com/science/article/pii/030439759190224P.

Christel Baier et Joost-Pieter Katoen : *Principles of model checking*. MIT Press, 2008. ISBN 978-0-262-02649-9.

Christel Baier, Joost-Pieter Katoen, Holger Hermanns et Verena Wolf : Comparative branching-time semantics for markov chains. *Inf. Comput.*, 200(2):149–214, 2005. URL http://dx.doi.org/10.1016/j.ic.2005.03.001.

Béatrice Bérard, Krishnendu Chatterjee et Nathalie Sznajder : Probabilistic opacity for markov decision processes. *Inf. Process. Lett.*, 115(1):52–59, 2015a. URL http://dx.doi.org/10.1016/j.ipl.2014.09.001.

Béatrice Bérard, Olga Kouchnarenko, John Mullins et Mathieu Sassolas : Probabilistic opacity in refinement-based modeling. *CoRR*, abs/1510.04316, 2015b. URL http://arxiv.org/abs/1510.04316.

Béatrice Bérard et John Mullins : Verification of information flow properties under rational observation. *CoRR*, abs/1409.0871, 2014. URL http://arxiv.org/abs/1409.0871.

Béatrice Bérard, John Mullins et Mathieu Sassolas : Quantifying opacity. *Mathematical Structures in Computer Science*, 25(2):361–403, 2015c. URL http://dx.doi.org/10.1017/S0960129513000637.

Fabrizio Biondi, Axel Legay, Bo Friis Nielsen et Andrzej Wasowski : Maximizing entropy over markov processes. *J. Log. Algebr. Meth. Program.*, 83(5-6):384–399, 2014. URL http://dx.doi.org/10.1016/j.jlamp.2014.05.001.

Jeremy Bryans, Maciej Koutny, Laurent Mazaré et Peter Y. A. Ryan : Opacity generalised to transition systems. *Int. J. Inf. Sec.*, 7(6):421–435, 2008. URL http://dx.doi.org/10.1007/s10207-008-0058-x.

Jeremy Bryans, Maciej Koutny et Peter Y. A. Ryan : Modelling dynamic opacity using petri nets with silent actions. *In Formal Aspects in Security and Trust : Second IFIP TC1 WG1.7 Workshop on Formal Aspects in Security and Trust (FAST), an event of the 18th IFIP World Computer Congress, August 22-27, 2004, Toulouse, France*, pages 159–172, 2004. URL http://dx.doi.org/10.1007/0-387-24098-5_12.

Jeremy Bryans, Maciej Koutny et Peter Y. A. Ryan : Modelling opacity using petri nets. *Electr. Notes Theor. Comput. Sci.*, 121:101–115, 2005. URL http://dx.doi.org/10.1016/j.entcs.2004.10.010.

JR Büchi : On a decision method in restricted second order arithmetic logic. *Proc. 1962 Internat. Congr. on Methodology and Philosophy of Sciences*, 1962.

Pavol Cerný, Krishnendu Chatterjee et Thomas A. Henzinger : The complexity of quantitative information flow problems. *In Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June, 2011*, pages 205–217, 2011. URL http://dx.doi.org/10.1109/CSF.2011.21.

Krishnendu Chatterjee, Koushik Sen et Thomas A. Henzinger : Model-checking omega-regular properties of interval markov chains. *In Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 - April 6, 2008. Proceedings*, pages 302–317, 2008. URL http://dx.doi.org/10.1007/978-3-540-78499-9_22.

Michael R. Clarkson et Fred B. Schneider : Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010. URL http://dx.doi.org/10.3233/JCS-2009-0393.

Benoît Delahaye, Kim G Larsen, Axel Legay, Mikkel L Pedersen et Andrzej Wąsowski : Decision problems for interval markov chains. *In International Conference on Language and Automata Theory and Applications*, pages 274–285. Springer, 2011.

Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen et Andrzej Wasowski : Consistency and refinement for interval markov chains. *J. Log. Algebr. Program.*, 81(3):209–226, 2012. URL http://dx.doi.org/10.1016/j.jlap.2011.10.003.

Joseph A. Goguen et José Meseguer : Security policies and security models. *In 1982 IEEE Symposium on Security and Privacy, Oakland, CA, USA, April 26-28, 1982*, pages 11–20, 1982. URL http://dx.doi.org/10.1109/SP.1982.10014.

Bengt Jonsson et Kim Guldstrand Larsen : Specification and refinement of probabilistic processes. *In Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91), Amsterdam, The Netherlands, July 15-18, 1991*, pages 266–277, 1991. URL http://dx.doi.org/10.1109/LICS.1991.151651.

Feng Lin : Opacity of discrete event systems and its applications. *Automatica*, 47(3):496–503, 2011.

Heiko Mantel : Possibilistic definitions of security - an assembly kit. *In Proceedings of the 13th IEEE Computer Security Foundations Workshop, CSFW '00, Cambridge, England,*

UK, July 3-5, 2000, pages 185–199, 2000. URL http://dx.doi.org/10.1109/CSFW.2000.856936.

Heiko Mantel : Preserving information flow properties under refinement. *In 2001 IEEE Symposium on Security and Privacy, Oakland, California, USA May 14-16, 2001*, pages 78–91, 2001. URL http://dx.doi.org/10.1109/SECPRI.2001.924289.

Laurent Mazaré : Using unification for opacity properties. *Proceedings of the 4th IFIP WG1*, 7:165–176, 2004.

John Mullins et Moez Yeddes : Opacity with orwellian observers and intransitive non-interference. *In 12th International Workshop on Discrete Event Systems, WODES 2014, Cachan, France, May 14-16, 2014.*, pages 344–349, 2014. URL http://dx.doi.org/10.3182/20140514-3-FR-4046.00016.

Nir Piterman : From nondeterministic büchi and streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3), 2007. URL http://dx.doi.org/10.2168/LMCS-3(3:5)2007.

John Rushby : *Noninterference, transitivity, and channel-control security policies*. SRI International, Computer Science Laboratory, 1992.

Anooshiravan Saboori et Christoforos N Hadjicostis : Current-state opacity formulations in probabilistic finite automata. *IEEE Transactions on automatic control*, 59(1):120–133, 2014.

Mathieu Sassolas : *Qualitative and Quantitative Methods for Detection of Hidden Information*. Theses, Université Pierre et Marie Curie - Paris VI, novembre 2011. URL https://tel.archives-ouvertes.fr/tel-00683086.

Claude Elwood Shannon : A mathematical theory of communication. 1948.

Geoffrey Smith : On the foundations of quantitative information flow. *In Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, pages 288–302, 2009. URL http://dx.doi.org/10.1007/978-3-642-00596-1_21.

David Sutherland : A model of information. *In Proc. 9th National Computer Security Conference*, pages 175–183. DTIC Document, 1986.