Caractérisation des canaux cachés en logique temporelle alternante
Rapport de stage de master

Aldric Degorre - ENS Cachan, antenne de Bretagne
stage encadré par
Loïc Hélouët - IRISA, équipe DISTRIBCOM

22 juin 2005

Résumé

Dans un système distribué, les canaux cachés sont des flots d'information illicites et généralement implicites. La caractérisation la plus classique des flots d'information est celle donnée par la notion d'interférence. Ici nous considérons qu'un canal caché est une interférence arbitrairement réitérée par un pirate. Nous proposons ainsi un moyen de détecter de tels canaux cachés dans un système modélisé par un automate fini. Pour cela nous montrons comment obtenir une représentation plus pratique, sous forme d'arène, de ce système. Puis, en s'aidant de la notion d'interférence, nous exprimons la propriété de canal caché dans une logique temporelle appelée μ-calcul à temps alternant. Nous caractérisons ensuite autrement la propriété de canal caché : par un jeu sur une arène dérivée de la première. Enfin, nous vérifions que la propriété logique de canal caché implique effectivement l'existence d'une stratégie gagnante dans ce jeu pour d'éventuels attaquants du système.

Remerciements

Avant d'entamer le compte-rendu de mon stage, je souhaiterais remercier vivement l'équipe DISTRIBCOM qui m'a accueilli, et Loïc Hélouët qui m'a aidé à garder le fil conducteur tout au long de ce stage.

Je tiens aussi à remercier tout spécialement Sophie Pinchinat qui a accepté de consacrer quelques heures de son temps pour m'éclairer sur les questions de logique temporelle. En particulier, sans elle, je n'aurais peut-être pas entendu parler des logiques à temps alternant, qui se sont révélées être l'outil primordial dans le travail présenté ici.

Table des matières

Introduction

Dans ce stage réalisé parmi l'équipe DISTRIBCOM, on se propose de diagnostiquer, dans un système distribué utilisé par plusieurs agents, la présence d'un type particulier de flots d'information, à savoir le canal caché.

Les canaux cachés sont des flots d'information illicites, et souvent implicites, qui peuvent être maîtrisés par d'éventuels pirates pour se transmettre de l'information en violant la politique de sécurité du système. Pour provoquer ces flots, les pirates détournent certaines capacités du système qui n'étaient pas prévues à sa conception pour communiquer de l'information de cette manière-là. Pour cette raison, ces canaux sont considérés comme « cachés », c'est-à-dire non (directement) visibles dans la spécification du système. Outre les problèmes de sécurité purs que posent les canaux cachés, se posent des problèmes divers tels que ceux liés à la facturation (pour des réseaux loués par un opérateur) ou aux performances du système, qui évidemment se dégradent quand celui-ci est détourné de son utilisation normale et que les ressources sont utilisées abusivement pour cette raison. Ainsi, par exemple, utiliser la sonnerie du téléphone pour communiquer gratuitement en morse est un canal caché qui fait travailler les équipements de l'opérateur téléphonique et rend donc le réseau moins disponible.

Les flots d'information dans les systèmes informatiques ont par le passé été caractérisés de plusieurs manières, en particulier par la propriété de non-interférence [10]. Cette dernière a été par la suite mise en évidence par différentes approches, dont nous avons fait la synthèse dans le travail de recherche bibliographique préalable, repris dans la première partie de ce rapport.

Ici nous ne nous plaçons pas dans l'approche analyse de programme, mais plutôt dans la vérification de propriétés comportementales sur un système de transitions.

Nous considérerons le cas particulier des canaux cachés à usage arbitrairement répété : c'est-à-dire une interférence réitérée, entre deux pirates, qui permette de transmettre un message de taille arbitrairement grande. En cela, ce travail s'inscrit dans la continuité de [13] pour ce qui est de ses objectifs, mais là nous recherchons une caractérisation qui utilise d'autres outils : nous remplacerons les langages de scénarios par des automates et des arènes, et parce que ce problème met en jeu une certaine adversité entre les pirates et le reste du système, nous le traiterons par une logique dite à temps alternant (nous avons choisi le μ-calcul à temps alternant) et par la théorie des jeux.

Ainsi, dans la deuxième partie de ce rapport, après avoir donné les définitions d'ordre général puis introduit notre logique, nous expliquerons ce que nous entendons par interférence, puis par canal caché en explicitant ces notions dans une variante à observations partielles, mais décidable, du μ-calcul à temps alternant. Après quoi, nous caractériserons le problème du canal caché par un jeu distribué où une équipe de deux « pirates », un émetteur et un récepteur auxquels la politique de sécurité voudrait interdire de communiquer, jouent contre l'environnement pour tenter de se transmettre un message. Et enfin, nous montrerons que les deux approches ont un lien, à savoir que si la formule de μ-calcul à temps alternant est satisfaite, alors il existe une stratégie gagnante pour les pirates.

Première partie

Étude bibliographique préliminaire

En préparation à ce stage, il nous avait été demandé d'effectuer quelques recherches bibliographiques afin d'une part de se mettre au courant, et d'autre part de justifier le fait d'entreprendre le travail proposé pour le stage. C'est le compte-rendu de ces recherches que je reprends ici dans cette partie et dont je conseille la lecture avant d'entamer celle du rapport de stage proprement dit (deuxième partie), à moins que le lecteur soit déjà familier avec la notion de non-interférence et les flots d'information.

Ainsi, dans cette partie, nous allons d'abord faire le point sur les modèles de fuite d'information, notamment les modèles d'interférence, puis résumer ce que l'on sait faire dans la détection de canaux cachés à l'aide des scénarios et de la théorie des jeux.

1 Modéliser la sécurité des flots d'information

Les modèles de sécurité des flots d'information ont pu prendre des formes sensiblement différentes selon les époques, les contextes et les différentes propriétés que l'on a voulu assurer. Nous allons passer en revue quelques uns des plus significatifs, du moins du point de vue de la non-interférence.

1.1 Modèle classique de Bell et LaPadula

Dans ce qui a constitué la première tentative marquante de formalisation mathématique de la sécurité informatique ([7, 6]), Bell et LaPadula ont proposé un modèle à plusieurs niveaux de sécurité (public, confidentiel, secret, top secret ...) où les agents de niveau haut n'ont pas le droit d'écrire dans des ressources de niveau plus bas et où les agents de niveau bas ne peuvent pas lire des ressources de niveau plus haut.

Ici, ce formalisme n'a pour nous d'intérêt autre qu'historique, car bien qu'il ait eu des applications dans les politiques de contrôle d'accès aux données, il est relativement peu adapté à la description des fuites de données dans des processus dynamiques.

1.2 Modèles de non interférence

La plupart des articles sur le sujet s'accordent pour dire qu'un processus U n'interfère pas avec un processus V dans le système $\Sigma = U \parallel P \parallel V$ si et seulement si on peut vérifier une formule ressemblant à celle-ci :

$$\forall U_1, U_2 \cdot U_1 \parallel P \parallel V \ \equiv_V U_2 \parallel P \parallel V$$

où

- U_1 et U_2 sont deux comportements possibles du processus U
- $\parallel$ est une loi de composition de processus, par exemple un produit synchronisé d'automates communicants
- $\equiv_V$ est une relation d'équivalence entre systèmes qui correspond à ce que V est capable d'observer (en général équivalence des traces projetées sur V).

Les approches qui suivent instancient ce schéma différemment selon leur manière de modéliser les processus, et selon le modèle de sécurité choisi.

1.3 Canaux cachés

Un canal est un mécanisme de transfert d'information dans un système. Certains d'entre eux exploitent des mécanismes dont le but originel n'est pas le transfert d'information. Ce sont les canaux cachés.

La notion de canal caché raffine celle d'interférence : pour qu'il y ait un canal caché, il faut en effet que d'une part il soit possible d'interférer, et que d'autre part cette interférence soit exploitable volontairement et plusieurs fois par un agent mal intentionné qui pourrait ainsi transmettre à un interlocuteur non autorisé une quantité arbitraire d'information.

Il existe des canaux cachés dès qu'il y a des ressources partagées entre plusieurs processus. Admettons par exemple que deux utilisateurs d'un système informatique ne soient pas autorisés à communiquer mais aient tous les deux leurs données stockées sur le même disque dur. Pour peu qu'il n'y ait pas de quotas, l'un des utilisateurs pourrait saturer et libérer répétitivement le disque rendant les écritures alternativement possibles et impossibles pour l'autre utilisateur qui peut en déduire ce que le premier voulait lui transmettre.

D'autres canaux cachés classiques consistent pour l'utilisateur à mesurer la vitesse à laquelle s'exécutent ses processus afin de déduire ce que font d'autres processus à ce moment alors qu'il ne peut pas les observer directement. Ce sont les *timing channels*.

Un paragraphe de [15] répertorie les types de canaux cachés les plus courants. Mais dans la deuxième partie de ce rapport, les canaux cachés considérés sont encore d'une autre sorte : ce seraient plutôt des canaux cachés qui exploitent les messages d'un protocole pour transférer des données, c'est-à-dire des canaux légitimes !

1.4 La non-interférence selon Goguen et Meseguer

Goguen et Meseguer, dans [10], sont les premiers à avoir formalisé la notion de non-interférence.

L'originalité de leur approche est d'avoir distingué modèles de sécurité et politiques de sécurité. Le modèle de sécurité est la modélisation du comportement d'un système alors que la politique de sécurité est une liste de propriétés que l'on souhaiterait voir respectées par le modèle.

Ce qui servira de modèle de sécurité, ici, sera la donnée d'un automate (pas forcément fini) appelé *capability system*, dont l'alphabet d'entrée est $U \times C$ (U : ensemble des utilisateurs, C : ensemble des commandes), et muni d'une fonction d'observation, out, qui associe à un utilisateur et à un état, un phénomène observable par cet utilisateur. On appelle $csdo$ la fonction de transition de l'automate.

Les politiques de sécurité sont elles données par une liste d'affirmations de non-interférence :

$$[[w]]_u = [[p_{G,A}(w)]]_u$$

où $G \subset U$, $A \subset C$, w est un mot d'entrée, $p_{G,A}(w)$ est le mot d'entrée w où l'on a supprimé les paires $(u, c) \in G \times A$, $[[w]]_u = out([[w]], u)$ et $[[w]] = csdo(w)$. Autrement dit, « les utilisateurs du groupe G avec les capacités de A n'ont pas d'influence sur ce que u observe du système ».

L'approche de Goguen et Meseguer rentre dans la grande famille de celles qui caractérisent la non-interférence par une propriété sur un programme d'une algèbre de processus.

1.5 Mesurer les fuites d'information dans une algèbre de processus

C'est ce que propose de réaliser Gavin Lowe dans [14] avec le langage CSP à temps discret.

Dans l'exemple suivant, le processus Bas (capable d'envoyer li) peut distinguer trois comportements de P, suivant la manière dont a communiqué Haut (capable d'envoyer hi) avec P :

$$P \triangleq \begin{pmatrix} h1 \rightarrow l1 \rightarrow STOP \\ \Box h2 \rightarrow l2 \rightarrow STOP \end{pmatrix} \triangleright^1 STOP$$

où

- $e \rightarrow P$ est le processus qui attend l'événement e, puis se comporte comme P
- $P \Box Q$ est le processus qui se comporte soit comme P, soit comme Q, l'indétermination étant levée par des événements extérieurs
- $P \triangleright^t Q$ est le processus qui se comporte comme P, puis comme Q si aucun événement de P n'a eu lieu pendant les t premières unités de temps.

On considère le système où P est composé parallèlement avec un processus Haut et un processus Bas.

Si Haut n'a rien envoyé, P refusera tous les événements de Bas. Si Haut a envoyé $h1$, P n'acceptera que $l1$ de la part de Bas, et si Haut a envoyé $h2$, Bas peut seulement envoyer $l2$. De cette manière Haut peut influencer trois comportements, transmettant ainsi $\log_2 3$ bits à Bas. C'est ainsi que Lowe quantifie les fuites d'information.

Dans ce même papier, Lowe propose aussi sa version de la non-interférence (appelée ici *Testing Non Deducibility on Composition* ou TNDC) :

$$P \texttt{ sat } TNDC \triangleq \forall Q_0, Q_1 \in CSP_H \cdot (P \parallel_H Q_0) \backslash H \equiv_T (P \parallel_H Q_1) \backslash H$$

où

- $\parallel_H$ est la composition parallèle synchronisée sur les événements de H
- CSP_H est l'ensemble des processus de niveau haut
- $P \backslash \Sigma$ est le processus P où l'on a *caché* les événements de Σ (on les a rendu internes)
- $\equiv_T$: $P_0 \equiv_T P_1$ si pour tout test $T \in CSP_L$, $P_0 \parallel_L T \backslash L$ et $P_0 \parallel_L T \backslash L$ permettent de lire la même information

Notons aussi, puisqu'il sera question de stratégies plus tard dans ce rapport, que Lowe discute des stratégies de Haut et Bas afin de maximiser le transfert d'informations : les processus Q_i matérialisant les différentes stratégies de Haut selon l'information qu'il souhaite transmettre, et T matérialisant la stratégie d'écoute de Bas.

1.6 La non-interférence caractérisée par un système de types

Une approche assez différente, proposée notamment par Volpano et Smith dans [16], puis raffinée plus tard par Boudol et Castellani dans [8] consiste à prouver la non-interférence directement par un système de typage adéquat.

L'idée de Volpano et Smith est de typer les variables suivant leur niveau de sécurité (les niveaux forment un ensemble partiellement ordonné), puis de typer les morceaux de programme de telle sorte qu'il soit impossible

- d'une part d'affecter une expression de type haut à une variable de type bas
- d'autre part d'affecter des variables de type bas dans une portion de programme gardée par une expression de type haut

On interdira par exemple, si l est une variable de type bas et h une variable de type haut, des expressions de la forme :

- $l := 2 * l + h$
- `if` $h = 0$ `then` $l := 1$ `else` $l := 2$

Volpano et Smith démontrent en outre qu'un programme bien typé vérifie trois propriétés (trois théorèmes) que l'on pourrait qualifier de non-interférence :

1. dans la sémantique naturelle « à grands pas » du langage (où un programme est vu comme un transformateur de mémoire), si
   - c est un programme de type τ dans le contexte,
   - μ et ν sont deux mémoires de même support (portant sur les mêmes variables),
   - et μ et ν sont en accord sur les valeurs des emplacements de types inférieurs à τ (dans le même contexte),

   alors soit c ne termine correctement ni sur μ ni sur ν, soit il termine correctement à partir des deux mémoires et les mémoires résultantes sont en accord sur les valeurs de tous les emplacements de types inférieurs à τ. C'est le théorème d'accord sur la terminaison.
2. dans la sémantique naturelle « à petits pas », théorème de progrès mutuel : le programme progresse sur μ si et seulement s'il progresse sur ν, donc le programme doit terminer ou bloquer de la même manière sur les deux mémoires. Dans l'accord sur la terminaison, on autorisait qu'un programme boucle sur μ tout en ayant une terminaison anormale sur ν.
3. toujours dans la sémantique « à petits pas», théorème d'accord sur le *timing* : version renforcée du précédent où l'on sait que l'on termine en le même nombre d'étapes, ce qui exclut les *timing channels* (mais on a renforcé la règle de typage du `if`).

Boudol et Castellani proposent, eux, un système de type un peu plus précis, où pour une commande, on type d'une part le type minimum des variables affectables et d'autre part le type maximum des gardes, pour un langage qui autorise en plus la composition parallèle.

Ils définissent ensuite une propriété, sur la sémantique opérationelle de leur langage d'exemple, qu'ils baptisent non-interférence. Leur premier théorème la garantit pour tout programme bien typé.

Dans la seconde partie de l'article, les auteurs élargissent le langage avec des constructions d'ordonancement des processus (un programme peut contrôler l'avancement d'un autre). Leur second théorème garantit la non-interférence dans le langage élargi, écartant ainsi la possibilté de *timing channels* dans un programme bien typé.

Chez Boudol et Castellani, la non-interférence s'exprime ainsi :

$$\forall \mu, \nu \quad \mu =_\Gamma^{\mathcal{L}} \nu \implies (P, \mu) \simeq_\Gamma^{\mathcal{L}} (P, \nu)$$

où

- μ, ν sont des mémoires
- P est un programme
- Γ est l'environnement de typage
- $\mathcal{L}$ est un ensemble de niveaux de sécurité clos vers le bas
- $=_\Gamma^{\mathcal{L}}$ est l'égalité de mémoires sur les variables typées dans Γ dans des niveaux de $\mathcal{L}$ ($(\Gamma, \mathcal{L})$-égalité)
- $\simeq_\Gamma^{\mathcal{L}}$ est la plus grande $(\Gamma, \mathcal{L})$-bisimulation

et où par $(\Gamma, \mathcal{L})$-bisimulation on entend une relation d'équivalence $\mathcal{R}$ sur les configurations (couples programme, mémoire) telle que $(P, \mu)\mathcal{R}(Q, \nu)$ implique

1. $\mu =_\gamma^{\mathcal{L}} \nu$
2. $(P, \mu) \to (P', \mu') \implies \exists Q', \nu'.(Q, \nu) \twoheadrightarrow (Q', \nu') \wedge (P', \mu')\mathcal{R}(Q', \nu')$
3. $(Q, \nu) \to (Q', \nu') \implies \exists P', \mu'.(P, \mu) \twoheadrightarrow (Q', \mu') \wedge (P', \mu')\mathcal{R}(Q', \nu')$

Autrement dit, deux configurations sont $(\Gamma, \mathcal{L})$-bisimilaires si leurs mémoires sont $(\Gamma, \mathcal{L})$-égales et leurs comportements sont similaires. Ainsi $\simeq_\Gamma^{\mathcal{L}}$ est la relation la plus fine vérifiant cette propriété et la non-interférence implique qu'aucune $(\Gamma, \mathcal{L})$-bisimulation ne peut distinguer les différents comportements d'un programme tant qu'il s'exécute sur des mémoires $(\Gamma, \mathcal{L})$-égales.

1.7 Autres approches basées sur les langages

Donner la liste exhaustive des modèles de sécurité des flots d'information serait difficile, tellement celle-ci est longue. Il se trouve qu'un certain nombre d'entre eux (ce qui est le cas des quelques dernières approches citées ici) se basent sur l'étude des langages (typage, analyse de programmes, etc.).

Néanmoins, dans [15], Sabelfeld et Myers nous offrent un véritable panorama sur le sujet en répertoriant et catégorisant un certain nombre de ces approches.

Il ressort de notre analyse que la non-interférence se résume souvent à un problème d'accessibilité, ce qui est indécidable en général. Il suffirait par exemple d'examiner un modèle exprimé en termes d'automates communicants asynchrones pour s'en rendre compte : dans un tel modèle on ne peut même pas décider si un message émis a bien été reçu. Comment dans ces conditions savoir s'il y a eu interférence ou non ?

De plus la non-interférence ne prend pas en compte la vivacité, où l'on entend par vivacité une propriété qui assurerait la réutilisation du canal un nombre arbitraire de fois. Pour les approches passées en revue, mettre en évidence une telle propriété n'était en effet pas le but, celui-ci étant simplement de dire si tel ou tel flot d'informations existe ou non. Pourtant cette propriété est essentielle pour les canaux cachés.

2 Scénarios, canaux cachés et jeux

Dans cette section, nous entrons dans le cadre plus précis de la manière dont on aborde le problème de la sécurité des flots d'information chez DISTRIBCOM, ainsi que dans le cadre du formalisme qui sera utilisé et adapté pendant le stage, à savoir celui des jeux.

2.1 Analyse de langages de scénarios

Nous introduisons ici un formalisme, celui des HMSC, qui malgré ses communications asynchrones et ses comportements infinis, présente l'avantage de rendre certaines propriétés de non-interférence décidables, ce qui est assez rare pour être signalé. C'est pour cela que nous pensons que les HMSC sont un bon modèle pour les canaux cachés. Un scénario (MSC : Message Sequence Chart) est un ordre partiel constitué d'événements qui ont lieu sur un certain nombre d'instances, ou processus. Les événements d'une même instance sont totalement ordonnés. Deux événements situés sur deux instances différentes peuvent aussi être ordonnés par une relation de type émission/réception, puis par transitivité.

Deux scénarios de même support (mêmes ensembles de processus) peuvent être concaténés en prolongeant la ligne de vie d'un processus du premier par sa ligne de vie dans le second. Le scénario vide est l'élément neutre pour la concaténation des scénarios. Ainsi, les scénarios munis de la concaténation forment un monoïde, et l'on parle de langages de scénarios.

L'objet qui sert le plus souvent à construire ces langages de scénarios est le HMSC (High-level MSC) : c'est un automate étiqueté par des MSC. Le langage d'un HMSC est naturellement obtenu en concaténant les étiquettes des chemins que l'on peut parcourir depuis le nœud initial.

Dans un HMSC, on appellera nœud à choix local un nœud dont tous les événements minimaux des étiquettes des arcs qui en partent sont situés sur la même instance. On dit alors que cette dernière contrôle le nœud.

Dans [12], on montre qu'il est possible de mettre en évidence des canaux cachés dans des spécifications de protocoles données par des HMSC.

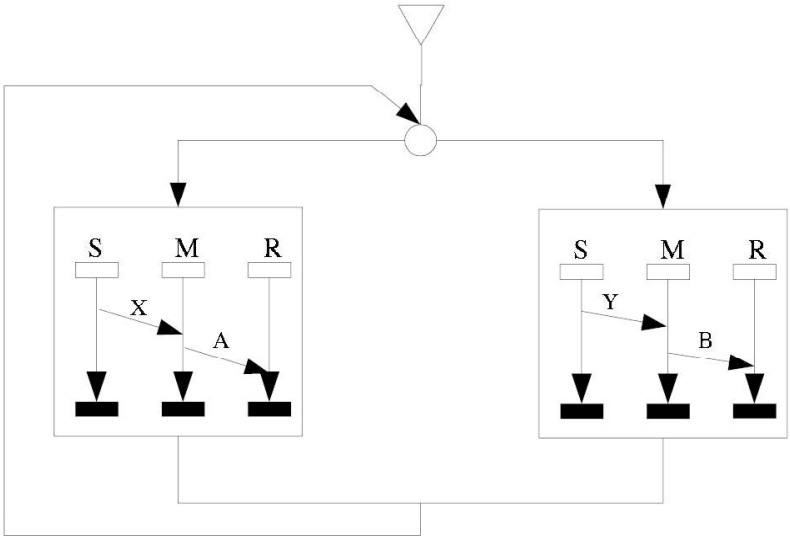

FIG. 1 – Exemple de HMSC dans lequel il existe un canal caché entre les instances S et R.

Par exemple, la figure 1 montre un HMSC où un tel canal caché existe. Ici, S communique de manière légitime avec M en envoyant des paquets de type X ou de type Y contenant des données légitimes elles aussi. M réagit en envoyant alternativement des paquets de type A ou B à R. Mais l'alternance de ces deux types de paquets permet à S de coder n'importe quel flux d'information illégitime par dessus les communications légitimes, en considérant par exemple que dans le canal caché, un paquet de type A code un 0 et un paquet de type B un 1. S dispose ainsi d'un canal illégitime avec R qu'il peut utiliser pour lui envoyer des messages arbitrairement longs.

Ce que l'on constate dans un tel exemple, artificiel, s'observe en fait aussi dans des protocoles réels. On a pu ainsi mettre en évidence un canal caché dans le protocole RMTP2 ([12]).

2.2 Généralités sur les jeux

Dans le deuxième chapitre de [11], René Mazaala définit les quelques notions de théorie des jeux qui nous intéressent ici

Arène Une arène est un graphe biparti (V_0, V_1, E) où V_0 et V_1 sont deux ensembles de sommets disjoints (on pose $V = V_0 \cup V_1$) et $E \in V_0 \times V_1 \cup V_1 \times V_0$ est un ensemble d'arcs appelé encore ensemble des coups jouables.

L'idée est que 0 et 1 sont des joueurs et que les deux sortes de sommets représentent les états où c'est à un joueur ou à l'autre de jouer (déplacer le jeton d'un sommet à un sommet successeur par E).

Partie Une partie est un chemin maximal dans l'arène : c'est à dire une succession de sommets se succédant, qui soit

- soit un chemin infini
- soit un chemin dont le dernier sommet n'a pas de successeur

Jeu Un jeu est un couple $(\mathcal{A}, \mathcal{W}in)$ où $\mathcal{A}$ est une arène et $\mathcal{W}in$ est un ensemble de parties infinies de $\mathcal{A}$ appelé ensemble gagnant.

Les parties d'un jeu sont les parties de son arène.

Une partie π est gagnante pour le joueur 0 si et seulement si $\pi \in \mathcal{W}in$ ou bien π est fini et son dernier sommet appartient à V_1 (ce serait au joueur 1 de jouer, mais celui-ci n'a plus de coup jouable).

Stratégie Soit une fonction partielle $f_\sigma : V^*V_\sigma \to V$. Une partie $\pi = v_0 v_1 \cdots$ est dit conforme à f_σ si pour tout i où $i \in \mathbb{N}$ et $v_i \in V_\sigma$, $f_\sigma(v_0 \cdots v_i)$ est défini et vaut v_{i+1}. f_σ est une stratégie sur U, sous-ensemble de V, si elle est définie sur tout préfixe de toute partie qui part de U et ne finit pas sur un sommet de V_σ sans coup jouable. Dans un jeu $\mathcal{G} = (\mathcal{A}, \mathtt{W}in)$, une stratégie gagnante sur U est une stratégie telle que toute partie qui part de U et qui lui est conforme est gagnante pour σ.

2.3 Un canal caché est un jeu

L'idée développée dans [13] est de transformer un HMSC en une arène.

On suppose que sur ce HMSC sont présentes trois instances : S, M et R. S comme *sender* est un utilisateur qui veut transmettre de l'information illégitime à un autre utilisateur R, comme *receiver*. M, comme *medium*, représente le reste des acteurs du protocole, modélisés en un protagoniste unique. On suppose aussi que tous les choix du HMSC sont locaux et contrôlés par S, R, ou M.

L'arène est, comme on l'a dit, donnée par le HMSC : un nœud de choix contrôlé par M est un nœud de V_0 et un n$\frac{1}{2}$ud contrôlé par S ou R est un n$\frac{1}{2}$ud de V_1.

Une partie gagnante pour le medium est une partie pour laquelle ce que peut observer R ne lui donne qu'une information finie, ce qui revient, pour le médium, à pouvoir forcer soit des comportements finis, soit des comportements périodiques. Autrement dit, dans le jeu du canal caché, $\mathcal{W}in = O^*p^\omega$ où O est l'ensemble des événements observables par R et $p \in O^*$ est un mot qui se répète indéfiniment.

L'absence de canal caché équivaut alors à l'existence d'une stratégie gagnante pour le médium.

Pour savoir s'il existe un canal caché, il s'agira de découper l'arène en deux zones : l'une où M peut empêcher quoiqu'il arrive le transfert d'information de S vers R, l'autre où S peut envoyer de l'information arbitrairement longue à R sans être irrémédiablement gêné par M. Autrement dit, on cherche les zones où il existe des stratégies gagnantes, soit pour M, soit pour S.

Le calcul de ces zones fait intervenir la décomposition de l'arène en composantes fortement connexes. Dans chacune de ces composantes, en commençant par les plus profondes, on vérifie s'il existe un nœud d'interférence ou non. Un nœud d'interférence est un nœud v contrôlé par S, tel que S peut soit forcer le retour en v par deux chemins distinguables par R (traces d'intersection vide), ou bien, au pire, peut être forcé par M à tomber dans une autre composante plus profonde où S est déjà gagnant. Les cas de base gagnants pour M sont les autres : à savoir les composantes où il n'existe pas de nœud d'interférence, et celles où M peut forcer le protocole à aller vers une composante déjà gagnante.

3 Conclusion

Dans la première partie de cette étude bibliographique nous avons vu différentes manières de modéliser des fuites d'information, notamment dans le cadre de la non-interférence. Nous

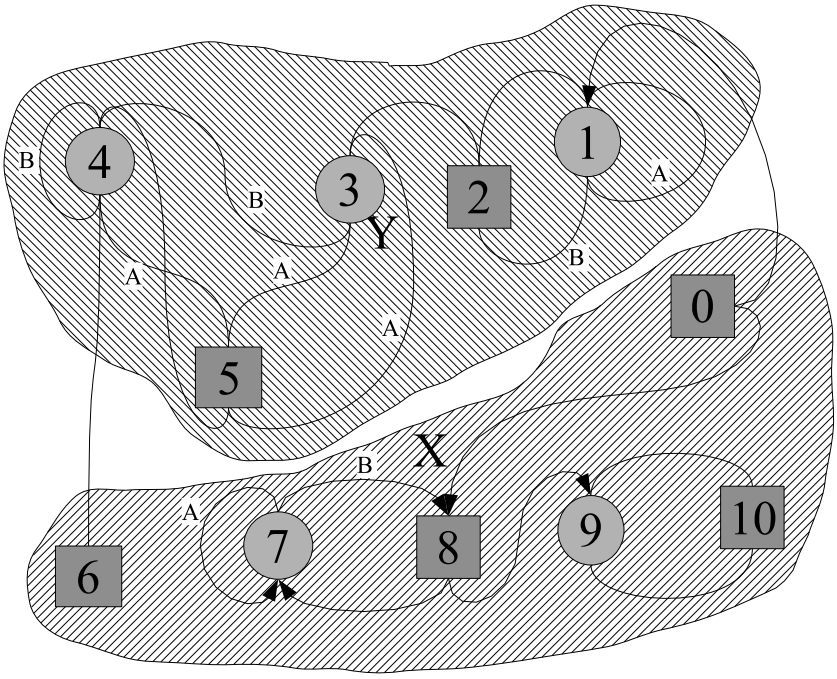

FIG. 2 – Les nœuds contrôlés par M sont représentés par des carrés, ceux contrôlés par S ou R sont représenté par des cercles. X représente la zone gagnante de M, Y celle de S.

avons vu aussi qu'il se trouvait de grandes ressemblances entre tous ces formalismes : toutes les propriétés de non interférence se ressemblent dans le sens où elles mettent en œuvre le même schéma logique, mais dans des modèles différents. C'est ce schéma que nous voudrions réussir à instancier dans un premier temps dans le cadre de la théorie des jeux, mais aussi sous la forme de propriétés exprimées dans une logique telle que le μ-calcul.

Partant de là, nous pourrions obtenir une caractérisation intéressante du jeu du canal caché, en tant qu'interférence « vivace », à savoir qu'un canal caché existe en un point de l'arène si et seulement si ce point est un point d'interférence et s'il vérifie une propriété de vivacité : l'attaquant peut toujours forcer le retour en ce point en un nombre fini d'étapes.

Si tous ces objectifs étaient atteints à la fin du stage, nous disposerions d'une méthode très générale pour détecter et documenter automatiquement des canaux cachés dans n'importe quelle spécification de système distribué, tant qu'on est capable de l'exprimer sous forme d'arène. Cette méthode pourrait de plus par la suite être adaptée pour rendre compte d'autres failles de sécurité que les canaux cachés tant qu'on peut les considérer comme des jeux entre l'attaquant et le reste du protocole.

Deuxième partie

Résultat obtenus pendant le stage

Ici commence le rapport de stage proprement dit, où nous énonçons les résultats auxquels il est fait allusion dans l'introduction de ce document. Nous verrons que contrairement à ce qui était envisagé à l'époque où l'étude bibliographique a été rédigée, les langages de scénarios ne seront évoqués qu'à titre anecdotique : à savoir que moyennant une certaine approximation, un système décrit dans un langage de scénario peut être traduit en automate afin de lui appliquer les méthodes expliquées ci-dessous. En revanche, la théorie des jeux, et surtout la logique temporelle occupent une place prépondérante.

4 Cadre : des automates aux arènes

Nous supposerons que le système de transitions soumis à notre analyse est la donnée d'un automate fini. Ce choix se justifie par son usage très répandu et par le fait que l'on puisse facilement obtenir de tels automates en transformant d'autres modèles, moyennant des hypothèses de finitude (par exemple : canaux bornés) ou éventuellement des approximations. Ainsi, ce travail pourra être adapté aux automates communiquants et autres IOLTS ou même aux langages de scénarios. Notons aussi que le fait de faire une approximation du système n'est pas vraiment gênant, puisque de toute manière on ne travaillera jamais que sur des modèles qui sont des abstractions de la réalité, et inévitablement il existe des canaux cachés que l'on ne pourra pas prendre en compte. Par exemple nous ne tenons absolument pas compte du temps, qui pourtant a aussi une influence sur les canaux cachés et peut même en être la source (dans la littérature, on parle de *timing channels*).

L'automate est censé représenter un système distribué sur plusieurs sites auquel participent plusieurs agents. Nous représenterons seulement les événements qui jouent un rôle dans l'interaction des différents acteurs avec le système : c'est à dire les envois et réceptions de messages. Tout ce qui est événement interne, qui n'a aucune influence sur l'observabilité ou le contrôle, sera abstrait du modèle.

4.1 Automate

Le modèle de base, que l'on complétera ensuite, est un automate à états finis, défini conformément à l'usage :

Définition 4.1 (Automate). Un automate est un quadruplet $\langle Q, q_0, \Sigma, \delta \rangle$ où
- Σ, Q sont des ensembles d'*événements* et d'*états*,
- $q_0 \in Q$ est l'*état initial*,
- $\delta \subseteq Q \times \Sigma \times Q$ est la *relation de transition*. Les éléments de δ sont appelés *transitions*.

L'automate est *déterministe* si δ est une relation fonctionnelle.

Seuls nous intéresseront les comportements maximaux, donc la notion d'état accepteur ne joue pas de rôle ici. Les mots acceptés sont donc les mots maximaux dont les préfixes finis respectent la relation de transition en partant de l'état q_0 : c'est-à-dire les mots (finis) associés aux chemins qui arrivent dans les états-puits ainsi que les mots associés aux chemins infinis.

À ce modèle, on ajoute de l'information d'observabilité ou de localisation des événements, ainsi qu'une certaine information de contrôle, avec l'idée qu'un agent contrôle et observe ce qu'il envoie, observe sans contrôler ce qu'il reçoit, et ne contrôle ni n'observe tout le reste des événements.

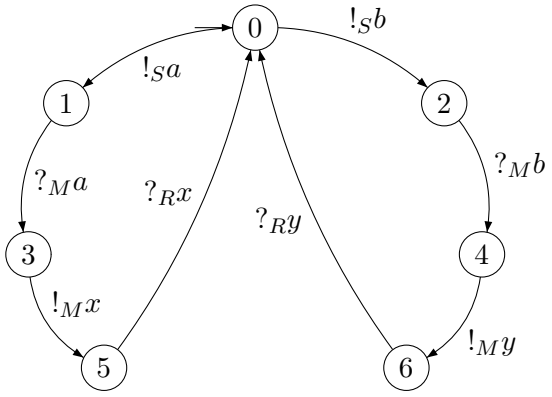

FIG. 3 – Exemple d'automate localisé : $\mathcal{A}$

Définition 4.2 (Automate localisé). Un automate localisé est un triplet $\langle \mathcal{A}, \Pi, \lambda \rangle$ où

- $\mathcal{A} = \langle Q, q_0, \Sigma, \delta \rangle$ est un automate
- Σ est de la forme $\{!, ?\} \times M$ où M est un ensemble de *messages*
- Π est un ensemble de processus , *joueurs* ou *agents* (ces termes sont pour nous équivalents)
- $\lambda : \Sigma \to \Pi$ est une *fonction de localisation*

Si $p \in \Pi$, un événement σ tel que $\lambda(\sigma) = p$ est dit *localisé* sur p ou *observable* par p. Une transition $t = (q, \sigma, q')$ est dite observable par p si et seulement si σ est observable.

Un événement observable par p, appartenant à $\{!\} \times M$ (noté $!m$ avec $m \in M$) est un envoi de message par p.

Un événement observable par p, appartenant à $\{?\} \times M$ (noté $?m$ avec $m \in M$) est une réception de message par p.

Si p est un joueur, on notera Σ^p l'ensemble $\lambda^{-1}(p)$ des événements observables par p.

On notera aussi respectivement $\Sigma^!$, $\Sigma^?$, $\Sigma^{p!}$, δ^p, $\delta^{p!}$ les ensembles $\{!\} \times M$, $\{?\} \times M$, $\Sigma^p \cap \Sigma^!$, $(Q \times \Sigma^p \times Q) \cap \delta$ et $(Q \times \Sigma^{p!} \times Q) \cap \delta$.

La figure 3 nous montre l'exemple que nous utiliserons tout au long de ce rapport, donné d'abord sous forme d'automate localisé. Cet exemple représente un système où dans l'état initial, un utilisateur S peut envoyer vers M, les messages a ou b et M réagit, après réception, en envoyant vers R, soit x, si c'est a qu'il a reçu, soit y, si c'est b qu'il a reçu. Après cela, le système revient dans l'état initial. Nous remarquons que ce système présente un canal caché « évident », dans la mesure où S peut faire observer à R n'importe quel mot de $(x + y)^*$ en fonction de la suite de choix a/b qu'il fait, et ce quelle que soit la réaction de l'environnement M (qui n'a en fait jamais aucun choix dans l'exemple).

4.2 Transformations d'automates localisés

Pour rendre exploitable l'automate proposé, on aura besoin de deux opérations particulières.

La première d'entre elle est la projection de l'automate sur un certain processus p. Le résultat voulu est un automate déterministe dont les états résument tout ce que p peut savoir de

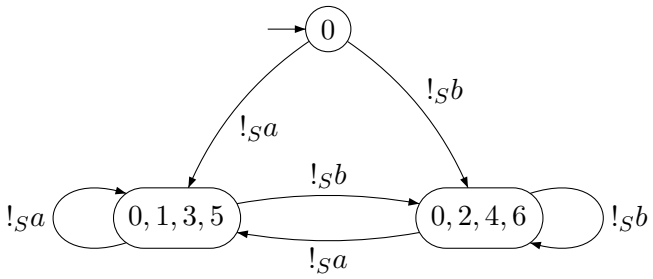

FIG. 4 – Projection sur S : $\mathcal{A}_{/S}$

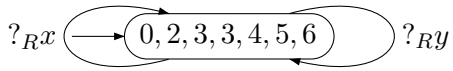

FIG. 5 – Projection sur R : $\mathcal{A}_{/R}$

l'état global du système en tenant compte de tous les événements qu'il a pu observer depuis le début, et uniquement de cela.

La seconde opération est le produit d'automates synchronisé sur un certain ensemble d'événements, dont la définition est celle que l'on trouve habituellement dans la littérature ([3]).

Définition 4.3 (Projection d'un automate localisé sur un processus). La projection sur un processus p d'un automate localisé $\langle \mathcal{A}, \Pi, \lambda \rangle$ est l'automate $\mathcal{A}_{/p} = \langle Q_{/p}, q_{0/p}, \Sigma^p, \delta_{/p} \rangle$ qui est la déterminisation de l'automate $\langle Q, q_0, \Sigma \cup \{\varepsilon\}, \delta' \rangle$ où $\delta' = \delta^p \cup \{(q, \varepsilon, q') | \exists \sigma \in \Sigma \backslash \Sigma^p, (q, \sigma, q') \in \delta\}$.

Remarquons qu'à cause de la déterminisation, $Q_{/p} \subseteq 2^Q$.

Les figures 4 et 5 montrent les projections de l'automate de la figure 3 sur les agents S et R. Observons que dans cet exemple, R ne sait jamais rien de l'état du système (sa projection a un seul état.).

Après projection sur p, on obtient l'automate qui reconnaît le langage d'événements observables depuis p quand le système s'exécute et qui permet à p à tout moment de déduire exactement l'ensemble d'état possibles que le système global a pu atteindre, étant données ses observations. Plus les états du projetés sont petits (en cardinal, puisque ce sont des ensembles d'états globaux), plus la connaissance est précise, et plus il sera facile pour p d'élaborer des stratégies pertinentes pour forcer certains comportements.

Notons néanmoins que l'ensemble des états de la projection ne forme pas une partition des états de l'automate du système, et que donc connaître l'état du système dans cet automate ne permet pas de savoir ce que p pense de cet état à coup sûr. Or cela est gênant dans le cadre des canaux cachés puisque le pirate qui veut envoyer de l'information va chercher à faire observer à l'autre pirate des occurrences précises d'événements, et aurait donc besoin d'en savoir plus sur ce que ce dernier sait. De plus, pour d'autres raisons, notamment pour la sémantique de la logique temporelle qui sera employée, il sera plus pratique de raisonner sur un automate partitionné selon l'information connue par les participants. C'est pour arriver à un tel automate que nous introduisons l'opération qui suit :

Définition 4.4 (Produit d'automates synchronisé sur un ensemble d'événements). Si $\mathcal{A}_1 = \langle Q_1, q_1^0, \Sigma_1, \delta_1 \rangle$ et $\mathcal{A}_2 = \langle 2, q_2^0, \Sigma_2, \delta_2 \rangle$ sont deux automates, leur produit synchronisé sur

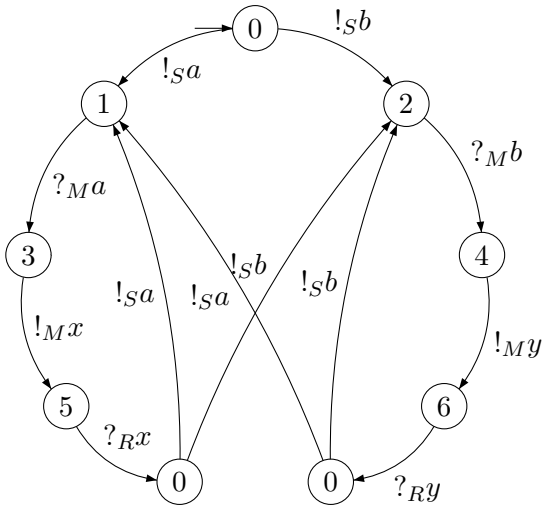

FIG. 6 – Produit synchronisé : $\mathcal{A}_{/S} \otimes_{\Sigma_S} \mathcal{A}$ (nous n'avons fait figurer sur les états que l'information relative aux états correspondants sur $\mathcal{A}$)

l'ensemble d'événements Σ_s est l'automate noté $\mathcal{A}_1 \otimes_{\Sigma_s} \mathcal{A}_2$ que l'on définit comme la partie accessible de l'automate $\langle Q_1 \times Q_2, (q_1^0, q_2^0), \Sigma_1 \cup \Sigma_2, \delta \rangle$ où

$$\begin{aligned}\delta = \{((q_1,q_2),\sigma,(q_1',q_2)) | (q_1,\sigma,q_1') \in \delta_1 \wedge q_2 \in Q_2 \wedge \sigma \notin \Sigma_s\} \\ \cup \{((q_1,q_2),\sigma,(q_1,q_2')) | (q_2,\sigma,q_2') \in \delta_2 \wedge q_1 \in Q_1 \wedge \sigma \notin \Sigma_s\} \\ \cup \{((q_1,q_2),\sigma,(q_1',q_2')) | (q_1,\sigma,q_1') \in \delta_1 \wedge (q_2,\sigma,q_2') \in \delta_2 \wedge \sigma \in \Sigma_s\}\end{aligned}$$

La figure 6 montre un exemple de produit. Il se trouve que dans cet exemple, le produit $\mathcal{A}_{/S} \otimes_{\Sigma_S} \mathcal{A}$ est isomorphe à $\mathcal{A}_{/S} \otimes_{\Sigma_S} \mathcal{A} \otimes_{\Sigma_R} \mathcal{A}_{/R}$.

L'intérêt de l'opération de produit est que si, comme dans l'exemple de la figure 6, on a fait le produit de l'automate du système avec l'automate projeté sur p, on sait alors que tout état du produit correspond à un unique état de la projection sur p. Autrement dit, se donner un état du produit suffit pour savoir à coup sûr ce que p sait du système quand il est dans cet état, et ce sans avoir à considérer la suite des transitions franchies depuis le début. Une conséquence est que cela vaut aussi pour la dernière transition observable franchie : si l'on vient de franchir une certaine transition observable par p sur le produit, on sait quelle transition p vient de franchir sur sa projection, et donc cela suffit pour savoir ce que p sait de la dernière transition franchie.

Toujours dans la figure 6, on remarque que le canal caché est toujours visible : on a trois états (marqués 0) à partir desquels S a un choix qui permet à R d'observer x ou y au bout d'un nombre fini de transitions.

4.3 Des automates aux arènes

Le problème des automates est que l'information concrètement utile ne se trouve pas dans l'état, mais sur la transition, car le franchissement d'une transition s'accompagne toujours de l'émission ou de la réception d'un message. En effet, au fond, c'est bien grâce aux messages qu'un canal caché peut exister et, à terme, on voudra caractériser le problème du canal caché par un jeu dans lequel, pour gagner, un émetteur doit être capable d'envoyer n'importe quel message. Or, dans la théorie des jeux, on a tendance à utiliser le formalisme des arènes où

l'information se trouve associée aux nœuds du graphe par le biais d'un étiquetage par des propositions logiques, et non sur les arêtes.

Ainsi, pour continuer à travailler, nous choisissons de traduire l'automate en arène. Notons que les logiques temporelles se prêtent très bien à l'étude des arènes.

La première étape sera en quelque sorte de passer au graphe dual, où les nœuds deviennent les arêtes et vice-versa, ce qui nous amène à un graphe que nous appellerons *semi-arène*. La différence entre ce type de graphe et une arène sera qu'on ne peut pas associer une information de contrôle bien définie à une position (à un nœud) puisqu'il existe plusieurs coups jouables contrôlés par des joueurs (agents) différents depuis une position.

Ainsi, la seconde étape consistera à séparer les coups jouables de telle sorte qu'à partir d'une position, l'initiative ne puisse venir que d'un seul joueur (jeu basé sur un système de tours). Pour ce faire, nous introduisons un joueur supplémentaire η_S (*ordonnanceur*) qui contrôle toutes les positions à contrôle ambigu et donne le contrôle à l'un des joueurs qui aurait dû pouvoir jouer (moyennant la duplication de ces positions).

Définition 4.5 (Semi-arène). Une semi-arène est un triplet $\langle \Pi, V, (E_p)_{p\in\Pi}\rangle$ où
- Π est un ensemble de *joueurs*,
- V est un ensemble de *positions*,
- $\forall p \in \Pi, E_p \subseteq V \times V$ est un ensemble d'arêtes représentant les *coups jouables* du joueur p. Si $p \neq p'$ alors E_p et $E_{p'}$ sont disjoints.

Une position v de laquelle ne partent que des flèches de E_p est dite *contrôlée* par le joueur p ($p \triangleright v$). Une position de laquelle partent les deux sortes de flèches n'est pas contrôlée ($\not\triangleright v$).

Définition 4.6 (Arène). Une arène est un triplet $\langle \Pi, (V_p)_{p\in\Pi}, E\rangle$ où
- Π est un ensemble de *joueurs*,
- $\forall p \in \Pi, V_p$ est un ensemble de *positions* qui sont dites *contrôlés* par le joueur p. Si $p \neq p'$ alors V_p et $V_{p'}$ sont disjoints.
- $E \subseteq V \times V$ (avec $V = \bigcup_{p\in\Pi} V_p$) est un ensemble d'arêtes représentant tous les coups jouables de l'arène.

On appellera *run* un chemin maximal de positions successives (tel que deux positions consécutives soient un coup jouable). Le *langage reconnu* par une arène à partir d'une position est l'ensemble des runs qui en partent.

Nous souhaitons traduire un système sous forme d'automate en arène étiquetée. Nous allons pour cela donner la transformation d'un automate en semi-arène équivalente (pour les langages), puis donner une transformation des semi-arènes en arènes.

Théorème 4.7 (Traduction d'un l'automate localisé en semi-arène). *Soit $\mathcal{A} = \langle\langle Q, s, \Sigma, \delta\rangle, \Pi, \lambda\rangle$ un automate localisé.*

Le triplet $\langle \Pi \cup \{\eta_N\}, \delta \cup \{\bot\}, (E_p)_{p\in\Pi\cup\{\eta_N\}}\rangle$ avec

$$\begin{aligned}\forall p \in \Pi, E_p = &\quad \{((q,\sigma,q'),(q',!m,q'')) \in \delta\times\delta \mid \lambda(!m) = p\}\\ &\cup \{(\bot,(q_0,!m,q')) \mid (q_0,!m,q') \in \delta \wedge \lambda(!m) = p\}\\ E_{\eta_N} = &\{((q,\sigma,q'),(q',?m,q'')) \in \delta\times\delta\}\end{aligned}$$

est une semi-arène.

De plus cette semi-arène garde toute l'information contenue dans l'automate localisé $\mathcal{A}$. En outre le langage des chemins initiaux de sommets (chemins qui partent de la position $\bot$) de la semi-arène est le langage des chemins initiaux de transitions de l'automate.

L'idée intuitive de cette transformation est que dans la semi-arène ainsi construite, la position courante représente la dernière transition tirée sur l'automate.

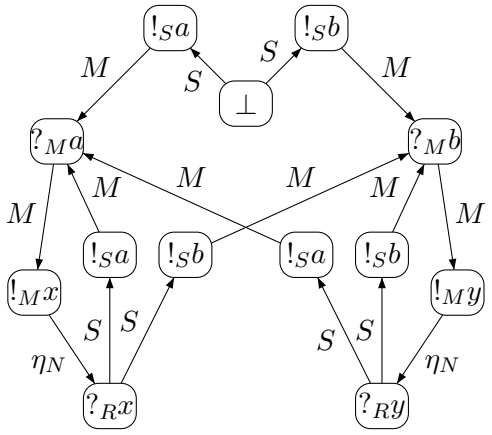

FIG. 7 – Semi-arène de l'automate $\mathcal{A}_{/R} \otimes_{\Sigma_R} \mathcal{A} \otimes_{\Sigma_S} \mathcal{A}_{/S}$. Les étiquettes sur les flèches indiquent qui contrôle les coups jouables.

Démonstration.

L'objet construit est une semi-arène.

Égalité des langages : par induction

- Le mot vide est reconnu dans les deux cas. On se trouve sur $\bot$ dans la semi-arène et sur q_0 dans l'automate.
- Si $t = (q_0, \sigma, q')$ est reconnu dans l'automate, par définition $(\bot, t)$ est un coup jouable depuis $\bot$ dans la semi-arène, donc t est y reconnu. Réciproquement, si t est reconnu dans la semi-arène, c'est que $(\bot, t)$ est un coup jouable, et que donc par définition t est une position de la forme (q_0, σ, q'). Ainsi il existait une transition (q_0, σ, q') dans l'automate et t était un mot de transitions reconnu.
- Soit v, mot de longueur non nulle, reconnu dans les deux modèles. Si la dernière lettre est $t = (q, \sigma, q')$, on se trouve en (q, σ, q') dans la semi-arène et en q' dans l'automate. Si $t' = (q', \sigma', q'')$ existe dans l'automate, alors, par définition, t' est une position de la semi-arène, et (t, t') est aussi par définition un coup jouable, donc si $v \cdot t$ est reconnu dans l'automate il l'est aussi dans la semi-arène. Réciproquement, si t' n'existait pas comme transition de l'automate, on ne l'aurait pas ajoutée à l'ensemble des positions de la semi-arène.

□

La figure 7 montre la semi-arène obtenue en transformant l'automate de la figure 6 par ce procédé.

Théorème 4.8 (Traduction d'une semi-arène en arène). *Si $\langle \Pi, V, (E_p)_{p\in\Pi}\rangle$ est une semi-arène, alors $\langle \Pi \cup \{\eta_S\}, (V_p)_{p\in\Pi\cup\{\eta_S\}}, E\rangle$ est une arène, si on la construit comme suit :*

- *$\forall p \in \Pi$, V_p contient les positions $v \in V$ telles que $p \triangleright v$, plus, pour chaque position v telle que $\not\triangleright v$, un nouveau sommet $v_{/p}$.*
- *V_{η_S} contient toutes les positions $v \in V$ telle que $\not\triangleright v$. Ainsi les positions qui n'étaient pas contrôlées dans la semi-arène deviennent les positions contrôlées par l'ordonnanceur.*

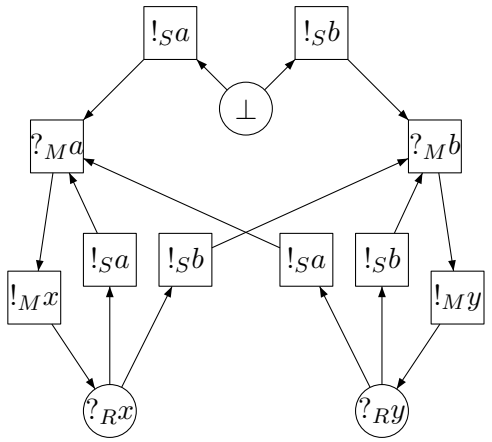

FIG. 8 – Arène obtenue à partir de la semi-arène de la figure 7. Les positions représentées par des cercles sont contrôlées par S.

– *et l'ensemble des coups jouables est*

$$E = \bigcup_{p \in \Pi \cup \{\eta_S\}} (((V_p \times V) \cap E_p) \cup \{(v, v_{/p}) \mid \not\triangleright v\} \cup \{(v_{/p}, v') \mid \not\triangleright v \wedge (v, v') \in E_p\})$$

avec la convention $E_{\eta_S} = \emptyset$ si $\eta_S \notin \Pi$.

De plus, l'ensemble des runs de cette arène est l'ensemble des runs de la semi-arène, à condition d'effacer des mots les lettres $v_{/p}$ où $p \in \Pi$.

Démonstration.

L'objet construit est une arène.

L'équivalence des langages modulo les décisions de l'ordonnanceur se prouve par induction. □

La figure 8 montre la semi-arène obtenue en transformant l'automate de la figure 7 par ce procédé. Ici, nous n'avons pas eu besoin d'ajouter de positions, l'information de contrôle n'ayant été ambiguë nulle part sur la semi-arène.

La figure 9 illustre, en revanche, ce qui se passe quand une position v n'est pas contrôlée : celle-ci devient une position contrôlée par l'ordonnanceur η_S qui peut jouer vers une des deux nouvelles positions $v_{/p_1}$ et $v_{/p_2}$ contrôlées par un des deux agents p_1 ou p_2 qui contrôlent les coups jouables depuis v dans la semi-arène. p_1 ou p_2 peut ensuite jouer son coup depuis cette dernière position.

Définition 4.9 (Position observable). Dans une semi-arène ou arène obtenue à partir d'un automate par le procédé ci-dessus, une position $v \in V$ est dite *observable* par un joueur p si v est de la forme (q, σ, q') avec $\lambda(\sigma) = p$.

Définition 4.10 (Arène étiquetée). Une arène étiquetée est un quadruplet $\langle \mathbb{A}, L, (L_p)_{p \in \Pi}, \omega \rangle$ où

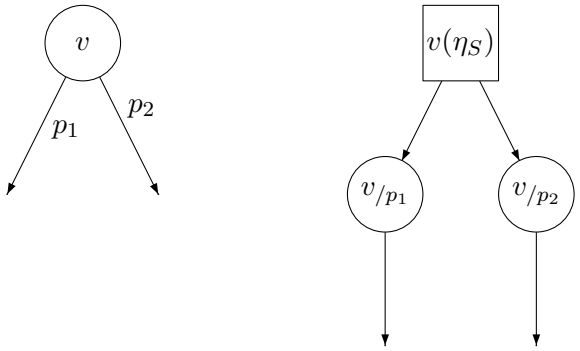

FIG. 9 – Séparation d'une position non contrôlée en plusieurs positions dans la transformation d'une semi-arène en arène (à gauche une partie de semi-arène, à droite, la partie d'arène correspondante : v est une position de la semi-arène, p_1 et p_2 sont deux agents différents).

- $\mathbb{A}$ est une arène,
- L un ensemble d'étiquettes,
- pour tout $p \in \Pi$, $L_p \subseteq L$ est l'ensemble des étiquettes observables par le joueur p,
- et $\omega : L \to 2^V$ est une fonction qui associe un ensemble de positions à chaque étiquette.

L'information sur l'arène est dite *complète* pour un joueur p si $\forall v_1, v_2 \in V$, $\exists l \in L_p$ tel que $v_1 \in \omega(l)$ et $v_2 \notin \omega(l)$, autrement dit si p est capable de distinguer toutes les positions.

Si $\forall l \in L_p$, on a $v_1 \in \omega(l)$ si et seulement si $v_2 \in \omega(l)$ on notera $v_1 \simeq_p v_2$ et on dira que v_1 et v_2 sont *indistinguables* pour p. Cette relation est une relation d'équivalence. On notera $[v]_p$ la classe d'équivalence de v par $\simeq_p$. On notera $V_{/\simeq_p}$ l'ensemble des classes de positions pour $\simeq_p$ (l'ensemble quotient de V par $\simeq_p$).

On étendra ces notations aux mots de positions : $w = v_1 \cdot v_2 \cdot \cdots \cdot v_n \simeq_p v'_1 \cdot v'_2 \cdot \cdots \cdot v'_n = w'$ si et seulement si en posant $v_{j_1} \cdot \cdots \cdot v_{j_m}$, la restriction de w aux positions observables par p et $v'_{j'_1} \cdot \cdots \cdot v'_{j'_{m'}}$, celle de w', on a $m = m'$ et $\forall i \leq m, v_{j_i} \simeq_p v'_{j'_i}$.

Si $p \in \Pi$, un ensemble de positions W de l'arène est dit *compatible* avec $\simeq_p$ si $v \in W \implies [v]_p \subseteq W$.

5 Logique : le μ-calcul à temps alternant

Dans cette section, nous allons introduire la logique temporelle qui nous permettra de caractériser la présence d'un canal caché sur une arène. Pour exprimer la notion de stratégie contre des joueurs adverses, nous avons besoin d'une logique à temps alternant telle que celles introduites dans [2]. Les logiques temporelles à temps alternant sont des logiques temporelles dans lesquelles les quantificateurs de chemins sont munis d'une expressivité plus forte que simplement « Il existe un chemin tel que... » ou « Tous les chemins sont tels que... » car intervient la notion d'agents qui permet de dire des propriétés comme « Il existe un ensemble de décisions des agents de l'ensemble P tel que tous les chemins qui tiennent compte de ces décisions sont tels que... ». Ainsi grâce à ces logiques, on peut exprimer l'existence ou non de stratégies coopératives pour une équipe d'agents donnée face à une certaine adversité (les autres agents).

La logique que nous utiliserons sera une variante du μ-calcul à temps alternant qui prendra en compte la notion d'observation partielle du système.

5.1 μ-calcul à temps alternant

Le μ-calcul à temps alternant (AMC) est une extension du μ-calcul, dans laquelle on a juste modifié le quantificateur « à l'instant suivant on a φ » ($\langle\rangle\varphi$), pour le remplacer par « à l'instant suivant, les joueurs de P peuvent forcer φ malgré les décisions des autres agents » ($\langle\langle P\rangle\rangle\varphi$).

5.1.1 Syntaxe

Les formules d'AMC sont construites comme suit :
- l où l est une étiquette ou proposition
- X où X est une variable propositionnelle
- $\neg\varphi$, la négation de la formule φ
- $\varphi_1 \vee \varphi_2$, la disjonction des formules φ_1 et φ_2
- $\langle\langle P\rangle\rangle\bigcirc\varphi$, « les processus de P peuvent faire en sorte que la formule φ soit vraie à l'instant suivant »
- $\mu X.\varphi$ où les occurrences libres de X dans φ apparaissent sous la portée d'un nombre pair de négations, représentant le plus petit point fixe de la formule φ, fonction de X.

On s'autorisera aussi les raccourcis syntaxiques suivants :
- $\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2)$, la conjonction des formules φ_1 et φ_2
- $[\![P]\!]\bigcirc\varphi \equiv \neg\langle\langle P\rangle\rangle\bigcirc\neg\varphi$, « φ est vraie à l'instant suivant malgré tous les choix des processus de P »
- $\exists \equiv \langle\langle\Pi\rangle\rangle$, « φ est possible à l'instant suivant »
- $\forall \equiv [\![\Pi]\!]$, « φ est vraie à l'instant suivant »
- $\nu X.\varphi \equiv \neg\mu X.\neg\varphi$ où les occurrences libres de X dans φ apparaissent seulement sous la portée d'un nombre pair de négations, représentant le plus grand point fixe de φ, fonction de X.

Définition 5.1 (Variables liées et libres, formules closes). Une variable propositionnelle X sera dite *liée* si elle est sous la portée d'un opérateur $\mu X.$ ou $\nu X.$ et *libre* sinon. Une formule d'AMC est dite *close* si toutes ses variables sont liées.

5.1.2 Sémantique

Définition 5.2. Étant donnée une arène étiquetée $\mathbb{A} = \langle\Pi, (V_p)_{p\in\Pi}, E, L, (L_p)_{p\in\Pi}, \omega\rangle$, une valuation est une fonction de l'ensemble des variables propositionnelles dans les parties de V.

Si $\mathcal{V}$ est une valuation, X une variable, et V' un ensemble de positions, on dénotera $\mathcal{V}[X := V']$ la valuation qui à X associe V' et à toute autre variable Y associe $\mathcal{V}(Y)$.

On interprétera une formule d'AMC φ comme une application $\varphi^{\mathbb{A}}$ des valuations aux ensembles de positions.

Cette application est définie par induction, comme suit :

1. Pour une étiquette $l \in L$, on a $l^{\mathbb{A}}(\mathcal{V}) = \omega(l)$.
2. Pour une variable X, on a $X^{\mathbb{A}}(\mathcal{V}) = \mathcal{V}(X)$.
3. $(\neg\varphi)^{\mathbb{A}}(\mathcal{V}) = V\backslash\varphi^{\mathbb{A}}(\mathcal{V})$
4. $(\varphi_1 \vee \varphi_2)^{\mathbb{A}}(\mathcal{V}) = \varphi_1^{\mathbb{A}}(\mathcal{V}) \cup \varphi_2^{\mathbb{A}}(\mathcal{V})$.
5. $$(\langle\langle P\rangle\rangle \bigcirc \varphi)^{\mathbb{A}}(\mathcal{V}) = \bigcup_{p\in P}\{v \in V_p | \exists (v,v') \in E, [v']_p \subseteq \varphi^{\mathbb{A}}(\mathcal{V})\} \cup \{v \in V\backslash V_P | \forall (v,v') \in E, v' \in \varphi^{\mathbb{A}}(\mathcal{V})\}$$

6. $(\mu X.\varphi)^{\mathbb{A}}(\mathcal{V}) = \bigcap\{V' \subseteq V \mid \varphi^{\mathbb{A}}(\mathcal{V}[X := V']) \subseteq V'\}$.

Ainsi on s'interdit (règle de $\langle\langle p\rangle\rangle \bigcirc \varphi$) de considérer comme une stratégie valable le fait de jouer un coup (v, v') alors qu'il n'y a aucun moyen de savoir si on n'est pas en train de jouer un autre coup équivalent (v'', v''') avec $v' \simeq_p v'''$ sans que v''' ne soit aussi une destination acceptable ($\mathbb{A}, v''' \models \varphi$).

Notons toutefois que nous proposons ici une variante d'AMC et que dans la sémantique classique d'AMC, l'information est totale, c'est-à-dire que pour tout agent p, $V_{/\simeq_p} \simeq V$. Ainsi la règle 5 y est moins restrictive et bien plus simple. Mais pour les canaux cachés, il n'était pas pertinent de considérer une logique où tous les agents savent tout ce qui se passe dans le système, auquel cas la notion de canal caché perdrait beaucoup de son intérêt.

Définition 5.3 (Notations).

On dira que deux formules φ_1 et φ_2 sont *équivalentes* dans l'arène $\mathbb{A}$ si $\varphi_1^{\mathbb{A}} = \varphi_2^{\mathbb{A}}$. On notera dans ce cas $\varphi_1 \equiv_{\mathbb{A}} \varphi_2$.

On dira qu'une position v de l'arène $\mathbb{A}$ *satisfait* la formule close φ si $v \in \varphi^{\mathbb{A}}$. On notera $\mathbb{A}, v \models \varphi$.

Si φ est une formule close, vu que la fonction $\varphi^{\mathbb{A}}$ ne dépend pas de la valuation passée en paramètre, on s'autorisera à noter $\varphi^{\mathbb{A}}$ l'ensemble $\varphi^{\mathbb{A}}(\mathcal{V})$.

5.2 Décidabilité du model-checking d'AMC avec information incomplète

Une formule d'AMC dans le cas de l'information complète peut être model-checkée en temps polynomial dans la taille de l'arène, et exponentiel dans la profondeur de la formule.

Cependant, il faut noter qu'il existe de nombreux résultats d'indécidabilité pour les logiques à temps alternant avec information incomplète (liés à l'indécidabilité des jeux d'équipe avec observation partielle). Nous allons voir qu'AMC appliqué aux arènes étiquetées finies ne pose pas ce problème : en effet la seule règle où l'information incomplète intervient est celle de $\langle\langle\cdot\rangle\rangle \bigcirc \cdot$, or il n'est pas difficile, quand on veut savoir si $v \in (\langle\langle P\rangle\rangle \bigcirc \varphi)^{\mathbb{A}}$ de vérifier algorithmiquement s'il existe un coup jouable (v, v') tel que $v \in V_p$ et $v' \in \varphi^{\mathbb{A}}$, avec $p \in P$, et que d'autre part, et c'est là la nouveauté, toute la classe $[v']_p$ est bien dans $\varphi^{\mathbb{A}}$.

Algorithme de model-checking : on calcule l'ensemble des positions vérifiant une certaine formule d'AMC close.

- Cas φ est de la forme l, X, $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$: on calcule directement $\varphi^{\mathbb{A}}(\mathcal{V})$ à partir de $\varphi_1^{\mathbb{A}}(\mathcal{V})$ et $\varphi_2^{\mathbb{A}}(\mathcal{V})$
- Cas $\varphi = \langle\langle P\rangle\rangle \bigcirc \varphi_1$: les classes $[v]_p$ étant connues, le nombre de positions et de coups jouables par position étant finis, on parcourt V en vérifiant directement si on peut ajouter ou non chacune des positions en fonction de $\varphi_1^{\mathbb{A}}(\mathcal{V})$.
- Cas $\varphi = \mu X.\varphi_1$: on pose $h : W \subseteq V \mapsto \varphi_1^{\mathbb{A}}(\mathcal{V}[X := W])$ et on calcule l'ensemble recherché par approximations successives : $\varphi^{\mathbb{A}}(\mathcal{V}) = \bigcup_{i\geq 0} h^i(\emptyset)$.

Théorème 5.4. *Cet algorithme converge.*

Démonstration. Ceci se montre par induction sur la profondeur de la formule :

Profondeur 0 : on a une proposition ou une variable, et le premier cas décide tout de suite.

Profondeur $n+1$: dans tous les cas, on ne fait des appels que sur des sous-formules (profondeur $\leq n$). Dans les 2 premiers cas, on fait un nombre borné d'appels (0,1 ou 2). Dans le dernier cas, on a un nombre d'approximations finies car l'approximation est croissante et le nombre de positions dans l'arène, fini. □

Théorème 5.5. *Les ensembles calculés sont bien ceux spécifiés par la sémantique.*

Démonstration.

Dans les deux premiers cas, on ne fait que calculer directement ce qui est spécifié dans la sémantique.

La définition de la sémantique donne, pour le troisième cas, $(\mu X.\varphi)^{\mathbb{A}}(\mathcal{V}) = \bigcap\{V' \subseteq V \mid \varphi^{\mathbb{A}}(\mathcal{V}[X := V']) \subseteq V'\} = \bigcap\{V' \subseteq V \mid h(V') \subseteq V'\}$ soit le plus petit point fixe de h. Or, si on pose $W \triangleq \varphi^{\mathbb{A}}(\mathcal{V}) = \bigcup_{i\geq 0} h^i(\emptyset)$, on a bien $h(W) \subseteq W$. Donc $(\mu X.\varphi)^{\mathbb{A}}(\mathcal{V}) \subseteq W$. Or la fonction h est monotone. Ainsi, si on se donne W', point fixe de h, on a $W = \bigcup_{i\geq 0} h^i(\emptyset) \subseteq \bigcup_{i\geq 0} h^i(W') = W'$. Donc W est le plus petit point fixe. D'où $W = (\mu X.\varphi)^{\mathbb{A}}(\mathcal{V})$. □

Nous verrons à la fin de ce rapport que cette logique, munie de cette sémantique, n'est pas assez puissante pour exprimer toutes les stratégies : il se pourra que la formule dont on croit que la signification est « Il existe une stratégie pour arriver dans W. » soit fausse en une position, mais qu'une telle stratégie existe pourtant. Ceci n'est guère étonnant, puisque les jeux distribués sont indécidables en général [4], et que cette logique est, elle, décidable.

5.3 Syntaxe « maison »

Pour des raisons de concision, le langage logique utilisé par la suite sera celui défini par les constructions ci-dessous :

- l
- $\neg\varphi$
- $\varphi_1 \vee \varphi_2$
- $\varphi_1 \wedge \varphi_2$
- $\langle\langle P\rangle\rangle \bigcirc \varphi$
- $\langle\langle P\rangle\rangle \varphi_1 \mathcal{U} \varphi_2$
- $\nu X.\varphi$

où l est une étiquette ou proposition, φ, φ_1, φ_2 sont des formules, P est un ensemble de joueurs, X est un symbole de variable.

Toutes les constructions ci-dessus existant dans AMC gardent leur sémantique. La construction $\langle\langle P\rangle\rangle \varphi_1 \mathcal{U} \varphi_2$ est un raccourci syntaxique pour $\mu X.(\varphi_2 \vee (\varphi_1 \wedge \langle\langle P\rangle\rangle \bigcirc X))$ et qui a pour signification « les processus de P peuvent faire en sorte que l'on reste sur un chemin où φ_1 est vérifée *jusqu'à ce que* φ_2 le soit ».

Ainsi, toute proposition écrite dans cette syntaxe pourra être interprétée comme une formule d'AMC.

Notons que cette syntaxe est celle de l'*Alternating-time Temporal Logic* (ATL), une extension à temps alternant de CTL introduite dans [2], à laquelle nous aurions adjoint l'opérateur de point fixe ν.

6 Interférence

Dans cette partie, nous expliquons et justifions comment on peut modéliser la propriété d'interférence grâce aux formalismes des sections 4 et 5 : c'est-à-dire par une formule d'AMC à vérifier sur une arène obtenue à partir d'un automate.

6.1 Préparation du problème

On suppose que le système est décrit par $\mathcal{A}$ un automate localisé avec $\Pi = \{S, M, R\}$, où S (sender) est un utilisateur qui souhaite envoyer à R (receiver) un message au moyen d'un protocole dont ils auront convenu au préalable. M (medium) représente l'environnement ainsi que tous les joueurs qui ne sont ni S ni R (M n'est pas un vrai joueur, mais une abstraction).

On va transformer cet automate en arène étiquetée comme suit :

1. Calculer les projections $\mathcal{A}_S$ et $\mathcal{A}_R$
2. Calculer le produit $\mathcal{A}'\mathcal{A}_S \otimes_{\Sigma_S} \mathcal{A} \otimes_{\Sigma_R} \mathcal{A}_R$ synchronisé sur les actions localisées en S pour le premier produit, en R pour le second. Ceci permet de raisonner par la suite sur un automate où chaque état correspond à un état précis sur les projections sur S et R du système initial (partitionnement des états suivant la connaissance des agents S et R).
3. Transformer l'automate obtenu en semi-arène, puis en arène selon le procédé décrit plus haut. L'arène obtenue sera notée $\mathbb{A}(\mathcal{A})$.
4. Choisir comme étiquetage $L_S = \{l_{S,t_S} | t_S \in \delta_{/S}\}$, $L_M = \emptyset$ et $L_R = \{l_{R,t_R} | t_R \in \delta_{/R}\}$ tels que

$$\forall (q_S, \sigma, q'_S) \in \delta_{/S}, \omega(l_{S,(q_S,\sigma,q'_S)}) = \{t \in \delta_{produit} | t = ((q_S, q, q_R), \sigma, (q'_S, q', q'_R))\}.$$

et

$$\forall (q_R, \sigma, q'_R) \in \delta_{/R}, \omega(l_{R,(q_R,\sigma,q'_R)}) = \{t \in \delta_{produit} | t = ((q_S, q, q_R), \sigma, (q'_S, q', q'_R))\}.$$

Rappelons que les transitions du produit deviennent des positions de l'arène. Ainsi, pour S (resp. R), nous avons étiqueté chaque position observable par la dernière transition observée par S (resp. R).

On pourrait reprocher à cet étiquetage de ne jamais rendre distinguables deux positions inobservables, alors que pourtant, en se rappelant la dernière transition observée, on pourrait reporter un peu plus d'information à propos de l'état courant sur l'étiquette d'une position inobservable.

Cette « perte » d'information n'est pas gênante car, de toute façon quand un choix se présente, la sémantique d'AMC stipule que l'on ne considère que l'information de la position vers laquelle on va (qui contient aussi l'état d'origine), qui est forcément observable, vu qu'on vient de faire un choix et que cette position correspond à une transition tirée par l'agent qui fait le choix.

Pour ce qui est de l'application réelle de stratégies, le pirate n'aura qu'à se rappeler la dernière transition observée pour faire son choix.

Nous travaillerons donc sur l'arène étiquetée $\langle \mathbb{A}(\mathcal{A}), L_S \cup L_M \cup L_R, (L_S, L_M, L_R), \omega \rangle$.

6.2 Interférence

Ici, on dit que S interfère avec R si, quels que soient les événements déclenchés par quiconque qui n'est ni S ni R, S est en mesure de faire observer deux ensembles disjoints de comportements à R.

Notons qu'on aurait pu choisir une définition duale : S interfère avec R si M ne peut pas empêcher à coup sûr S de faire observer deux ensembles de comportements disjoints à R. Du fait que M est une abstraction du système, les deux définitions ne sont pas équivalentes : il se pourrait en effet que les différents agents réels qui composent M ne puissent pas se mettre d'accord sur une stratégie coopérative de non-interférence et que pourtant, par une chance extraordinaire, arrivent toujours à empêcher S et R de communiquer.

Cette dernière définition constitue certainement une direction de recherche valide, mais c'est la première que nous retiendrons ici.

Revenons à la définition. Qu'entendons nous par « comportements » ? Un comportement observable ne peut raisonnablement qu'être une suite d'événements observables. Donc pouvoir faire observer deux ensembles de comportements disjoints, c'est faire observer à R deux langages de chemins disjoints L_0 et L_1. Admettons que R observe un mot w qui est dans l'union de ces deux langages et qu'il cherche à savoir si w est dans L_0 ou L_1. Comme on se restreint au seul cas utile où l'on veut transmettre un bit en un temps fini, il existe une certaine longueur l de préfixe au delà de laquelle tout préfixe d'un mot de L_0 est différent du préfixe de même longueur d'un mot de L_1. Donc il suffit de déplier un automate ($\mathcal{A}_{/R}$) sur une profondeur d'au plus l pour savoir si $w \in L_0$ ou $w \in L_1$. Dans tous les cas, il existera une première transition (occurrence d'événement) t_0 ou t_1 dans w qui permettra de dire que $w \in L_0$ ou que $w \in L_1$.

Dans la pratique, on ne dépliera pas d'automate, considérant que si on peut interférer grâce à deux ensembles de transitions dans le dépliage, alors on peut aussi interférer en considérant deux ensembles de transitions dans l'automate non déplié.

En effet, considérer des chemins sans cycle devrait suffire (idée de preuve : si deux chemins différaient d'un cycle seulement, si les deux étaient dans L_0 et le choix appartenait à S de faire boucler ou non, il pourrait décider de ne jamais boucler, et si le choix appartenait à M, il pourrait décider de boucler indéfiniment et le mot pourrait ne jamais être reconnu ; si par contre l'un des mots était dans L_0 et l'autre ailleurs, on devrait pouvoir les départager avant la fin du cycle).

Et si les chemins de L_0 et L_1 sont sans cycle, on peut les reconnaître grâce à un morceau de dépliage injectif sur l'automate initial, ce qui veut dire que les transitions distinguantes du dépliage correspondent à des transitions uniques sur l'automate et que le fait de les traverser suffit à décider si on a observé L_0 ou L_1 (inutile de se rappeler le début du chemin).

Donc, pour distinguer deux ensembles de comportements disjoints, il suffit de se donner les deux ensembles de transitions distinguantes de l'automate $\mathcal{A}_{/R}$ qui permettent de classer w soit dans L_0, soit dans L_1. Ceci revient à se donner, sur l'arène des ensembles de positions C_0 et C_1 qui soient les réunions respectives des positions étiquetées par ces deux ensembles de transitions ($C_0 = \bigcup\{\omega(l'_{R,t_0}) | t_0 \text{ distinguant } L_0\}$ et $C_1 = \bigcup\{\omega(l'_{R,t_1}) | t_1 \text{ distinguant } L_1\}$).

C'est pourquoi le fait, pour S, d'interférer depuis une position se caractérise par l'existence de deux ensembles de positions observables C_0 et C_1 compatibles avec $\simeq_R$ et tels que S soit en mesure, depuis cette position, de guider le système vers C_0 et aussi de le guider vers C_1. A priori, cela peut se faire avec l'aide de R, à condition que ses choix ne dépendent pas de ce que S cherche à envoyer, vu que S n'est pas censé le savoir à l'avance. Cependant, dans un premier temps, nous supposerons que S cherche à interférer sans l'aide de R.

Suite à ces considérations, nous fixons une fois pour toutes la notion de protocole éligible :

Définition 6.1 (Protocole éligible). Dans l'arène étiquetée obtenue par la méthode expliquée en 6.1, un protocole éligible $(C_0, C_1) \in 2^V \times 2^V$ est un couple d'ensembles de positions non vides, disjoints, compatibles avec $\simeq_R$ et observables par R.

6.3 Formalisation

Nous allons maintenant formaliser cette notion grâce à AMC. On suppose que S et R se sont mis d'accord sur un protocole éligible (C_0, C_1). Comme C_0 et C_1 sont des ensembles de positions, nous les assimilerons à des propositions (étiquettes) et nous les ferons apparaître tels quels dans les formules logiques. On posera $C \triangleq C_0 \cup C_1$.

En AMC, le fait de pouvoir immédiatement envoyer un 0 (on est dans L_0) s'écrit $\langle\langle S \rangle\rangle \bigcirc C_0$.

Le fait de pouvoir envoyer un 0, plus tard, sans brouiller le canal (c'est-à-dire envoyer des signaux C_0 ou C_1 autre que ceux voulus par S), se note $\langle\langle S\rangle\rangle\neg C\mathcal{U}(\neg C \wedge \langle\langle S\rangle\rangle \bigcirc C_0)$.

Donc pouvoir envoyer un 0 éventuellement à terme se note $\langle\langle S\rangle\rangle \bigcirc C_0 \vee \langle\langle S\rangle\rangle\neg C\mathcal{U}(\neg C \wedge \langle\langle S\rangle\rangle \bigcirc C_0)$, ce qui est équivalent à $\langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle\neg C\mathcal{U}C_0$.

Pouvoir interférer, c'est donc pouvoir faire cela pour 0 et pour 1 : $(\langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle\neg C\mathcal{U}C_0) \wedge (\langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle\neg C\mathcal{U}C_1)$

On en déduit la définition suivante :

Définition 6.2 (Position d'interférence (à 1 bit)). Étant donné un protocole éligible (C_0, C_1), une position $v \in V$ est une *position d'interférence* pour ce protocole si et seulement si

$$\mathbb{A}, v \models (\langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle\neg C\mathcal{U}C_0) \wedge (\langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle\neg C\mathcal{U}C_1) \quad (1)$$

7 Caractérisation logique des canaux cachés

Il s'agit maintenant d'expliquer la notion de canal caché en s'aidant de celle d'interférence. L'idée sera qu'un canal caché existe à partir d'une position si non seulement c'est une position d'interférence, mais aussi si après avoir interféré, on est encore dans le S-attracteur de cette même position, ou bien d'une autre position à canal caché (c'est-à-dire la zone à partir de laquelle S peut forcer le retour à une de ces positions, éventuellement en jouant plusieurs coups ; une définition formelle de S-attracteur est donnée en annexe, mais n'est pas utile ici).

7.1 Formulation en AMC

Nous rechercherons en un premier temps l'existence d'un canal caché pour un protocole éligible (C_0, C_1) donné. La présence d'un canal caché est le fait pour S et R de pouvoir se ramener à une position d'interférence (pour (C_0, C_1)), puis après avoir interféré d'interférer à nouveau indéfiniment.

D'abord, quelques notations pour simplifier les écritures :

- $\mathcal{S}_0(\varphi) \triangleq \langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle\neg C\mathcal{U}(C_0 \wedge \varphi)$ (symbolisant l'existence d'une stratégie pour que S émette un 0, puis fasse φ)
- $\mathcal{S}_1(\varphi) \triangleq \langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle\neg C\mathcal{U}(C_1 \wedge \varphi)$(symbolisant l'existence d'une stratégie pour que S émette un 1, puis fasse φ)
- $\mathcal{S}_{pos}(\varphi) \triangleq \varphi \vee \langle\langle S, R\rangle\rangle \bigcirc \langle\langle S, R\rangle\rangle\neg C\mathcal{U}(\neg C \wedge \varphi)$ (symbolisant la possibilité pour S et R de se coordonner pour se placer là où φ est satisfaite sans émettre de 0 ou de 1)

Interférer, c'est pouvoir faire en sorte d'émettre un 0 tout en étant aussi capable d'émettre un 1 :

$$(\langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle\neg C\mathcal{U}C_0) \wedge (\langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle\neg C\mathcal{U}C_1)$$

Cette formule se résume en $\mathcal{S}_0(\top) \wedge \mathcal{S}_1(\top)$

Interférer, puis avoir la possibilité de faire "autre chose", qui serait décrit dans une formule φ, s'écrirait :

$$\mathcal{S}_{int}(\varphi) \triangleq (\langle\langle S\rangle\rangle\neg C\mathcal{U}\langle\langle S\rangle\rangle \bigcirc (C_0 \wedge \varphi)) \wedge (\langle\langle S\rangle\rangle\neg C\mathcal{U}\langle\langle S\rangle\rangle \bigcirc (C_1 \wedge \varphi))$$

Soit $\mathcal{S}_{int}(\varphi) = \mathcal{S}_0(\varphi) \wedge \mathcal{S}_1(\varphi)$.

Pouvoir se ramener à une telle position, sans brouiller le canal (émettre des 0 ou des 1, sans compter celui qui vient d'être émis) s'écrirait :

$$\mathcal{S}_{int}(\varphi) \vee \langle\langle S, R\rangle\rangle \bigcirc \langle\langle S, R\rangle\rangle(\neg C_0 \wedge \neg C_1)\mathcal{U}(\neg C \wedge \mathcal{S}_{int}(\varphi))$$

ou encore $\mathcal{S}_{pos}(\mathcal{S}_{int}(\varphi))$ ou $\mathcal{S}_{pos}(\mathcal{S}_0(\varphi) \wedge \mathcal{S}_1(\varphi))$

Enfin, en utilisant l'opérateur ν qui va permettre de décrire φ comme le plus grand ensemble permettant à une telle interférence de pouvoir être répétée indéfiniment, on décrirait la présence d'un canal caché par la formule ci-dessous :

$$CC \triangleq \nu X.\mathcal{S}_{pos}(\mathcal{S}_0(X) \wedge \mathcal{S}_1(X)) \tag{2}$$

7.2 Première formulation algorithmique

On veut vérifier si le protocole éligible donné (C_0, C_1) est source d'interférence dans l'arène considérée.

Pour trouver l'ensemble des positions à partir desquelles un canal caché existe, il suffit de model-checker la formule $CC = \nu X.\mathcal{S}_{pos}(\mathcal{S}_0(X) \wedge \mathcal{S}_1(X))$.

On a vu que le temps de calcul était polynomial. On peut même prouver qu'il sera quadratique en le nombre de positions : en effet, on a pu prouver [9] que la complexité du model-checking du μ-calcul était de m^{d+1} avec m le nombre de positions de l'arène, et d la profondeur maximale d'alternation de la formule (c'est à dire le nombre de changement d'opérateur μ/ν rencontrés sur le chemin syntaxique entre une variable et le quantificateur μ/ν dont elle dépend). Le model-checking d'AMC en diffère seulement par la signification de l'opérateur $\langle\langle\cdot\rangle\rangle \bigcirc \cdot$, qui est plus fine que celle des modalités $\langle\ \rangle\cdot$ et $[\ ]\cdot$ du μ-calcul, mais ne nécessite pas plus d'exploration.

Ici, la profondeur maximale est de 1. Donc le temps de calcul est quadratique.

Maintenant, pour résoudre le problème entier, c'est-à-dire l'existence d'un protocole de canal caché qui fonctionne, il s'agit de vérifier cette propriété CC pour tout couple C_0, C_1 d'ensembles de positions éligible. Il s'agira donc a priori d'explorer toutes les partitions en trois ensembles de $V_{/\simeq_R}$, c'est-à-dire l'ensemble des classes de positions observationnellement équivalentes pour R jusqu'à trouver un protocole de canal caché qui marche. La complexité de cette exploration est exponentielle en le nombre de transitions d'émission dans la projection sur R.

7.3 Seconde formulation algorithmique

En annexe, nous donnons un algorithme (non encore prouvé) qui permettrait de calculer des protocoles éligibles tels que CC soit satisfaite dans certaines positions, évitant ainsi de model-checker CC pour tout protocole éligible. Le temps de calcul devrait ainsi être moindre.

8 Le jeu du canal caché

Le choix du formalisme des arènes n'était pas innocent : il permet d'utiliser la théorie des jeux. Et la théorie des jeux devrait permettre une caractérisation plus intuitive des canaux cachés que celle donnée par une formule d'AMC.

Cependant, l'arène obtenue auparavant ne permet pas de définir facilement le jeu du canal caché où pour gagner, il faut être capable d'envoyer n'importe quel mot... ce qui ferait naïvement un jeu par mot, arbitrairement long, qu'on est susceptible de vouloir envoyer.

L'idée sera donc d'intégrer le message au jeu et de faire jouer S et R à la fois contre l'environnement, et à la fois contre le message.

8.1 Définitions

D'abord rappelons ce qu'est un jeu :

Définition 8.1 (Jeu). Un jeu $\mathcal{G} \triangleq \langle \mathbb{A}, \mathcal{W}in \rangle$ est une arène $\mathbb{A}$ munie d'une condition de gain $\mathcal{W}in \subseteq V^* \cup V^\omega$.

Définition 8.2 (Chemin gagnant). Un chemin $w \in V^*$ de l'arène $\mathbb{A}$ est dit gagnant dans le jeu $\langle \mathbb{A}, \mathcal{W}in \rangle$ si $w \in \mathcal{W}in$.

Définition 8.3 (Conformité). Soit $\langle \Pi, (V_p)_{p\in\Pi}, E \rangle$ une arène. Soit f une fonction partielle des chemins de sommets $(V^* \cdot V_p)$ dans 2^V. Un chemin w est *conforme* à f dans cette arène, si pour tout préfixe π de w, $\pi \in V^* \cdot V_p$ implique qu'il existe $v \in f(\pi)$ tel que $\pi \cdot v$ est un préfixe de w.

Définition 8.4 (Stratégie (locale)). Soit $\langle \Pi, (V_p)_{p\in\Pi}, E, L, (L_p)_{p\in\Pi}, \omega \rangle$ une arène étiquetée, et $p \in \Pi$ un joueur. Une *stratégie* pour le joueur p sur un ensemble de positions U de cette arène est une fonction partielle de $V^* \cdot V_p$ dans $V_{/\simeq_p}$

- définie sur tout préfixe d'un chemin partant de U, conforme à f et qui finit par une position de V_p,
- telle que pour tout chemin w tel que $f(w)$ existe, il existe $v \in f(w)$ tel que $w \cdot v$ est un chemin
- et telle que si f_p est définie sur w_1 et w_2, alors $w_1 \simeq_p w_2 \implies f(w_1) = f(w_2)$.

Définition 8.5 (Stratégie sans mémoire). Une stratégie est dite *sans mémoire* si elle dépend uniquement de la dernière classe de positions du chemin observée par p. On s'autorisera à noter une stratégie sans mémoire comme une fonction de $V_{/\simeq_p} \to V_{/\simeq_p}$.

Définition 8.6 (Stratégie distribuée).

Une stratégie *distribuée* sur un en semble de joueurs P est une famille de stratégies $(f_p)_{p\in P}$ telle que pour tout joueur $q \in P$, f_q est une stratégie pour le joueur q et telle que pour tout chemin conforme à $\bigcup_{p\in P} f_p$ finissant en V_q, f_q est définie.

Un chemin de $V^* \cup V^w$ se conforme à une stratégie distribuée $(f_p)_{p\in P}$ si et seulement si pour tout $p \in P$ il se conforme à f_p.

Définition 8.7 (Stratégie gagnante). Une stratégie (locale ou distribuée) est *gagnante* dans le jeu $\mathcal{G}$ si les chemins qui s'y conforment sont des chemins gagnants dans ce jeu.

Interprétation des stratégies locales : une stratégie de p dit ce qui doit être joué dans les différentes positions de ce joueur. Or le joueur n'observe (et ne distingue) que les positions observables sans savoir exactement à quel moment le système va passer dans une position effectivement contrôlée par p.

Donc si p veut effectuer une certaine action dictée par la stratégie après l'observation d'une position de la classe $[v]_p$, on considérera que celle-ci sera « bloquée » jusqu'à ce que ce soit effectivement son tour de jouer et qu'elle ait effectivement lieu. On considérera aussi que le système ne peut pas arriver sur une autre position observable sans passer par une position de p (ou que p ne peut pas tenter d'effectuer l'action si le système ne peut pas passer par une position contrôlée par p), car cela reviendrait à dire que p a effectué une action non prévue dans le modèle.

Évidemment, le fait que, dans un modèle, une action soit rendue impossible à cause d'un événement qui n'est pas observable n'est pas raisonnable. Ainsi, il conviendrait d'étudier seulement des modèles où ce cas ne se présente pas : à savoir qu'une action invisible pour un processus p ne change pas les actions possibles pour p (modulo ordonnancement).

8.2 Canal caché

On se place à nouveau dans le cas où l'on s'est déjà fixé un protocole éligible (C_0, C_1) entre S et R.

Les chemins gagnants pour le coupe (S, R) sont ceux où ils peuvent se retrouver indéfiniment en position d'émettre n'importe quel bit d'information : quels que soient les choix de l'environnement (M), quel que soit le bit à envoyer à l'instant, il est toujours possible d'envoyer ce bit en un temps fini. En résumé, il s'agit d'un jeu de S et R contre M et le texte du message à envoyer que l'on symbolisera par un nouveau joueur ennemi η_T.

Donc on complète l'arène étiquetée en ajoutant l'information de la dernière exigence du texte. On construit l'arène $\mathbb{A}' \triangleq \langle \Pi', (V'_p)_{p\in\Pi'}, E', L', (L'_p)_{p\in\Pi'}, \omega' \rangle$ de la façon suivante :

- $\Pi' \triangleq \Pi \cup \{\eta_T\}$
- $p \in \Pi, V'_p \triangleq V_p \times \{0,1\}$ et $V'_{\eta_T} = (C_0 \cup C_1 \cup \{\bot\}) \times \{+,-\}$
-

$$\begin{aligned} E' \triangleq & \{((v,0),(v',0)) | (v,v') \in E \text{ et } v' \notin C_0 \cup C_1\} \\ & \cup \{((v,1),(v',1)) | (v,v') \in E \text{ et } v' \notin C_0 \cup C_1\} \\ & \cup \{((v,0),(v',+)) | v \in V_S \text{ et } (v,v') \in E \text{ et } v' \in C_0\} \\ & \cup \{((v,1),(v',+)) | v \in V_S \text{ et } (v,v') \in E \text{ et } v' \in C_1\} \\ & \cup \{((v,0),(v',-)) | v \in V_S \text{ et } (v,v') \in E \text{ et } v' \in C_1\} \\ & \cup \{((v,1),(v',-)) | v \in V_S \text{ et } (v,v') \in E \text{ et } v' \in C_0\} \\ & \cup \{((v,+),(v,0)) | v \in C_1 \cup C_0\} \\ & \cup \{((v,+),(v,1)) | v \in C_1 \cup C_0\} \end{aligned}$$

  Explication :
  - Tous les coups jouables de la composante $V \times \{i\}$ qui envoient le bit i ramènent en un point contrôlé par le joueur représentant le texte $(v,+)$.
  - Les coups qui envoient le bit $1-i$ envoient dans des puits $(v,-)$.
  - Les points contrôlés par le message renvoient à leur tour au choix dans la composante $V \times \{i\}$ ou $V \times \{1-i\}$ sur la position correspondant au bit qui vient d'être émis.
  - Les autres transitions (non émettrices) à l'intérieur d'une composante se comportent comme dans l'arène $\mathbb{A}$.
- $L' = \{l_0, l_1\} \cup L$
- $\forall p \in \Pi, p \neq S, L'_p = L_p$
- $L'_S = L_S \cup \{l_0, l_1\}$
- $\forall l \in L'_p, \omega'(l) = (\omega(l) \times \{0,1,+,-\}) \cap V'$
  Les anciennes étiquettes marquent les positions de la nouvelle arène qui correspondent à celles qui ont la même étiquette dans l'ancienne arène, sans permettre de distinguer les composantes 0 et 1.
- $\omega'(l_0) = \{v \in V | \exists l \in L_S, v \in \omega(l)\} \times \{0\}$
- $\omega'(l_1) = \{v \in V | \exists l \in L_S, v \in \omega(l)\} \times \{1\}$
  S et lui seul sait ce qu'il doit envoyer (parmi les positions qu'il peut observer, il distingue les deux composantes 0 et 1).
- $L_{\eta_T} \triangleq \emptyset$
  Nous n'avons pas besoin de considérer les étiquettes du « joueur texte ».

On adoptera aussi les notations suivantes :

- $V^T \triangleq (C_0 \cup C_1 \cup \{\bot\}) \times \{+\}$: ce sont les positions contrôlées par le texte qui ne sont

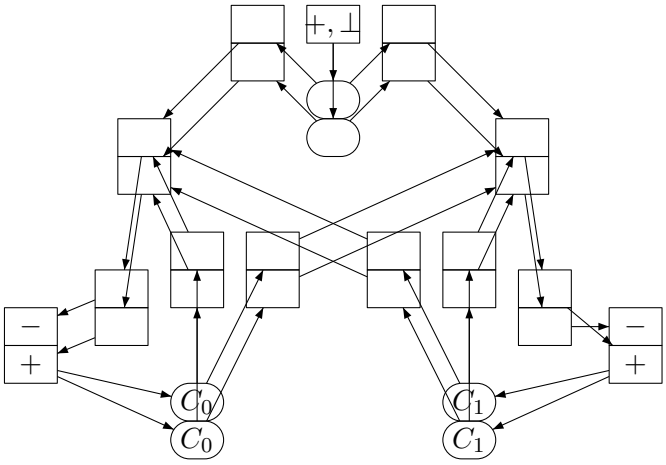

FIG. 10 – Arène modifiée $\mathbb{A}'$ obtenue à partir de l'arène $\mathbb{A}$ de la figure 8

pas des puits, c'est-à-dire soit la position initiale $(\perp, +)$, soit les positions où la dernière exigence du texte a été satisfaite (C_0 et C_1).
- $V^0 \triangleq V \times \{0\}$: se sont toutes les positions contrôlées par des joueurs normaux, et où S se rappelle que l'exigence courante à satisfaire est d'envoyer un 0.
- $V^1 \triangleq V \times \{1\}$: se sont toutes les positions contrôlées par des joueurs normaux, et où S se rappelle que l'exigence courante à satisfaire est d'envoyer un 1.

La figure 10 montre ce que donne l'arène de la figure 8, une fois transformée par ce procédé. Ici nous avons représenté les positions « dédoublées » par des positions superposées en mettant toujours en dessous la position où l'objectif de S est d'envoyer un 0, et au dessus celle où son objectif est d'envoyer un 1.

Dans cette arène, on peut considérer que le texte fixe un nouveau challenge à chaque émission d'un bit caché. Remarquons que, par construction, un chemin qui fait émettre le bit inverse au challenge fixé par le texte à envoyer se retrouve automatiquement dans un puits et ne peut pas être infini. De plus, après être passé par V^T, on sait qu'au coup suivant on arrive dans une position où on émet le bon bit. Or, on veut que les runs gagnants soient ceux qui font émettre des bons bits infiniment souvent (et jamais de mauvais), ce qui se ramène aux runs infinis qui passent infiniment souvent dans V^T. Ce sont donc les runs de

$$\mathcal{W}in \triangleq ((V^{0*} + V^{1*}) \cdot V^T)^\omega. \tag{3}$$

Nous avons ainsi achevé la définition du jeu du canal caché :

$$\mathcal{G} = \langle \mathbb{A}', \mathcal{W}in \rangle. \tag{4}$$

8.3 Cohérence avec la partie logique

Nous allons maintenant établir un lien entre cette nouvelle caractérisation sous forme de jeu du canal caché et la caractérisation logique de la section précédente.

Pour cela, il serait intéressant de disposer dans la nouvelle arène d'une formule d'AMC qui décrive au plus près l'existence de stratégies gagnantes pour le jeu $\mathcal{G}$. La formule suivante, inspirée de CC, semble raisonnablement rendre compte du fait que S et R disposent d'une

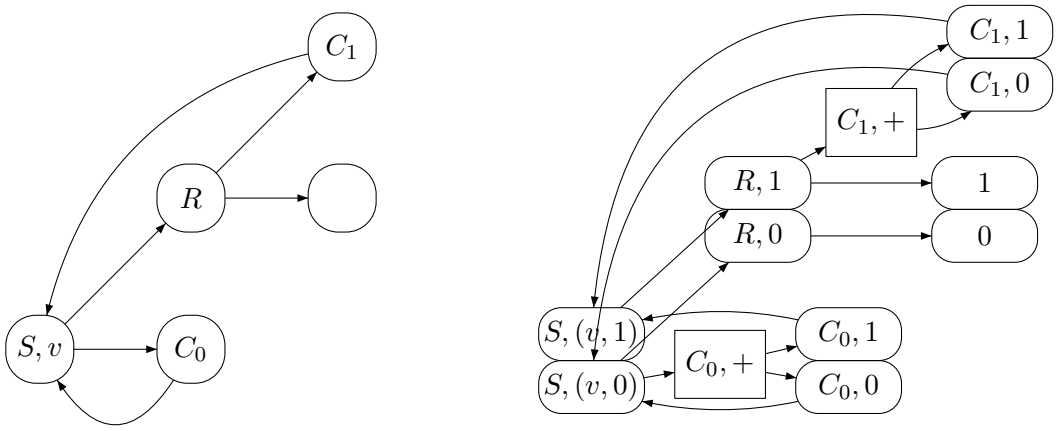

FIG. 11 – À gauche : cas d'une arène $\mathbb{A}$ telle que $\mathbb{A}, v \not\models CC$ mais $\mathbb{A}', (v_0, 0) \models CC'$ (Les positions contrôlées par S, resp. R, sont signalées par la lettre S, resp. R. Les positions de l'ensemble C_0, resp. C_1, sont marquées C_0, resp. C_1.). À droite : $\mathbb{A}'$.

stratégie pour rester dans $\mathcal{W}in$:

$$CC' \triangleq \nu\varphi.\langle\langle S, R\rangle\rangle \neg V^T \mathcal{U} (V^T \wedge \forall \bigcirc \varphi) \tag{5}$$

Nous nous proposons ainsi de démontrer que si CC se vérifie en une position v de $\mathbb{A}$, alors CC' se vérifie dans la position v_0 (et v_1) de $\mathbb{A}'$ et que si cela est vrai, alors il existe une stratégie gagnante pour S, R dans $\mathcal{G}$ en v_0 (et v_1).

Théorème 8.8. *$\mathbb{A}, v \models CC$ implique $\mathbb{A}, (v, 0) \models CC'$ et $\mathbb{A}, (v, 1) \models CC'$.*

Théorème 8.9. *Si $\mathbb{A}, v \models CC'$, alors il existe une stratégie distribuée gagnante et sans mémoire pour S, R en v.*

Les preuves de ces deux théorèmes sont données en annexe. Les réciproques sont fausses, nous le montrerons par contre-exemples : figures 11 et 12.

La figure 12 est le contre exemple du théorème 8.9. Nous y avons représenté seulement une partie de l'arène $\mathbb{A}'$.

Les positions marquées R (resp. S), sont les positions contrôlées par R (resp. S). Les positions marquées z_0 (resp. z_1 et z_2) sont dans une même classe d'équivalence de $\simeq_S$ (z_0, z_1 et z_2 sont trois classes différentes). R, en revanche distingue les deux positions de z_0.

Une fois que S sait que l'on est dans z_0, il ne peut pas savoir s'il faut jouer z_1 ou z_2, donc la classe z_0 ne peut pas être rangée dans $\langle\langle S, R\rangle\rangle \bigcirc \langle\langle S, R\rangle\rangle \bigcirc \varphi$. Pourtant si S et R se mettent d'accord au préalable pour que R joue toujours vers la même position de z_0 (par exemple celle du haut), S saurait ce qu'il devrait jouer (dans l'exemple : z_2) pour que le chemin suivi soit dans $(\neg V^T)^* \cdot \varphi^{\mathbb{A}'}$. Ainsi S et R ont une stratégie depuis v pour que soit joué le préfixe $(\neg V^T)^* \cdot \varphi^{\mathbb{A}'}$.

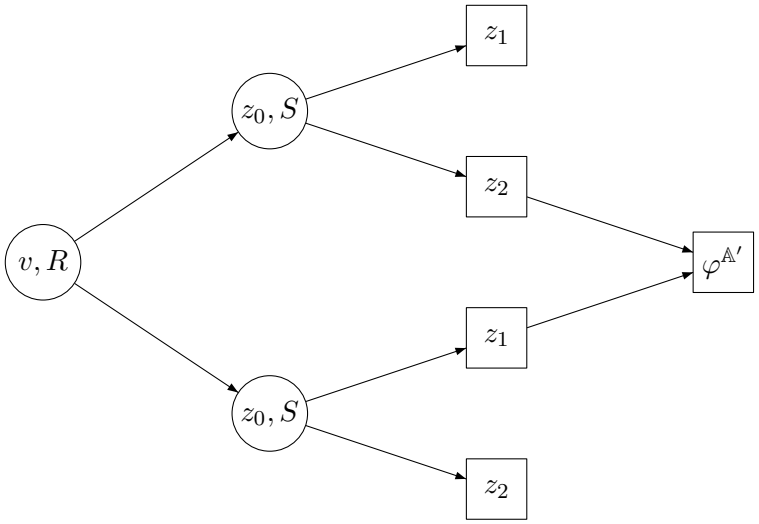

FIG. 12 – Cas d'une arène où il existe une stratégie depuis v pour rejoindre $\varphi^{\mathbb{A}'}$ sans passer par V^T alors que $\mathbb{A}', v \not\models \langle\langle S, R\rangle\rangle \neg V^T \mathcal{U} \varphi$.

Conclusion et perspectives

Pour conclure, nous avons accompli les objectifs principaux qui étaient de caractériser le jeu du canal caché et de montrer le lien avec une caractérisation sous forme de logique temporelle.

Cependant, il apparaît ici que la logique choisie est insuffisante pour caractériser la non existence d'une stratégie distribuée gagnante. On aimerait bien affiner la sémantique de cette logique afin que l'on puisse prendre en compte toutes les stratégies gagnantes, mais alors se poserait le problème du model-checking qui serait a priori indécidable en général dans le cadre d'une observation partielle du système par les agents.

Par ailleurs, on s'est aussi rendu compte que la première caractérisation logique du canal caché était trop restrictive, puisqu'on a trouvé une autre caractérisation satisfaisante qui en est sa conséquence stricte. On pourrait se contenter de n'utiliser alors que la seconde caractérisation, mais la première présentait l'intérêt d'être une formule logique à vérifier sur une arène qui ne dépend pas du choix du protocole (C_0, C_1) (seule la formule en dépend) et donc pouvait servir de fondement à un algorithme qui essayerait de vérifier cette formule tout en construisant le protocole.

Ce dernier algorithme aurait l'avantage de pouvoir trouver le protocole qui va faire en sorte que la formule soit satisfaite dans le plus de positions possibles sans avoir à parcourir tous les protocoles éligibles imaginables (quasiment toutes les partitions en trois ensembles de $V_{/\simeq_R}$), ou bien, moyennant une légère modification, de trouver un protocole qui fonctionne sur un ensemble de positions non vide. Une proposition, non démontrée, pour un tel algorithme est donnée en annexe.

Et en effet, pour une application industrielle réaliste de ces méthodes, il serait appréciable, pour le temps de calcul, d'avoir un moyen efficace pour éviter d'avoir à tester tous les protocoles admissibles.

Enfin, même si la vérification restait relativement coûteuse, la logique à temps alternant et information incomplète, ainsi que la théorie des jeux, apparaissent comme des outils particulièrement adaptés pour décrire la propriété de canal caché, à condition bien sûr d'accepter que nous n'obtenons qu'une sous-approximation de la réalité. En fait, ne pas détecter certains canaux cachés n'est pas forcément un problème puisque, de la spécification d'un système à son implantation, de nouveaux canaux peuvent être introduits. Maintenant, je pense qu'il serait intéressant de décrire d'autres modèles de faille de sécurité faisant intervenir une adversité entre les agents malveillants et les autres par des formules de logique alternante, afin de tenter systématiser cette approche.

Ce qui pourrait aussi être prometteur, serait de décrire le système par un automate temporisé [1] que l'on transformerait en jeu temporisé [5]. On appliquerait ainsi des méthodes similaires à celles de ce stage pour calculer la présence de canaux cachés faisant éventuellement intervenir le temps comme vecteur d'information, en espérant ne pas trop augmenter la complexité de la vérification. Après cela, on pourrait de plus calculer le temps nécessaire à l'émission d'un bit et estimer ainsi le débit du canal.

Enfin, une autre voie d'approfondissement serait de ne plus considérer que l'environnement fait toujours le choix le pire pour les pirates, ce qui n'est évidemment pas le cas systématique dans la nature. Ainsi, on pourrait plutôt associer à chaque position contrôlée par l'environnement une distribution de probabilité sur les différents coups jouables, et tenter de caractériser une propriété d'interférence à $\log_2 x$ bits où x n'est pas forcément un entier, puis caractériser la propriété de canal caché associée.

A Algorithme qui construit le protocole

Plutôt que de relancer l'algorithme de model-checking pour chaque protocole possible, nous allons ici proposer un algorithme qui construit le protocole (C_0, C_1) à mesure qu'il calcule l'ensemble des sommets où il va fonctionner. Ceci est possible en séparant l'arène en morceaux qui peuvent être vérifiés en isolation du reste, à savoir en examinant composante fortement connexe par composante fortement connexe comme dans [13]. L'idée est qu'une fois qu'on ne peut plus agrandir la zone gagnante, on repart en arrière dans les choix faits dans l'algorithme et on réessaye avec d'autres choix (backtracking).

A.1 Définitions

Définition A.1 (P-prédécesseur). d'un ensemble de positions W_X d'une arène par un ensemble de processus P.

C'est l'ensemble de positions $Pre_P(W_X) \triangleq (\langle\langle P\rangle\rangle \bigcirc X)^{\mathbb{A}}([X := W_X])$.

Définition A.2 (P-attracteur). d'un ensemble de positions W_X d'une arène par un ensemble processus P en évitant un ensemble de positions W_Y.

C'est l'ensemble de positions $Att_P(W_X|W_Y) \triangleq (\langle\langle P\rangle\rangle \neg Y \mathcal{U} (X \wedge \neg Y))^{\mathbb{A}}([X := W_X, Y := W_Y])$.

A.2 Algorithme

Éligibles := Fonction(pile, D, C_0, C_1, W_0, W_1, W)

On cherche toutes les paires $(\tilde{C}_0, \tilde{C}_1)$ avec $\tilde{C}_0$ et $\tilde{C}_1$:

- parties de D,
- disjointes,
- non toutes deux vides,
- observables par R (positions étiquetées par L_R)
- compatibles avec $\simeq_R$,
- tels que R distingue toute position de $\tilde{C}_0$, $\tilde{C}_1$ de toute position hors-pile,
- et tels qu'en posant
  $J := Pre_S(Att_S(\tilde{C}_0 \cup W_0'|C_0 \cup C_1)) \cap Pre_S(Att_S(\tilde{C}_1 \cup W_1'|C_0 \cup C_1))$
  on ait $\tilde{C}_0 \cup \tilde{C}_1 \subseteq J \cup W \cup Att_{S,R}(J \cup W|C_0 \cup C_1)$.

  On retourne l'ensemble $\{(C_0 \cup \tilde{C}_0, C_1 \cup \tilde{C}_1, J)|(\tilde{C}_0, \text{ et } \tilde{C}_1) \text{ sélectionnés}\}$.

FinFonction

Stabiliser := Fonction$(C_0, C_1, pile, W_0, W_1, W)$

- $W := W \cup J \cup Att_{S,R}(W \cup J|C_0 \cup C_1)$;
- Jusqu'à obtention d'un point fixe, faire :
  - $W_0' := Att_S(C_0 \cap W|C_0 \cup C_1)$;
  - $W_0 := Pre_S(W_0')$;
  - $W_1' := Att_S(C_1 \cap W|C_0 \cup C_1)$;
  - $W_1 := Pre_S(W_1')$;
  - $W := W_0 \cap W_1\, Att_{S,R}(W_0 \cap W_1|C_0 \cup C_1)$

  FinJusqu'à ;
- empiler les CFC de $D \setminus Att_{S,R}(J \cup W|C_0 \cup C_1)$ en ordre d'accessibilité ;
- Retourner (pile, W_0, W_1, W)

FinFonction

Essayer := Fonction(C_0, C_1,pile, W_0, W_1, W)
- TantQue pile des CFC non vide
  - On dépile D
  - Si $D \cap W \neq \emptyset$:
    - $D := D \backslash W$
    - empiler les CFC de D résultantes
    - on passe à la suite

    Sinon :
    - E := Eligibles (pile, D, C_0, C_1, W_0, W_1, W) ;
    - Pour tous les (C_0, C_1, J) de E, essayer (C_0, C_1,Stabiliser(C_0, C_1,pile, W_0, W_1, W))

    FinSi

  FinTantQue
- Imprime (C_0, C_1, W)

FinFonction

// initialisation :

On décompose l'ensemble V des positions de l'arène en CFC (composantes fortement connexes, algorithme de Tarjan).

On empile les CFC de V en ordre d'accessibilité -> pile

$W := \emptyset$; // zone gagnante (canal caché) pour S et R

$W_0 := \emptyset$; // zone gagnante (envoi d'un 0, puis canal caché) pour S

$W_1 := \emptyset$; // zone gagnante (envoi d'un 1, puis canal caché) pour S

$C_0 := \emptyset$;

$C_1 := \emptyset$;

Essai ($\emptyset, \emptyset$, pile, $\emptyset, \emptyset, \emptyset$) ; // lancement du programme

B Preuve des implications entre caractérisations

B.1 Preuve du théorème 8.8

Démonstration.

Dans la suite, à chaque ligne, nous affirmons que pour toute formule logique φ, et pour toute formule φ' tel que $\mathbb{A}, v \models \varphi$ implique $\mathbb{A}', (v, 0) \models \varphi'$ et $\mathbb{A}', (v, 1) \models \varphi'$, les assertions qui suivent sont vérifiées :

Supposons v tel que $\mathbb{A}, v \models \langle\langle S \rangle\rangle \bigcirc \varphi)$. Alors, par construction,

$$\mathbb{A}', (v, 0) \models \langle\langle S \rangle\rangle \bigcirc ((V^T \wedge \forall \bigcirc (C_0 \wedge \varphi')) \vee (V^0 \wedge \neg C \wedge \varphi')).$$

Il en découle que si $\mathbb{A}, v \models (\neg C \wedge \langle\langle S \rangle\rangle \bigcirc Y) \vee (\neg C \wedge \langle\langle S \rangle\rangle (C_0 \wedge \varphi))$, avec $Y \implies \neg C$, alors

$$\mathbb{A}', (v, 0) \models (\neg C \wedge \langle\langle S \rangle\rangle \bigcirc (V^0 \wedge m(Y))) \vee (\neg C \wedge \langle\langle S \rangle\rangle \bigcirc (V^T \wedge \forall \bigcirc \varphi')).$$

Donc si $\mathbb{A}, v \models \langle\langle S\rangle\rangle \neg C\mathcal{U}(\neg C \wedge (\langle\langle S\rangle\rangle \bigcirc (C_0 \wedge \varphi)))$, alors

$$\mathbb{A}, v \models \mu Y.((\neg C \wedge \langle\langle S\rangle\rangle \bigcirc Y) \vee (\neg C \wedge \langle\langle S\rangle\rangle (C_0 \wedge \varphi))),$$

et donc

$$\begin{aligned}\mathbb{A}', (v,0) \models \mu Y.((&\neg C \wedge \langle\langle S\rangle\rangle \bigcirc (V^0 \wedge Y)) \\ &\vee (\neg C \wedge \langle\langle S\rangle\rangle \bigcirc (V^T \wedge \forall \bigcirc \varphi'))).\end{aligned}$$

Comme $(v,0) \in V^0$, ce dernier résultat est équivalent à

$$\mathbb{A}', (v,0) \models \mu Y.((\neg C \wedge V^0 \wedge \langle\langle S\rangle\rangle \bigcirc Y) \vee (\neg C \wedge V^0 \wedge \langle\langle S\rangle\rangle \bigcirc (V^T \wedge \forall \bigcirc \varphi')))$$

que l'on réécrit

$$\mathbb{A}', (v,0) \models \langle\langle S\rangle\rangle (\neg C \wedge V^0)\mathcal{U}(\neg C \wedge V^0 \langle\langle S\rangle\rangle \bigcirc (V^T \wedge \forall \bigcirc \varphi')).$$

Ceci implique $\mathbb{A}', (v,0) \models \langle\langle S, R\rangle\rangle \neg V^T \mathcal{U}(\neg V^T \wedge \langle\langle S, R\rangle\rangle \bigcirc (V^T \wedge \forall \bigcirc \varphi'))$, qui est lui-même équivalent à $\mathbb{A}', (v,0) \models \neg V^T \wedge \langle\langle S, R\rangle\rangle \bigcirc \langle\langle S, R\rangle\rangle \neg V^T \mathcal{U}(V^T \wedge \forall \bigcirc \varphi')$, que l'on écrira $\mathbb{A}, (v,0) \models \mathcal{S}'(\varphi')$.

Donc, pour résumer, $\mathbb{A}, v \models \langle\langle S\rangle\rangle \neg C\mathcal{U}(\neg C \wedge (\langle\langle S\rangle\rangle \bigcirc (C_0 \wedge \varphi)))$ implique $\mathbb{A}, (v,0) \models \mathcal{S}'(\varphi')$. Or $\langle\langle S\rangle\rangle \neg C\mathcal{U}(\neg C \wedge (\langle\langle S\rangle\rangle \bigcirc (C_0 \wedge \varphi))) \equiv_{\mathbb{A}} (\neg C \wedge \langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle \neg C\mathcal{U}(C_0 \wedge \varphi)) \equiv_{\mathbb{A}} (\neg C \wedge \mathcal{S}_0(\varphi))$.

Donc $\mathbb{A}, v \models \neg C \wedge \mathcal{S}_{int}(\varphi)$ implique $\mathbb{A}, v \models \neg C \wedge \mathcal{S}_0(\varphi)$ qui implique $\mathbb{A}, (v,0) \models \mathcal{S}'(\varphi')$.

$$\begin{aligned}\mathcal{S}_0(\varphi) &\equiv_{\mathbb{A}} (\neg C \wedge \mathcal{S}_0(\varphi)) \vee (C \wedge \mathcal{S}_0(\varphi)) \\ &\equiv_{\mathbb{A}} (\neg C \wedge \mathcal{S}_0(\varphi)) \vee C \wedge \langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle \neg C\mathcal{U}(C_0 \wedge \varphi) \\ &\equiv_{\mathbb{A}} (\neg C \wedge \mathcal{S}_0(\varphi)) \\ &\quad \vee C \wedge \langle\langle S\rangle\rangle \bigcirc ((C_0 \wedge C) \vee (\neg C \wedge \langle\langle S\rangle\rangle \bigcirc \langle\langle S\rangle\rangle \neg C\mathcal{U}(C_0 \wedge \varphi))) \\ &\equiv_{\mathbb{A}} (\neg C \wedge \mathcal{S}_0(\varphi)) \vee C \wedge \langle\langle S\rangle\rangle \bigcirc ((C_0 \wedge C) \vee (\neg C \wedge \mathcal{S}_0(\varphi))).\end{aligned}$$

Donc $\mathbb{A}, v \models \mathcal{S}_0(\varphi)$ implique que

$$\mathbb{A}', (v,0) \models \mathcal{S}'(\varphi') \vee C \wedge \langle\langle S\rangle\rangle \bigcirc ((V^T \wedge \forall \bigcirc \varphi) \vee \mathcal{S}'(\varphi')).$$

Or

$$\begin{aligned}&C \wedge \langle\langle S\rangle\rangle \bigcirc ((V^T \wedge \forall \bigcirc \varphi') \vee \mathcal{S}'(\varphi')) \\ \equiv_{\mathbb{A}'} &C \wedge \langle\langle S\rangle\rangle \bigcirc ((V^T \wedge \forall \bigcirc \varphi') \\ &\quad \vee (\neg V^T \wedge \langle\langle S, R\rangle\rangle \bigcirc \langle\langle S, R\rangle\rangle \neg V^T \mathcal{U}(V^T \wedge \forall \bigcirc \varphi'))) \\ \equiv_{\mathbb{A}'} &C \wedge \langle\langle S\rangle\rangle \bigcirc (\langle\langle S, R\rangle\rangle \neg V^T \mathcal{U}(V^T \wedge \forall \bigcirc \varphi')).\end{aligned}$$

Donc finalement $\mathbb{A}', (v,0) \models \mathcal{S}'(\varphi') \vee C \wedge \langle\langle S \rangle\rangle \bigcirc ((V^T \wedge \forall \bigcirc \varphi) \vee \mathcal{S}'(\varphi'))$ a pour conséquence $\mathbb{A}', (v,0) \models \langle\langle S, R \rangle\rangle \bigcirc (\langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (V^T \wedge \forall \bigcirc \varphi'))$, que l'on réécrira $\mathbb{A}', (v,0) \models \mathcal{S}''(\varphi')$. C'est-à-dire $\mathbb{A}, v \models \mathcal{S}_0(\varphi)$ implique que $\mathbb{A}', (v,0) \models \mathcal{S}''(\varphi')$. Et donc $\mathbb{A}, v \models \mathcal{S}_{int}(\varphi)$ implique que $\mathbb{A}', (v,0) \models \mathcal{S}''(\varphi')$

De même on montre que $\mathbb{A}, v \models \mathcal{S}_{pos}(\varphi)$ implique que $\mathbb{A}', (v,0) \models \varphi' \vee \langle\langle S, R \rangle\rangle \bigcirc \langle\langle S, R \rangle\rangle (\neg C \wedge V^0) \mathcal{U} (\neg C \wedge V^0 \wedge \varphi')$, qui implique que $\mathbb{A}', (v,0) \models \varphi' \vee \langle\langle S, R \rangle\rangle \bigcirc \langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (\neg C \wedge V^0 \wedge \varphi')$ ($\mathcal{S}'_{pos}(\varphi')$).

$\langle\langle S, R \rangle\rangle \bigcirc \langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (\neg C \wedge V^0 \wedge \mathcal{S}''(\varphi'))$ implique $\langle\langle S, R \rangle\rangle \bigcirc \langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (\neg V^T \wedge \mathcal{S}''(\varphi')) \equiv_{\mathbb{A}'}$ $\langle\langle S, R \rangle\rangle \bigcirc \langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (\mathcal{S}'(\varphi')) \equiv_{\mathbb{A}'} \langle\langle S, R \rangle\rangle \bigcirc \langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (\neg V^T \wedge \langle\langle S, R \rangle\rangle \bigcirc \langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (V^T \wedge \forall \bigcirc \varphi'))$ qui implique tout simplement $\langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (V^T \wedge \forall \bigcirc \varphi')$.

Il se trouve que $\mathbb{A}', (v,0) \models \mathcal{S}''(\varphi')$ implique aussi que $\mathbb{A}', (v,0) \models \langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (V^T \wedge \forall \bigcirc \varphi')$

On en déduit que $\mathbb{A}, v \models \mathcal{S}_{pos}(\mathcal{S}_{int}(\varphi))$ implique que $\mathbb{A}', (v,0) \models \mathcal{S}''(\varphi') \vee \langle\langle S, R \rangle\rangle \bigcirc \langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (\mathcal{S}'(\varphi'))$, qui implique tout simplement $\mathbb{A}', (v,0) \models \langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (V^T \wedge \forall \bigcirc \varphi')$.

Donc enfin $\mathbb{A}, v \models \mu X. \mathcal{S}_{pos}(\mathcal{S}_{int}(X))$ implique que $\mathbb{A}', (v,0) \models \mu X \langle\langle S, R \rangle\rangle \neg V^T \mathcal{U} (V^T \wedge \forall \bigcirc \varphi')$. En réécrivant, cela donne $\mathbb{A}, v \models CC$ implique $\mathbb{A}', (v,0) \models CC'$.

On fait de même pour montrer que $\mathbb{A}, v \models CC$ implique $\mathbb{A}', (v,0) \models CC'$.

□

B.2 Preuve du théorème 8.9

Démonstration. Nous allons montrer qu'en model-checkant la formule CC', il est facile de construire simultanément une stratégie sans mémoire distribuée gagnante pour S et R.

CC', en effet, peut se développer en

$$\nu X. \mu Y. (V^T \wedge (\forall \bigcirc X)) \vee (\langle\langle S, R \rangle\rangle \bigcirc (\neg V^T \wedge Y)))$$

Formule dont l'algorithme de model-checking s'écrit :

1. $W_X := V$
2. jusqu'à stabilisation de X, faire :
   (a) $W_Y := \emptyset$
   (b) jusqu'à stabilisation de Y, faire :
      i. $W_Y := ((V^T \wedge (\forall \bigcirc X)) \vee (\langle\langle S, R \rangle\rangle \bigcirc (\neg V^T \wedge Y)))^{\mathbb{A}'}[X := W_X, Y := W_Y]$
   (c) $W_X := W_Y$

On remarquera que dans l'avant dernière ligne, on doit d'une part calculer $(V^T \wedge (\forall \bigcirc X))^{\mathbb{A}'}[X := W_X]$, ce qui ne fait pas intervenir de choix de la part de S ou R : en effet, les positions de V^T sont contrôlées par T.

D'autre part, on calcule $(\langle\langle S, R \rangle\rangle \bigcirc (\neg V^T \wedge Y))^{\mathbb{A}'}[Y := W_Y]$, à savoir un certain ensemble de positions à partir desquelles S et R peuvent faire en sorte qu'au prochain coup on soit dans

$\neg V^T \cap Y$. Ceci implique, pour les positions v de cet ensemble contrôlées par S ou R, qu'il existe au moins un coup jouable qui amène dans $[v']_p \subseteq (\neg V^T \cap Y)^{\mathbb{A}'}[Y := W_Y]$ (avec $p = S$ ou R).

Ainsi, on pourrait ajouter une instruction dans l'algorithme qui au moment où l'on calcule $(\langle\langle S, R\rangle\rangle \bigcirc (\neg V^T \wedge Y))^{\mathbb{A}'}[Y := W_Y]$, enregistre, pour chaque classe de positions $[v]_p$ telle que $v \in (\langle\langle S, R\rangle\rangle \bigcirc (\neg V^T \wedge Y))^{\mathbb{A}'}[Y := W_Y] \backslash (\neg V^T \cap W_Y)$ et v contrôlée par $p \in \{S, R\}$, un couple $([v]_p, [v']_p)$ avec $[v']_p \subseteq \neg V^T \cap Y$ et $(v, v') \in E'$, qui correspondrait au coup à jouer pour S ou R.

1. $W_X := V$
2. jusqu'à stabilisation de X, faire :
   (a) $W_Y := \emptyset$
   (b) $s := \emptyset$
   (c) jusqu'à stabilisation de Y, faire :
      i. $W_Y := ((V^T \wedge (\forall \bigcirc X)) \vee (\langle\langle S, R\rangle\rangle \bigcirc (\neg V^T \wedge Y)))^{\mathbb{A}'}[X := W_X, Y := W_Y]$
      ii. $s := s \cup s'$ (où s' est la liste des couples $([v]_p, [v']_p)$ décrits précédemment)
   (d) $W_X := W_Y$

On peut séparer l'ensemble s en deux ensembles f_S et f_R en mettant dans le premier les couples de la forme $([v]_S, [v']_S)$, et dans le second les couples de la forme $([v]_R, [v']_R)$.

Par construction de s, f_S et f_R sont des fonctions (partielles, respectivement sur $V_{/\simeq_S}$ et $V_{/\simeq_R}$ vers $V_{/\simeq_S}$ et $V_{/\simeq_R}$). Par construction, toujours, leurs ensembles image sont constitués soit de positions v contrôlées par S ou R telles que $f_p([v]_p)$ est définie, soit de positions contrôlées par d'autres agents tels que tous les chemins qui en partent arrivent éventuellement dans une position où f_S ou f_R est définie ou dans W_X, et ce, sans être passé par une position de S ou de R. Donc $f \triangleq (f_S, f_R)$ définit bien une stratégie sans mémoire distribuée sur S et R.

Toujours par construction, on sait qu'à tout moment de l'algorithme, quand on suit un chemin qui commence dans Y et qui se conforme à cette stratégie, on se trouvera dans V^T en un nombre fini de coups et qu'au coup suivant on sera forcément dans X. Comme on n'est pas passé par V^T, on en déduit que tout chemin maximal qui se conforme à la stratégie aura un préfixe de $(V^{0*} + V^{1*}) \cdot V^T \cdot X$.

À la fin de l'algorithme, on sait que X a atteint un point fixe et que l'avant dernière valeur de X vaut donc la dernière valeur de Y (qui vaut la dernière valeur de X). Donc, de toute position de l'ensemble calculé à la fin, tout chemin maximal qui se conforme à la stratégie admet un préfixe de $(V^{0*} + V^{1*}) \cdot V^T \cdot X$. Or, après avoir lu un tel préfixe, on est dans un sommet de X qui est égal à sa valeur finale. Ainsi, pour tout entier n positif, tout chemin maximal qui part de $CC'^{\mathbb{A}'}$ admet un préfixe de $((V^{0*} + V^{1*}) \cdot V^T)^n$. Donc tous les chemins maximaux qui se conforment à la stratégie sont dans $((V^{0*} + V^{1*}) \cdot V^T)^\omega = \mathcal{W}in$ □

Index

Références

[1] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2) :183–235, 1994.

[2] R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic, 1997.

[3] A. Arnold. *Finite transition systems*. Prentice-Hall, 1994.

[4] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303 :7–34, 2003.

[5] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In *Hybrid Systems*, pages 1–20, 1994.

[6] D. E. Bell and L. J. LaPadula. Secure computer system : A mathematical model. Technical Report MTR-2547, Vol. 2, The MITRE Corporation, 1973.

[7] D. E. Bell and L. J. LaPadula. Secure computer system : Mathematical foundations. Technical Report MTR-2547, Vol. 1, The MITRE Corporation, 1973.

[8] G. Boudol and I. Castellani. Non interference for concurrent programs. *Lecture Notes in Computer Science*, 2076 :382+, 2001.

[9] E.A. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Proc. 1st Symposium on Logic in Computer Science*, pages 267–278, 1986.

[10] J. A. Goguen and J. Meseguer. Security policies and security models. In *1982 Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, 1982.

[11] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games : A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[12] L. Hélouët. Finding covert channels in protocols with message sequence charts : the case of rmtp2. In *Proc. of SAM'2004, 4th conference on SDL and MSC*, June 2004.

[13] L. Hélouët, M. Zeitoun, and A. Degorre. Scenarios and covert channels, another game... In *Proc. Games in Design and Verification, GDV '04, Satellite of Computer Aided Verification, CAV'04*, Electronic Notes in Theoretical Computer Science. Elsevier, 2004. To appear.

[14] G. Lowe. Quantifying information flow. In *IEEE Computer Security Foundations Workshop*, pages 18–31, June 2002.

[15] A. Sabelfeld and A. Myers. Language-based information-flow security. In *IEEE Journal on Selected Areas in Communications, 21(1)*, 2003.

[16] Volpano and Smith. Eliminating covert flows with minimum typings. In *PCSFW : Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.