Sémantique et raisonnement automatique pour une infrastructure à clés publiques

A Semantics and a Calculi for Reasoning about the Simple Public Key Infrastructure

Nathalie Chetcuti-Sperandio[1] Fabio Massacci[2]

[1] CRIL CNRS
[2] DIT - Università di Trento - Italie

rue Jean Souvraz - SP 18 - 62307 Lens CEDEX
chetcuti@cril.univ-artois.fr

Résumé

Dans un environnement distribué tel qu'Internet, lors de la réception d'une requête il est essentiel de pouvoir authentifier l'expéditeur et de pouvoir lui accorder une autorisation en toute confiance. Dans ce papier nous nous intéressons à l'une des solutions proposées pour résoudre ce problème primordial en sécurité : l'infrastructure à clés publiques SDSI/SPKI. Contrairement à la plupart des formalisations logiques exposées dans la littérature, nous développons d'une part une sémantique générale et intuitive, d'autre part une méthode de décision, fruit d'une intégration de différentes techniques d'IA pour le raisonnement sur les modalités, les identités et le temps.

Mots Clef

Démonstration automatique, représentation des connaissances.

Abstract

In a distributed environment like the Internet, when receiving a request, authentication (who the sender is) and authorization (what it can do) are the essential task to tackle. This paper focuses on one of the solutions introduced to solve this problem : SDSI/SPKI. Unlike many logical formalizations proposed in the literature we provide both a general and intuitive semantics, and a decision method that uses a combination of many AI techniques : namely reasoning about modalities, identities and time.

Keywords

Automated deduction, knowledge representation.

1 Introduction

La sécurisation des transactions électroniques est actuellement une des préoccupations majeures des utilisateurs d'Internet et un défi pour les chercheurs. Des problèmes tels que l'authentification (qui vous êtes) et l'autorisation (ce que vous pouvez faire), difficiles à résoudre lorsque "vous" n'est qu'une entité inconnue qui envoie des octets de l'autre bout du réseau, sont compliqués par le manque d'une autorité centralisée de confiance.

Par exemple les échanges de données "peer-to-peer" [7], la protection de la vie privée dans les services de santé [5], les infrastructures à clés publiques [26] mettent en œuvre des activités qui nécessitent authentification et autorisation. Des difficultés supplémentaires apparaissent du fait de la disparition progressive du modèle traditionnel d'interaction client/serveur dans lequel les ressources sont centralisées sur un serveur qui connaît les clients et leur fournit les services demandés auxquels ils peuvent prétendre. Tout d'abord les clients ne sont plus connus des serveurs : l'idée derrière les services web est que les requêtes puissent provenir de quiconque possédant les bonnes pièces justificatives. Deuxièmement, les serveurs eux-mêmes sont souvent distribués et leur politique de sécurité peut émaner de différentes sources et de différents domaines administratifs. Par exemple la politique de sécurité d'un serveur d'e-santé peut être une combinaison entre le respect de la loi "Informatique et Libertés" pour la protection de la vie privée, les règles locales de l'hôpital et le règlement de l'université de médecine sur la divulgation de données expérimentales. Troisièmement, les clients ne sont plus négligeables et peuvent posséder des données de valeur telles que de l'argent électronique ou telles que leur identité.

Les questions traditionnelles d'authentification et d'autorisation se posent désormais dans les termes suivants : "L'ensemble des pièces justificatives de l'identité et des permissions obtenues qui a été présenté prouve-t-il que la requête satisfait l'ensemble des politiques locales et les informa-

tions sur les identités ?"

Pour fournir une réponse pratique à ce type de questions, les chercheurs en sécurité ont proposé un certain nombre de solutions qui réalisent ces tâches sans infrastructure de sécurité centralisée (le lecteur intéressé trouvera un panorama de ces propositions dans [29]).

L'une des plus fréquemment citées est SDSI (Simple Distributed Security Infrastructure) de Lampson et Rivest [25], raffiné plus tard en SPKI (Simple Public Key Infrastructure) dans un RFC Internet [11]. Plusieurs propositions postérieures telles que Binder [10] sont bâties sur les idées de ces travaux. Très schématiquement, SDSI/SPKI permet d'accorder des permissions à des entités identifiées par leur clé, au moyen de *certificats*. Ceux-ci servent de pièces justificatives pour appuyer les requêtes faites lors d'échanges d'information, ainsi sécurisés.

L'aspect novateur de SDSI/SPKI réside essentiellement dans l'introduction du concept de *nom local* et dans le fait d'avoir réduit le problème traditionnel de décision d'accès et d'autorisation en un problème de vérification d'une combinaison de *pièces justificatives liant des noms locaux et globaux* et de *pièces justificatives liant des noms et des permissions*. Par exemple, dans Président du Comité de Programme de RFIA, Président du Comité de Programme est un nom local, vraisemblablement différent de Président du Comité de Programme de RJCIA. L'individu désigné comme Président du Comité de Programme peut être lié par un certificat à un individu nommé Charlet (et à un autre individu nommé Dhome par un deuxième certificat). De même le collègue de Charlet peut être associé à plus d'un individu. Supposons maintenant que le collègue de Charlet a obtenu la permission d'accéder à l'Ordinateur de Charlet. Une demande d'un collègue du Président du Comité de Programme de RFIA doit-elle être acceptée par le serveur pc-142a.jussieu.fr ? Le raisonnement se complique encore du fait du temps : en effet quelqu'un peut être président du comité de programme ou collègue pour un temps seulement. La demande précédente doit-elle alors être acceptée en 2004 ?

La proposition de SDSI/SPKI a fait l'objet d'un intense débat et de nombreux chercheurs ont tenté de formaliser cette proposition ou ses alternatives en utilisant des logiques diverses pour analyser et mettre l'accent sur des différences ou des subtilités. Abadi [1] utilise une logique modale, modifiée plus tard par Howell et Kotz [17] ; Halpern et van der Meyden [15, 16] proposent également une logique modale ; Jha, Reps *et al.* [18, 27] utilisent la vérification de modèles (model-checking) pour le fragment atemporel de la logique précédente. Des contre-propositions par Li *et al.* [21, 22] utilisent la programmation logique et datalog.

L'intérêt de ce problème de représentation de connaissances et de raisonnement réside dans le fait que sa solution nécessite la mise en œuvre de différentes techniques d'IA, jusqu'ici assez séparées. Tout d'abord nous devons représenter les permissions, puis la dénomination et les identités, enfin nous avons besoin de représenter les informations temporelles – qualitatives et quantitatives – telles que les dates d'expiration et les intervalles de validité.

Par ailleurs il est nécessaire de fournir des mécanismes de raisonnement. En effet, il y a peu d'intérêt à modéliser la politique du gestionnaire des papiers soumis du président du comité de programme de RFIA en SDSI/SPKI s'il n'est pas possible ensuite de décider si l'utilisateur connecté à distance et auteur de la requête signée de la clé 0xF34567 est autorisé à consulter les rapports des relecteurs du papier IA045.pdf. Pour la partie modale et celle concernant la dénomination, il faut combiner différents résultats de travaux avancés en logique modale et logique dynamique [23, 8]. Quant à la partie temporelle, le raisonnement qualitatif basé sur les CSP et propre à l'algèbre des intervalles d'Allen [3, 4] n'est pas suffisant. Les TCSP (Temporal Constraint Satisfaction Problems) et les STP (Simple Temporal Problems) sont nécessaires pour manipuler les relations temporelles métriques [9].

Jusqu'ici les approches proposées se sont focalisées sur l'utilisation de logiques permettant d'obtenir *la* sémantique de la description opérationnelle de SDSI/SPKI. Cela a donné naissance à des formalismes assez lourds, orientés vers l'application et dépendants de celle-ci, d'une manière ou d'une autre. Il faut au contraire un cadre général pour représenter et raisonner sur la dénomination, les identités et le temps. Il faut ensuite montrer que des restrictions particulières sur les modèles permettent de capturer les propriétés souhaitées. Par exemple dans les logiques modales il n'y a pas "la" sémantique pour modéliser la connaissance, mais des structures de Kripke ; la notion particulière de la connaissance (introspection positive ou négative, ...) adaptée à notre application est obtenue en imposant différentes restrictions sur les modèles.

Nous basant sur des tentatives antérieures de formalisation par Abadi, Halpern et van der Meyden, Jha et Reps, nous proposons un modèle général de raisonnement sur la dénomination et les identités, l'autorisation, les pièces justificatives *et* le temps. Nous espérons avoir produit l'équivalent de ce que Syverson a appelé "une sémantique indépendamment motivée" (an independently motivated semantics) [28]. La sémantique proposée est très intuitive. En particulier elle permet de raisonner de façon naturelle tout à la fois sur le temps, l'intersection d'intervalles de validité et la dénomination.

Par ailleurs nous proposons une méthode de décision qui associe des méthodes de tableaux avancées pour les logiques modales et de description avec des techniques de raisonnement pour l'algèbre des intervalles d'Allen et des propositions mélant des contraintes à la fois qualitatives et quantitatives.

Dans la suite du papier nous introduisons SDSI/SPKI, puis nous présentons le modèle sémantique que nous proposons. Enfin nous exposons la méthode de décision et la preuve de correction et complétude avant de conclure.

2 Présentation de SDSI/SPKI

SDSI/SPKI [11] est basé sur l'idée que dans un système distribué sans contrôle centralisé, les services proposés évoluent rapidement. Donc l'ensemble des actions possibles et les utilisateurs pouvant les requérir ne sont pas connus à l'avance. Néanmoins, les serveurs ont des politiques (exprimées au moins en termes d'identificateurs locaux) et des chaînes de confiance relient les différents sites. Ceci implique que les autorisations et l'information sur les identités devront être créées, stockées et gérées à la volée. À la limite on pourrait attendre des utilisateurs qu'ils rassemblent toutes les pièces justificatives nécessaires pour autoriser une action et qu'ils les présentent en même temps que leur requête. Cependant la plupart de ces pièces justificatives ne sont pas sous le contrôle d'un service d'autorisation, il faut donc éviter qu'un utilisateur malveillant ne puisse les manipuler. Par conséquent les méthodes de vérification des pièces justificatives à clés publiques doivent être intégrées à l'infrastructure.

L'absence d'autorité de dénomination centralisée est caractéristique du modèle SDSI/SPKI. Chaque agent possède un ensemble de *noms locaux*, désignés par n, éventuellement indicé. Un nom peut être *composé*. Par exemple si l'on suppose que pour l'agent $Labévue$ le nom local $Gaston$ désigne $Lagaffe$ alors $Gaston's\ Copain$ désigne ce que $Lagaffe$ considère comme un copain. Dans SPKI les noms composés sont dénotés par le tuple `(name n1 n2 ...nk)` ou par l'expression équivalente $n_1's\ n_2's\ \ldots n_k$ où chaque `n-i` est un nom local et n_1 est soit un nom local, soit une clé publique. Dans ce dernier cas le nom est dit *complètement qualifié*.

L'interprétation d'un nom composé dépend de l'agent sauf pour les noms complètement qualifiés. Ainsi l'interprétation de $Gaston's\ Collègues - de - bureau$ par un agent dépend de son interprétation de $Gaston$ qui peut être différente de l'interprétation de $Gaston$ (et par suite de $Gaston's\ Collègues - de - bureau$) par un autre agent.

En l'absence d'une autorité de dénomination centralisée, il n'y a pas de noms globaux dans SPKI et les seules entités globales sont les clés publiques. Donc, à proprement parler, dans SPKI il n'y a pas d'agent interprétant des noms mais juste des clés. Ainsi on pourrait dire que l'agent $Lagaffe$ est l'interprétation de $Gaston$ par l'agent $Labévue$, en fait on peut seulement dire qu'à $Gaston$ est associée la clé publique k_g, de sorte que la clé publique k_l (qui correspond à ce que l'on appelle l'agent $Labévue$) considérera toute pièce vérifiable avec la clé publique k_g comme un message provenant de son camarade $Gaston$. Ceci n'est pas seulement une simplification, c'est ce qui se passe réellement dans un réseau : parler d'agents comme d'humains est une représentation fictive commode mais certainement pas une bonne représentation pour une suite d'octets venant de l'autre bout du réseau [14].

SPKI possède d'autres types de noms tels que le mot réservé "Self" qui désigne l'entité effectuant la vérification. Suivant l'exemple de Jha et Reps, nous ne considérerons ici que des noms composés.

En ce qui concerne les pièces justificatives, elles sont représentées par des certificats. Il en existe différents types dans SPKI : des certificats de dénomination, des certificats d'autorisation et des listes de révocation de certificats. Nous traiterons ici les deux premiers types.

Un *certificat de dénomination* a la forme d'un message signé cryptographiquement `(cert (issuer (name k n)) (subject p) valid)`, où `k` est une clé (qui représente l'expéditeur, dont la signature est sur le message), `n` est un nom local, `p` est un nom complètement qualifié et `valid` est une section optionnelle décrivant les contraintes de validité temporelle du certificat. Cette section décrit un intervalle de temps durant lequel le certificat est valide. Elle peut aussi décrire une suite de tests additionnels pour la validation de certificats. Nous dénotons un certificat de dénomination par $k\ \mathsf{cert}\ n \mapsto p : [t_1, t_2]$ dont l'interprétation est la suivante : pour l'agent k le nom local n est lié au nom complètement qualifié p pendant la période comprise entre les instants t_1 et t_2.

Un *certificat d'autorisation* est de la forme `(cert (issuer k) (subject p) (propagate) A valid)`, où `k` est une clé, `p` est un nom complètement qualifié, `A` est une autorisation (en substance un ensemble d'actions), et `valid` est une section pour la validité temporelle, comme ci-dessus. La section `(propagate)` est optionnelle. Intuitivement, l'expéditeur utilise un tel certificat pour attribuer à `p` qualité pour exécuter les actions de `A`. De plus, si le champ de propagation est présent, le sujet `p` peut également déléguer ce droit à d'autres. Nous dénotons un certificat d'autorisation par $k\ \mathsf{cert}\ \mathsf{Perm}(p, a) : [t_1, t_2]$. Nous ne traiterons pas ici du problème de la délégation.

Il faut noter que contrairement à la modélisation des problèmes d'autorisation généralisée de [27], le temps est ici présent dans les certificats. Ainsi nous pouvons raisonner avec des certificats concomitants ou non.

3 Syntaxe pour SPKI

La logique que nous considérons généralise les propositions de [1, 15, 16, 18, 27].

Nous avons tout d'abord un ensemble d'actions $\mathfrak{A}$, un ensemble de clés $\mathfrak{K}$, et un ensemble de noms $\mathfrak{N}$. Les agents ou entités sont appelés *principals* et sont dénotés par

- k où $k \in \mathfrak{K}$,
- n où $n \in \mathfrak{N}$,
- ou bien $p's\ q$ où p et q sont des principals.

Les *formules atomiques* de notre logique sont

- $p \mapsto q$ où p et q sont des principals,
- $\mathsf{Perm}(p, a)$ où p est un principal et a est une action.

Les formules composées sont ensuite construites à l'aide des opérateurs usuels de négation, conjonction et disjonction. Intuitivement si $p \mapsto q$ (que l'on peut lire "p parle pour q" ou "q est un p") alors pour l'agent qui évalue la formule toute autorisation accordée à p l'est également à q.

Comme dit précédemment il existe deux formes de *certificats* :
– k cert $p \mapsto q : [t_1, t_2]$ où $t_1, t_2 \in \mathbb{R}$
– et k cert $\mathsf{Perm}(p, a) : [t_1, t_2]$.
La première est une légère généralisation de SPKI dans la mesure où un sujet peut être un nom composé et pas seulement un nom local.
Remarquons qu'un certificat logique ne correspond pas nécessairement à un certificat physique. En réalité ce qui nous intéresse principalement dans la formalisation d'un système à base de pièces justificatives c'est la possibilité de répondre à la question :

> Étant donné un ensemble de certificats physiques avec certaines périodes de validité, une requête pour une permission donnée devrait-elle être acceptée dans un intervalle de temps donné ?

en répondant à la question suivante (et vice versa)

> Est-ce qu'un certificat logique pour une permission donnée dans un intervalle de temps donné est une conséquence logique d'un ensemble de certificats physiques avec certaines périodes de validité ?

La notion de conséquence ne pourra évidemment être définie qu'une fois donnée une sémantique.
Par rapport à la proposition de [2, 1], nous avons éliminé l'opérateur $says$ subsumé par l'opérateur cert . Le lecteur intéressé trouvera une méthode de raisonnement automatique pour des fragments du calcul d'Abadi *et al.* dans [23].
Contrairement à toutes les approches antérieures, les *intervalles de validité symboliques* et les *instants symboliques* sont permis. Par exemple il est possible de certifier que Charlet est un Président du Comité de Programme de RFIA jusqu'à RFIA04, sachant que la période considérée par le certificat chevauche 2004, débute au moment de la désignation des Présidents et se termine au plus tard à la fin du congrès. Le formalisme présenté section 5 peut prendre en charge de telles contraintes symboliques. Il suffit pour cela d'adapter légèrement les règles proposées (cela ne sera pas exposé dans ce papier).

4 Sémantique

La devise de tout système basé sur les pièces justificatives tel SPKI pourrait être "hors des clés publiques, point de salut" et notre modèle est construit sur cette intuition : les entités de base sont les clés et les noms permettent de relier les clés entre elles et avec les permissions.
Du point de vue des logiques de description, les clés peuvent être vues comme des individus, les autorisations comme des concepts et les noms comme des rôles connectant des individus. En logique dynamique les clés sont des états, les noms des programmes qui permettent de passer d'un état à l'autre, à $'s$ peut être associé l'opérateur de composition séquentielle et les certificats peuvent être vus comme une certaine forme d'opérateur de nécessitation [20]. Une différence majeure est la possibilité de nommer les états offerte par notre formalisme.
Nous donnons tout d'abord un modèle atemporel. Ainsi un *modèle* est une structure de Kripke généralisée, c'est-à-dire un triplet $\langle \mathcal{K}, \mathcal{N}, \mathcal{A} \rangle$ où $\mathcal{K}$ est un ensemble de clés sémantiques (tel que pour tout $k \in \mathfrak{K}$ il existe un $\underline{k} \in \mathcal{K}$). La relation de dénomination $\mathcal{N}$ est une fonction de l'ensemble des noms locaux vers un sous-ensemble de relations sur les clés $\mathfrak{N} \longrightarrow \mathcal{K} \longrightarrow 2^{\mathcal{K}}$, autrement dit une famille de relations indexées. La relation d'octroi $\mathcal{A}$ est une fonction qui associe aux actions également des sous-ensembles de relations sur les clés $\mathfrak{A} \longrightarrow \mathcal{K} \longrightarrow 2^{\mathcal{K}}$.
La signification intuitive de l'ensemble $\mathcal{K}$ devrait être évidente. Quant à la relation de dénomination elle associe à tout nom local une relation sur les clés : $\langle k, k' \rangle \in \mathcal{N}(n)$, ou de façon équivalente $k \, \mathcal{N}_n \, k'$, si le principal associé à la clé k associe au nom n au moins la clé k'. Il s'agit d'une *relation* de dénomination car le même nom n pourrait faire référence à plusieurs individus comme dans $Gaston's \ Collègues - de - bureau$. Par conséquent il est plus commode d'utiliser la notation inspirée de la sémantique de Kripke : $k' \in \mathcal{N}_n(k)$ ou $k' \in \mathcal{N}(n, k)$.
En ce qui concerne la relation d'octroi, elle associe à chaque action l'ensemble des possesseurs de clés qui acceptent de la permettre à d'autres principals. Ainsi $k' \in \mathcal{A}(a, k)$ signifie que le principal associé à la clé k autorise la clé k' à exécuter l'action a.
$\mathcal{N}$ peut être étendu pour les clés et les noms composés en donnant une sémantique à l'opérateur $'s$:
– $\mathcal{N}(k_2, \underline{k}_1) = \{\underline{k}_2\}$ où $k_2 \in \mathfrak{K}$
– $\mathcal{N}(p's \, q, \underline{k}_1) = \bigcup_{\underline{k} \in \mathcal{N}(p, \underline{k}_1)} \mathcal{N}(q, \underline{k})$
Le premier cas impose qu'à toute clé syntaxique soit associée la clé sémantique correspondante. Et le second cas est une simple représentation de la composition : si k associe k_p au nom p et k_p associe k_q au nom q alors k devrait associer k_q au nom $p's \ q$.
On peut remarquer qu'avec cette sémantique tout principal de la forme $p_1 \ 's \ p_2 \ 's \ \ldots \ 's \ p_l$ est équivalent à un principal $q_1 \ 's \ q_2 \ 's \ \ldots \ 's \ q_m$ où q_1 est soit une clé soit un nom et $q_2, \ldots, q_m$ sont des noms. C'est la raison pour laquelle nous ne considérerons désormais que des principals de la dernière forme.
Pour intégrer le temps à notre structure, nous introduisons le concept de *trace*, c'est-à-dire de fonction du temps vers les modèles. Une trace $\mathcal{M}$ associe un modèle à chaque instant du temps : $\mathcal{M}_t = \langle \mathcal{K}_t, \mathcal{N}_t, \mathcal{A}_t \rangle$ où $t \in \mathbb{R}$. Par suite $\mathcal{K}_t$ est l'ensemble des clés sémantiques (tel que pour tout $k \in \mathfrak{K}$ il existe une clé sémantique $\underline{k} \in \mathcal{K}_t$) associées aux clés utilisées à l'instant t, $\mathcal{N}_t$ est la relation de dénomination à l'instant t et $\mathcal{A}_t$ est la relation d'octroi à l'instant t.
La sémantique des principals est similaire, sauf pour l'indice t :
– $\mathcal{N}_t(k_2, \underline{k}_1) = \{\underline{k}_2\}$ où $k_2 \in \mathfrak{K}$
– $\mathcal{N}_t(p's \, q, \underline{k}_1) = \bigcup_{\underline{k} \in \mathcal{N}(p, \underline{k}_1)} \mathcal{N}(q, \underline{k})$
À ce stade, la comparaison avec des variations possibles de la logique est intéressante. Devrions-nous avoir persistance des clés, c'est-à-dire $\mathcal{K}_t = \mathcal{K}_{t'}$ pour tout t

et t' ? Cela signifierait grosso modo que toutes les clés ont déjà été créées, ce qui paraît peu raisonnable. On peut également se demander si les périodes de validité des clés devraient toujours être des intervalles connexes. Ainsi supposons que dans notre modèle nous ayons $k_1 \in \mathcal{N}_{30-octobre-2003}(k_2, RFIA)$, c'est-à-dire que la clé réelle k_1 (par exemple la clé du comité de pilotage de RFIA) associe la clé k_2 au Président du Comité de Programme de RFIA. Le certificat expire au bout d'une année. Nous avons donc $k_1 \notin \mathcal{N}_{30-octobre-2004}(k_2, RFIA)$. Voulons-nous maintenant imposer que pour tout instant t postérieur au 30 octobre 2004, $k_1 \notin \mathcal{N}_t(k_2, RFIA)$? Ceci signifierait qu'une même personne ne pourrait être présidente du comité de programme de RFIA qu'une seule fois. Plus généralement, une fois un certificat expiré, il ne serait pas possible d'accepter la revalidation de ce certificat pour une même clé et une période ultérieure. C'est un des modèles possibles, souhaitable dans certains cas et indésirable dans d'autres.

Nous avons maintenant tout le matériel nécessaire pour donner une sémantique aux *formules* :

- $\mathcal{M}_t, \underline{k}_1 \models p \mapsto q$ ssi $\forall \underline{k} \in \mathcal{N}_t(q, \underline{k}_1), \underline{k} \in \mathcal{N}_t(p, \underline{k}_1)$
- $\mathcal{M}_t, \underline{k}_1 \models \mathsf{Perm}(p, a)$ ssi $\forall \underline{k} \in \mathcal{N}_t(p, \underline{k}_1), \underline{k} \in \mathcal{A}_t(a, \underline{k}_1)$

Quant aux *certificats*, nous les évaluons comme suit :

- $\mathcal{M}_t \models k$ cert $p \mapsto q : [t_1, t_2]$ ssi $t \in [t_1, t_2] \Rightarrow \mathcal{M}_t, \underline{k} \models p \mapsto q$
- $\mathcal{M}_t \models k$ cert $\mathsf{Perm}(p, a) : [t_1, t_2]$ ssi $t \in [t_1, t_2] \Rightarrow \mathcal{M}_t, \underline{k} \models \mathsf{Perm}(p, a)$

Il faut noter qu'en comparaison avec les propositions de Halpern et van der Meyden [15, 16], nous n'obligeons pas les certificats à être satisfaits par un modèle. Un modèle est une entité indépendante des certificats. Il a des propriétés et il satisfait certaines formules. S'il satisfait des formules particulières alors il satisfait aussi certains certificats. De cette façon, comme nous l'avons déjà dit, une théorie de certificats déterminée caractérise un ensemble de modèles déterminé.

Nous pouvons maintenant définir la notion de *satisfaction par une trace* $\mathcal{M}$, c'est-à-dire de satisfaction à tout instant : $\mathcal{M} \models k$ cert $c : [t_1, t_2]$ ssi $\forall t, \mathcal{M}_t \models k$ cert $c : [t_1, t_2]$

Comme dans [18, 27], nous avons une notion de conséquence pour une chaîne de certificats :

Définition 1 *Un certificat χ est une* conséquence logique *d'un ensemble de certificats $\mathcal{C}$ si toute trace qui satisfait tous les certificats de $\mathcal{C}$ satisfait aussi χ.*

5 Tableaux

La question qui se pose maintenant est la suivante : comment savoir qu'un certificat est une conséquence logique d'un ensemble de certificats ? Et si ce n'est pas le cas, comment obtenir un contre-exemple ?

Ce n'est pas la seule question intéressante. Supposons que l'on nous donne une trace. Quelles contraintes supplémentaires devrait satisfaire cette trace pour satisfaire un certain ensemble de certificats ?

Afin d'apporter une réponse à ces questions nous développons une méthode de décision basée sur la méthode des tableaux et qui fait correspondre la vérification d'une conséquence logique à l'échec de la construction d'un contre-modèle. Intuitivement, pour montrer qu'un certificat χ est conséquence logique d'un ensemble de certificats $\mathcal{C}$, on tente simplement d'en construire un contre-modèle. En cas de succès on obtient un contre-modèle, sinon le certificat est bien conséquence logique de l'ensemble de certificats donné.

Nous utiliserons la terminologie usuelle pour les tableaux. On peut par exemple se référer à [8] pour la partie modale et [19] pour la partie temporelle.

La construction débute par les formules et tente de bâtir le modèle entier de façon incrémentale, en construisant les associations de dénomination, en attribuant les permissions et en déterminant les informations temporelles. Ainsi un *tableau* est une collection de branches, chacune correspondant intuitivement à la construction d'un modèle possible.

Une *branche* a trois composants : pour l'*information atemporelle* (telle que les relations de dénomination), l'*information temporelle qualitative* et l'*information temporelle quantitative*.

Pour l'information atemporelle nous avons un triplet $\langle \mathtt{K}, \mathtt{N}, (\mathtt{F}, \mathtt{A}) \rangle$ où

- $\mathtt{K}$ est l'ensemble des clés syntaxiques plus éventuellement les nouvelles clés introduites au cours du processus,
- la fonction $\mathtt{N} : \mathtt{K} \times \mathtt{K} \longrightarrow 2^{\mathfrak{N}}$ correspond à la relation de dénomination locale construite jusqu'ici,
- la fonction $\mathtt{A} : \mathtt{K} \times \mathtt{K} \longrightarrow 2^{\mathfrak{A}}$ correspond à la relation d'octroi construite jusqu'ici,
- $\mathtt{F} : \mathtt{K} \longrightarrow 2^{Formules}$ correspond aux formules (étiquetées par un intervalle de validité) à satisfaire.

Pour l'information temporelle qualitative nous avons un *réseau d'intervalles* $\langle \mathtt{V}, \mathtt{E} \rangle$ où

- $\mathtt{V}$ est un ensemble de variables représentant des intervalles temporels,
- la fonction $\mathtt{E} : \mathtt{V} \times \mathtt{V} \longrightarrow 2^{relations\ d'Allen}$ correspond aux relations temporelles qualitatives imposées jusqu'ici.

Les relations d'Allen [3] sont les 13 relations qui correspondent à toutes les positions relatives possibles de deux intervalles de la droite non ponctuels. Dans un réseau d'intervalles la relation entre deux variables est définie par un ensemble de relations d'Allen – ensemble qui traduit une disjonction.

Si v est une variable d'intervalle, v^- en dénote la borne inférieure et v^+ la borne supérieure.

Pour l'information temporelle métrique nous avons un *réseau de points* $\langle \mathtt{V}_T, \mathtt{E}_T \rangle$ où

- $\mathtt{V}_T$ est un ensemble de variables représentant des points temporels,
- $\mathtt{E}_T : \mathtt{V}_T \times \mathtt{V}_T \longrightarrow 2^{Intervalles}$ représente les contraintes métriques entre points temporels construites jusqu'ici.

Dans un réseau de points une relation entre deux variables est modélisée par un ensemble d'intervalles qui représentent les valeurs possibles pour la durée entre les deux points.

Construction. Après une étape d'initialisation, la construction s'effectue pas à pas, une étape correspondant à l'application d'une règle et la vérification de la cohérence de l'information temporelle. La construction s'arrête lorsque toutes les expressions ont été traitées ou lorsqu'une incohérence est détectée.

5.1 Initialisation

Au tout début $\mathtt{K}$ est l'ensemble des clés apparaissant dans $\{\chi\} \cup \mathcal{C}$. $\mathtt{F}$, $\mathtt{N}$, $\mathtt{A}$, $\mathtt{V}$ et $\mathtt{V}_T$ sont vides.
Pour chaque certificat k cert $c : [t_1, t_2]$ de $\{\chi\} \cup \mathcal{C}$, une nouvelle variable d'intervalle v_ω est ajoutée à $\mathtt{V}$ et v_ω^- et v_ω^+ sont ajoutés à $\mathtt{V}_T$. Puis $\mathtt{E}$, resp. $\mathtt{E}_T$, est enrichi des contraintes existant entre v_ω, resp. v_ω^- et v_ω^+, et le reste des variables d'intervalles, resp. de points. Intuitivement v_ω est la variable d'intervalle utilisée pour définir la période de validité du certificat.
Exemple Soit $\mathcal{C}_N = \{k_1 \text{ cert } (p \mapsto k_3) : [1,4]\}$ et $\mathcal{C}_A = \{k_2 \text{ cert } \mathsf{Perm}(k_1\,'s\ p, a) : [2,6]\}$. Alors $\mathtt{V} = \{v_1, v_2\}$ où v_1 correspond à $[1,4]$ et v_2 à $[2,6]$, $\mathtt{V}_T = \{v_1^-, v_1^+, v_2^-, v_2^+\}$, $\mathtt{E}(v_1, v_2) = \{\mathsf{overlaps}\}$, $\mathtt{E}_T(v_1^-, v_1^+) = \{[3,3]\}$, $\mathtt{E}_T(v_2^-, v_2^+) = \{[4,4]\}$, $\mathtt{E}_T(v_1^-, v_2^-) = \{[1,1]\}$ (les autres contraintes métriques étant déductibles de $\mathtt{E}_T$).

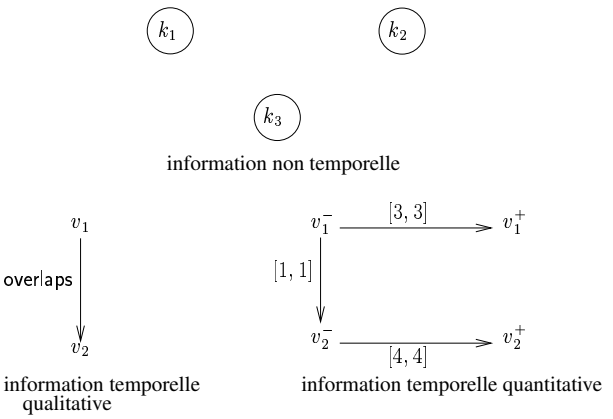

FIG. 1 – Illustration de l'exemple

Observons la figure 1 : à l'origine il y a les trois clés des deux certificats et, pour le moment, aucune relation de dénomination entre elles.
Nous introduisons une variable d'intervalle pour la validité du premier, resp. second, certificat : v_1, resp. v_2. Puis nous imposons que les deux intervalles se chevauchent (puisque [1,4] chevauche [2,6]), d'où la relation d'Allen $\mathsf{overlaps}$. Pour les contraintes métriques, la durée de la période allant du début de v_1 au début de v_2 se trouve comprise dans l'intervalle $[1,1]$, car $v_1 = [1,4]$ et $v_2 = [2,4]$, donc $2-1=1$. De même la durée de v_1, c'est-à-dire la différence entre $v_1^- = 1$ et $v_1^+ = 4$ est comprise dans l'intervalle $[3,3]$.

Enfin si le certificat χ est de la forme k_i cert $c : [t_{i1}, t_{i2}]$, alors $\{\neg c : v_i\}$ où v_i correspond à $[t_{i1}, t_{i2}]$, est ajouté à $\mathtt{F}(k_i)$. Intuitivement on cherche un modèle dans lequel le certificat ne soit pas satisfait pour l'intervalle donné.

5.2 Règles de construction

Dans la définition des règles nous utiliserons des abréviations pour décrire les relations (d'Allen) possibles entre les intervalles de validité des certificats.

Définition 2 *Soit v_1, v_2 et v_3 des variables d'intervalle.*
- $v_1 \cap v_2 = \emptyset$ *quand la contrainte suivante est satisfaite :* $v_1\{\mathsf{before}, \mathsf{meets}, \mathsf{met-by}, \mathsf{after}\}v_2$
- $v_1 \cap v_2 \neq \emptyset$ *quand la contrainte suivante est satisfaite :* $v_1\{\mathsf{overlaps}, \mathsf{overlapped-by}, \mathsf{starts}, \mathsf{started-by}, \mathsf{during}, \mathsf{contains}, \mathsf{finishes}, \mathsf{finished-by}, \mathsf{equals}\}v_2$
- $v_2 \subseteq v_1$ *quand la contrainte suivante est satisfaite :* $v_2\{\mathsf{starts}, \mathsf{during}, \mathsf{finishes}, \mathsf{equals}\}v_1$

Par exemple, pour le premier cas, deux intervalles ont une intersection vide si l'un des intervalles est strictement avant l'autre ou bien si l'un des intervalles se termine au moment où l'autre commence.

Définition 3 *Soit $v_3 = v_1 \cap v_2$.*
- *si* $v_1\{\mathsf{starts}, \mathsf{during}, \mathsf{finishes}, \mathsf{equals}\}v_2$ *alors* $v_3 = v_1$
- *si* $v_1\{\mathsf{started-by}, \mathsf{contains}, \mathsf{finished-by}\}v_2$ *alors* $v_3 = v_2$
- *si* $v_1\ \mathsf{overlaps}\ v_2$ *alors* $v_3\ \mathsf{finishes}\ v_1$ *et* $v_3\ \mathsf{starts}\ v_2$
- *si* $v_1\ \mathsf{overlapped-by}\ v_2$ *alors* $v_3\ \mathsf{starts}\ v_1$ *et* $v_3\ \mathsf{finishes}\ v_2$

Nous avons maintenant tous les outils nécessaires pour introduire les règles.

Certificats.

Dénomination Si k cert $(p \mapsto q) : [t_1, t_2] \in \mathcal{C}_N$ alors $\mathtt{F}(k) \leftarrow \mathtt{F}(k) \cup \{p \mapsto q : v\}$ où $v \in \mathtt{V}$ correspond à $[t_1, t_2]$

Autorisation Si k cert $\mathsf{Perm}(p, a) : [t_1, t_2] \in \mathcal{C}_A$ alors $\mathtt{F}(k) \leftarrow \mathtt{F}(k) \cup \{\mathsf{Perm}(p, a) : v\}$ où $v \in \mathtt{V}$ correspond à $[t_1, t_2]$

Intuitivement, si le certificat k cert $f : [t_1, t_2]$ est valide alors la formule correspondante f doit être vraie pour la clé correspondante k pendant l'intervalle v (correspondant à $[t_1, t_2]$), ce qui revient à ajouter la formule $f : v$ à l'étiquette du nœud k dans la partie "information atemporelle" du modèle.

Formules. Nous ne considérerons que le cas des formules atomiques. En effet les opérateurs booléens sont traités de façon classique : par exemple la disjonction divise la branche courante en deux sous-branches.

Porte-parole Ces règles peuvent se partager en deux groupes selon que la formule traitée est positive ou négative.
Pour les formules positives :
- si $p \mapsto k' : v \in \mathtt{F}(k)$ alors $\mathtt{N}(k, k') \leftarrow \mathtt{N}(k, k') \cup \{p : v\}$

– si $p \mapsto k'\ 's\ q : v \in \mathtt{F}(k)$ alors $\forall k_3 \in \mathtt{K}$, si $q : v' \in \mathtt{N}(k', k_3)$ alors soit $v \cap v' = \emptyset$, soit $v \cap v' \neq \emptyset$ et $\mathtt{N}(k, k_3) \leftarrow \mathtt{N}(k, k_3) \cup \{p : v_3\}$ où $v_3 = v \cap v'$
– si $p \mapsto n\ 's\ q : v \in \mathtt{F}(k)$ (où q peut être vide) alors $\forall k' \in \mathtt{K}$, si $n\ 's\ q : v' \in \mathtt{N}(k, k')$ alors soit $v \cap v' = \emptyset$, soit $v \cap v' \neq \emptyset$ et $\mathtt{N}(k, k') \leftarrow \mathtt{N}(k, k') \cup \{p : v_3\}$ où $v_3 = v \cap v'$

Intuitivement la première règle dit que si la clé k associe p à la clé k' alors cet étiquetage est ajouté à la relation de dénomination de k vers k', avec l'intervalle de validité approprié v. La figure 2 est une représentation graphique de cette règle.

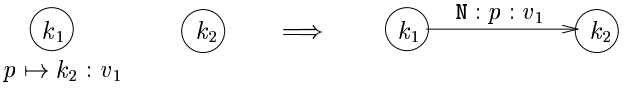

FIG. 2 – Illustration de la 1re règle pour $\mapsto$ (les graphes temporels sont inchangés)

En ce qui concerne la deuxième règle, supposons que l'on sache que la clé k associe le nom p au nom $k'\ 's\ q$ pour un certain intervalle de validité v. On cherche alors toutes les clés nommées q par k'. Chacune de ces relations de dénomination a aussi une période de validité, disons v' pour l'une d'entre elles. Alors de deux choses l'une : soit les périodes de validité ne se chevauchent pas (c'est-à-dire $v \cap v' = \emptyset$) – dans ce cas il n'y a rien à faire –, soit elles se chevauchent et il faut alors enchaîner les certificats sur les périodes simultanées, à savoir $v_3 = v \cap v'$.
Cette deuxième règle est illustrée par la figure 3 dans le cas particulier où v_1 chevauche v_2 (v_1 overlaps v_2). Comme nous pouvons le voir sur la figure, un lien de dénomination de k_1 vers k_3 a été ajouté mais celui-ci n'est valable que pour la période commune v_3. Les graphes temporels, quant à eux, spécifient que v_3 débute en même temps que v_2 et se termine en même temps que v_1.
La dernière règle suit le même schéma que la règle précédente. L'application de ces règles permet de compléter petit à petit les relations de dénomination entre les clés et de spécifier les contraintes entre les différentes périodes de validité.
Il n'y a qu'une règle pour les formules négatives :
– si $\neg(p \mapsto q) : v_1 \in \mathtt{F}(k_1)$ alors $\mathtt{F}(k_1) \leftarrow \mathtt{F}(k_1) \cup \{\ \mathsf{Perm}(p, a_\omega) : v_1, \neg\ \mathsf{Perm}(q, a_\omega) : v_1\}$, où a_ω est une nouvelle action

Cette règle stipule que q n'est pas un p pendant la période v_1 si p peut exécuter une action que q n'a pas le droit d'exécuter pendant cet intervalle v_1.

Permission Ces règles sont similaires aux règles précédentes, sauf qu'elles spécifient des actions permises au lieu de connecter des clés par des noms.
– si $\mathsf{Perm}(k_2, a) : v_1 \in \mathtt{F}(k_1)$ alors $\mathtt{A}(k_1, k_2) \leftarrow \mathtt{A}(k_1, k_2) \cup \{a : v_1\}$
– si $\mathsf{Perm}(k_2\ 's\ q, a) : v_1 \in \mathtt{F}(k_1)$ alors $\forall k_3 \in \mathtt{K}$, si $q : v_2 \in \mathtt{N}(k_2, k_3)$ alors soit $v_1 \cap v_2 = \emptyset$, soit $v_1 \cap v_2 \neq \emptyset$ et $\mathtt{A}(k_1, k_3) \leftarrow \mathtt{A}(k_1, k_3) \cup \{a : v_3\}$ où $v_3 = v_1 \cap v_2$

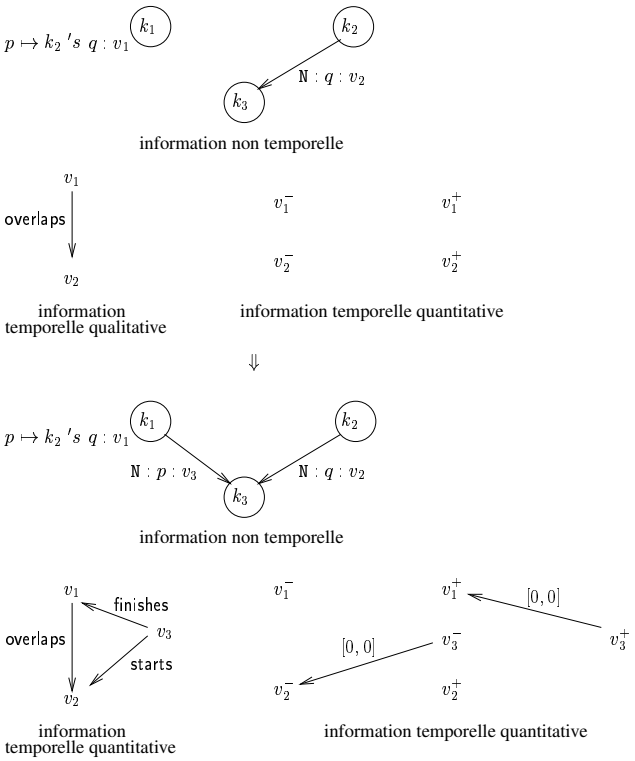

FIG. 3 – Illustration de l'une des règles pour $\mapsto$ (où v_1 overlaps v_2)

– si $\mathsf{Perm}(n\ 's\ q, a) : v_1 \in \mathtt{F}(k_1)$ (où q peut être vide) alors $\forall k_2 \in \mathtt{K}$, si $n\ 's\ q : v_2 \in \mathtt{N}(k_1, k_2)$ alors soit $v_1 \cap v_2 = \emptyset$, soit $v_1 \cap v_2 \neq \emptyset$ et $\mathtt{A}(k_1, k_2) \leftarrow \mathtt{A}(k_1, k_2) \cup \{a : v_3\}$ où $v_3 = v_1 \cap v_2$

Les règles pour les formules négatives sont les suivantes :
– si $\neg\ \mathsf{Perm}(k_2, a) : v_1 \in \mathtt{F}(k_1)$ alors $\mathtt{A}(k_1, k_2) \leftarrow \mathtt{A}(k_1, k_2) \cup \{\neg a : v_\omega\}$ où $v_\omega \subseteq v_1$ est un nouvel intervalle
– si $\neg\ \mathsf{Perm}(k_2\ 's\ q, a) : v_1 \in \mathtt{F}(k_1)$ alors $\mathtt{N}(k_2, k_\omega) \leftarrow \{q : v_\omega\}$ et $\mathtt{A}(k_1, k_\omega) \leftarrow \{\neg a : v_\omega\}$, où $k_\omega \in \mathtt{K}$ est une nouvelle clé et $v_\omega \subseteq v_1$ un nouvel intervalle
– si $\neg\ \mathsf{Perm}(n\ 's\ q, a) : v_1 \in \mathtt{F}(k_1)$ (où q peut être vide) alors $\mathtt{N}(k_1, k_\omega) \leftarrow \{n\ 's\ q : v_\omega\}$ et $\mathtt{A}(k_1, k_\omega) \leftarrow \{\neg a : v_\omega\}$, où $k_\omega \in \mathtt{K}$ est une nouvelle clé et $v_\omega \subseteq v_1$ un nouvel intervalle

Ici l'idée est qu'une clé n'a pas le droit d'exécuter une action pendant une période donnée à partir du moment où il existe un sous-intervalle de la période pendant lequel cette interdiction est effective.

Principals. Ces règles simplifient les relations de dénomination en éliminant les noms composés, mais permettent aussi de les composer – dans la mesure où les intervalles de validité le permettent :
– si $k_3\ 's\ q : v_1 \in \mathtt{N}(k_1, k_2)$ alors $\mathtt{N}(k_3, k_2) \leftarrow \mathtt{N}(k_3, k_2) \cup \{q : v_1\}$
– si $n\ 's\ q : v_1 \in \mathtt{N}(k_1, k_2)$ alors $\forall k_3 \in \mathtt{K}$, si $n : v_2 \in \mathtt{N}(k_1, k_3)$ alors soit $v_1 \cap v_2 = \emptyset$, soit $v_1 \cap v_2 \neq \emptyset$ et $\mathtt{N}(k_3, k_2) \leftarrow \mathtt{N}(k_3, k_2) \cup \{q : v_3\}$ où $v_3 = v_1 \cap v_2$

– si $q_1 : v_1 \in \mathtt{N}(k_1, k_2)$, $q_2 : v_2 \in \mathtt{N}(k_2, k_3)$ alors soit $v_1 \cap v_2 = \emptyset$, soit $v_1 \cap v_2 \neq \emptyset$ et $\mathtt{N}(k_1, k_3) \leftarrow \mathtt{N}(k_1, k_3) \cup \{q_1\ 's\ q_2 : v_3\}$ où $v_3 = v_1 \cap v_2$

5.3 Cohérence

Cohérence de la permission. Un principal (associé à une clé) ne peut avoir à la fois la permission et l'interdiction d'exécuter une même action au même moment :
– si $a : v_1 \in \mathtt{A}(k_1, k_2)$ et $\neg a : v_2 \in \mathtt{A}(k_1, k_2)$ alors $\mathtt{E}(v_1, v_2) = \mathtt{E}(v_1, v_2) \cap \{\mathsf{before}, \mathsf{after}, \mathsf{meets}, \mathsf{met-by}\}$

Cohérence de l'information temporelle. Schématiquement, les informations temporelles qualitatives et quantitatives sont traitées séparément puis combinées et remises à jour si une nouvelle information est apparue, sinon le calcul s'arrête. Si la relation vide est présente alors l'information initiale était incohérente [19]. L'algorithme est correct mais incomplet, sauf si le réseau d'intervalles est au moins *préconvexe* [6]. Pour assurer la complétude du traitement de l'information temporelle nous modifions légèrement certaines règles : au lieu d'introduire une contrainte de la forme $v_1 \cap v_2 = \emptyset$ nous insérons la contrainte $v_1\{\mathsf{before}, \mathsf{meets}\}v_2$ ou $v_1\{\mathsf{met-by}, \mathsf{after}\}v_2$.

Définition 4 *Une branche est dite* saturée *quand aucune nouvelle information ne peut apparaître par application d'une règle. Une branche est* fermée *si une incohérence est trouvée ; elle est* ouverte *lorsqu'elle est saturée mais pas fermée. Un tableau est fermé si toutes ses branches sont fermées et ouvert si l'une de ses branches l'est.*

La figure 4 est un tableau ouvert de l'exemple présenté précédemment.

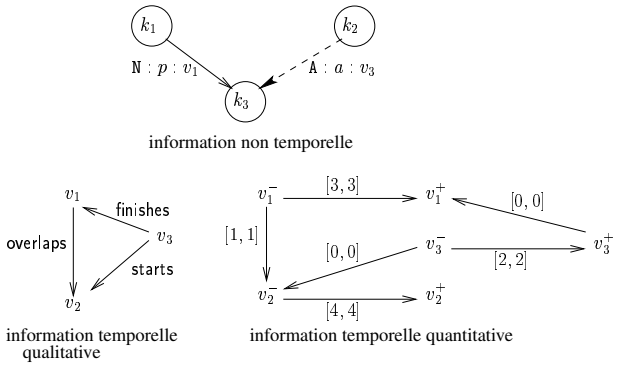

FIG. 4 – Un tableau (ouvert)

6 Correction et complétude

6.1 Correction

La preuve de la correction de la méthode proposée repose essentiellement sur le lemme 1. Celui-ci stipule que les règles maintiennent la "satisfaisabilité" des branches au fur et à mesure de la construction du tableau.

Définition 5 *Une branche B est SAT s'il existe une trace $\mathcal{M}$ et une application μ de B vers $\mathcal{M}$ telles que :*
– $\forall t \in \mathbb{R}$, $\forall k \in \mathtt{K}$, $\mu(k) \in \mathcal{K}_t$
– $\forall v \in \mathtt{V}$, $\mu(v)$ *est un intervalle sur les réels*
– $\forall e \in \mathtt{V}_T$, $\mu(e) \in \mathbb{R}$
– *si* $n : v \in \mathtt{N}(k_1, k_2)$ *alors* $\forall t \in \mu(v)$, $\mu(k_2) \in \mathcal{N}_t(\mu(n), \mu(k_1))$
– *si* $a : v \in \mathtt{A}(k_1, k_2)$ *alors* $\forall t \in \mu(v)$, $\mu(k_2) \in \mathcal{A}_t(\mu(a), \mu(k_1))$
– *si* $f : v \in \mathtt{F}(k)$ *alors* $\forall t \in \mu(v)$, $\mathcal{M}_t, \mu(k) \models f$
– *si* $\neg f : v \in \mathtt{F}(k)$ *alors* $\forall t \in v^*$, $\mathcal{M}_t, \mu(k) \not\models f$ *où* $v^* \subseteq \mu(v)$
et μ satisfait l'information temporelle :
– *si* $\mathsf{r} \in \mathtt{E}(v_1, v_2)$ *alors* $\mu(v_1)\ \mathsf{r}\ \mu(v_2)$
– *si* $v \in \mathtt{E}_T(e_1, e_2)$ *alors* $\mu(e_2) - \mu(e_1) \in \mu(v)$

Lemme 1 (Préservation de la satisfaisabilité) *Si une branche B est SAT alors l'application d'une règle à B donne une branche SAT.*

Preuve (idée) Supposons que B soit une branche SAT. Il existe une trace $\mathcal{M}$ et une application μ de B vers $\mathcal{M}$ qui vérifie la définition 5. En utilisant la sémantique (section 4) nous pouvons montrer qu'elle est toujours vérifiée après application d'une règle à B, éventuellement en étendant $\mathcal{M}$ ou μ. □

Théorème 1 (Correction) *Si un certificat χ a un tableau fermé, un ensemble de certificats $\mathcal{C}$ étant donné, alors χ est une conséquence logique de $\mathcal{C}$.*

Preuve (idée) La démonstration classique consiste à montrer qu'un certificat invalide donne un tableau ouvert. Si χ n'est pas une conséquence logique de $\mathcal{C}$ alors $\mathcal{C}$ et $\neg\chi$ sont satisfaisables par une trace $\mathcal{M}$. Donc la branche initiale du tableau pour χ étant donné $\mathcal{C}$ est SAT. Nous obtenons alors la conclusion grâce au lemme 1 et à la saturation du tableau. □

6.2 Complétude

En ce qui concerne la complétude, nous commençons par saturer le tableau (c'est-à-dire toutes les branches du tableau). Puis si une branche B est ouverte, nous pouvons alors construire une trace à partir de celle-ci de telle sorte que B soit SAT par rapport à une certaine application, cette trace étant un contre-modèle du certificat initial.

Définition 6 *Une* instantiation *des réseaux temporels d'une branche associe un intervalle réel à chaque variable d'intervalle et un nombre réel à chaque variable de point de façon à satisfaire les contraintes qualitatives et métriques entre variables temporelles.*

Définition 7 *Soit B une branche ouverte. Une* trace $\mathcal{M}$ associée à B *est construite de la façon suivante : soit ι une instantiation des réseaux temporels [6]*
– $\forall t \in \mathbb{R}$, $\mathcal{K}_t = \mathtt{K}$
– $\forall t \in \mathbb{R}$, $\mathcal{LN}_t(k_1, n) = \{k_2 |$ *il existe v tel que* $t \in \iota(v)$ *et* $n : v \in \mathtt{N}(k_1, k_2)\}$
– $\forall t \in \mathbb{R}$, $\mathcal{A}_t(k_1, a) = \{k_2 |$ *il existe v tel que* $t \in \iota(v)$ *et* $a : v \in \mathtt{A}(k_1, k_2)\}$

Lemme 2 *Si B est une branche ouverte alors B est SAT.*

Preuve (idée) Soit $\mathcal{M}$ une trace associée à B par rapport à une instantiation temporelle ι. Considérons maintenant l'application μ de B vers $\mathcal{M}$ où μ est l'identité sauf pour les intervalles et les points temporels où μ coïncide avec ι. Alors en utilisant la saturation de la branche et l'absence d'incohérence nous pouvons montrer que B est SAT par induction sur les principals et les expressions. □

Théorème 2 (Complétude) *Si un certificat χ est conséquence logique d'un ensemble de certificats C alors χ a un tableau fermé, C étant donné.*

Preuve (idée) Comme pour la preuve de correction, nous montrons la contraposée. Soit B une branche ouverte d'un tableau pour χ, C étant donné. À partir de l'initialisation du tableau, du lemme 2 et de la définition 5, nous obtenons que $\neg\chi$ et $\mathcal{C}$ sont satisfaisables par une même trace. Par suite, χ n'est pas conséquence logique de l'ensemble de certificats C. □

7 Discussion et conclusions

Dans la littérature, de nombreuses propositions logiques ont été faites pour tenter de rendre compte le plus justement possible des caractéristiques de SDSI/SPKI. Ainsi Abadi [1] s'est basé sur le formalisme DEC-SRC pour le contrôle d'accès [2]. Cependant un certain nombre de problèmes a été découvert dans ce formalisme par plusieurs auteurs [23, 15]. Howell et Kotz [17] ont proposé une autre sémantique, mais d'une part leur solution est assez maladroite d'un point de vue logique (par exemple, elle n'est pas fermée pour les opérateurs booléens classiques) et d'autre part elle ne comporte pas de procédure de raisonnement. Les deux logiques modales proposées par Halpern et van der Meyden [15, 16] ont été conçues en suivant le cadre de SDSI/SPKI et, à nouveau, aucune procédure de raisonnement automatique n'a été donnée. Jha, Reps *et al.* [18, 27] ont proposé une procédure de décision basée sur les automates à piles pour, grosso modo, le fragment atemporel de la logique de Halpern et van der Meyden. Dans tous les papiers le traitement du temps est soit absent (Abadi, Howell et Kotz, Jha et Reps), soit plutôt sommaire (Halpern et van der Meyden).

Par rapport aux systèmes de Li *et al.* [21, 22], nous n'avons que des règles propositionnelles. Les constructions de Li *et al.* sont basées sur la programmation logique et datalog et permettent la quantification. Cependant l'absence de fonctions de création de termes signifie que, du point de vue du pouvoir expressif, la quantification est juste une représentation plus efficace et plus compacte de la version propositionnelle mais n'introduit pas de nouvelles possibilités. Dans la proposition de Li *et al.* la sémantique n'est pas indépendante, c'est une autre de ses limitations.

Conceptuellement, il est possible d'avoir ici aussi du raisonnement au premier ordre sur les objets et il pourrait être intéressant de dériver des restrictions syntaxiques sur la quantification dans les certificats qui permettraient d'obtenir les mêmes résultats de décidabilité que Li *et al.* [22].

La transformation des permissions peut être faite sans difficultés : il suffirait d'importer des techniques des logiques modales du premier ordre [13]. En gros, les certificats pourraient être de la forme $\mathsf{cert}\ \mathsf{Perm}(p, a(o))$ quand la permission d'exécuter l'action a est accordée à l'objet o seulement et de la forme $\mathsf{cert}\ \mathsf{Perm}(p, \lambda x.a(x))$ quand la permission est accordée à tous les objets.

La transformation des relations de dénomination serait plus complexe : il n'y a pas d'interprétation intuitive du point de vue sécurité qui soit simple pour la signification de "la clé k est associée au nom n pour l'objet o". Or pour qu'une sémantique soit convaincante une telle condition est nécessaire. Même dans des domaines proches tels que les logiques de grammaire [12] ou les logiques de description, nous ne trouvons rien qui ressemble de près ou de loin à cette construction.

Une fois ce cadre porté au premier ordre il serait intéressant d'étudier les relations entre les deux cadres. En particulier on pourrait tenter de dériver les restrictions syntaxiques sur le modèle général rendant possible l'utilisation de la représentation datalog et de son moteur d'inférence.

Nous avons fait ici un exposé général qui utilise une combinaison de techniques de l'IA pour le raisonnement sur les modalités, les identités et le temps.

Nous appuyant sur de précédentes tentatives de formalisation, nous fournissons un modèle général pour raisonner sur la dénomination et les identités, l'autorisation, les pièces justificatives et le temps. Nous montrons également comment construire une méthode de raisonnement pour la logique proposée, méthode qui combine des méthodes de tableaux pour les logiques modales et les logiques de description [8] avec des systèmes de raisonnement sur l'algèbre des intervalles d'Allen [3, 4] et des propositions avancées qui exploitent des contraintes à la fois qualitatives et métriques [24, 19].

Références

[1] M. Abadi. On SDSI's Linked Local Name Spaces. *JCS*, 6(1-2) :3–21, 1998.

[2] M. Abadi, M. Burrows, B. Lampson, and G. D. Plotkin. A Calculus for Access Control in Distributed Systems. *TOPLAS*, 15(4) :706–734, 1993.

[3] J.F. Allen. An Interval-Based Representation of Temporal Knowledge. In *IJCAI'81*, pages 221–226, 1981.

[4] J.F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11) :832–844, 1983.

[5] R. Anderson. A Security Policy Model for Clinical Information Systems. In *Proc. of IEEE SSP-96*, pages 30–43, 1996.

[6] J.-F. Condotta. The Augmented Interval and Rectangle Networks. In *KR'2000*, pages 571–579, 2000.

[7] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A Reputation-based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In *ACM CCS*, pages 207–216, 2002.

[8] G. De Giacomo and F. Massacci. Combining Deduction and Model Checking into Tableaux and Algorithms for Converse-PDL. *Inf. and Comp.*, 162(1–2) :117–137, 2000.

[9] R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49(1–3) :61–95, 1991.

[10] J. DeTreville. Binder, a Logic-Based Security Language. In *IEEE SS&P-2002*, pages 105–113, 2002.

[11] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. *SPKI Certificate Theory*, September 1999. IEFT RFC 2693. disponible à http ://www.ietf.org/rfc/rfc2693.txt.

[12] L. Fariñas del Cerro and M. Pentonnen. Grammar logics. *Logique et Analyse*, pages 123–134, 1988.

[13] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Reidel, 1983.

[14] D. Gollmann. What do We Mean by Entity Authentication ? In *IEEE SSP-96*, pages 46–54, 1996.

[15] J. Halpern and R. van der Meyden. A Logic for SDSI's Linked Local Name Spaces. *JCS*, 9(1/2) :105–142, 2001.

[16] J. Halpern and R. van der Meyden. A Logical Reconstruction of SPKI. In *IEEE CSFW'01*, pages 59–70, 2001.

[17] J. Howell and D. Kotz. A Formal Semantics for SPKI. In *ESORICS 2000*, pages 140–158, 2000.

[18] S. Jha and T. Reps. Analysis of SPKI/SDSI Certificates Using Model Checking. In *CSFW'02*, 2002.

[19] H. A. Kautz and P. B. Ladkin. Integrating Metric and Qualitative Temporal Reasoning. In *AAAI-91*, pages 241–246, 1991.

[20] D. Kozen and J. Tiuryn. Logic of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume II, chapter 14, pages 789–840. 1990.

[21] N. Li. Local Names in SPKI/SDSI. In *IEEE CSFW 2000*, pages 2–15, 2000.

[22] N. Li, B. N. Grosof, and J. Feigenbaum. A Practically Implementable and Tractable Delegation Logic. In *IEEE SSP 2000*, pages 27–42, 2000.

[23] F. Massacci. Tableaux Methods for Formal Verification of Multi-Agent Distributed Systems. *JLC*, 8(3) :373–400, 1998.

[24] I. Meiri. Combining Qualitative and Quantitative Constraints in Temporal Reasoning. In *AAAI'91*, pages 260–267, 1991.

[25] R. Rivest and B. Lampson. SDSI - A Simple Distributed Security Infrastructure. http ://theory.lcs.mit.edu/ rivest/sdsi10.html, September 1996.

[26] P. Samarati, M.K. Reiter, and S. Jajodia. An Authorization Model for a Public Key Management Service. *TISSEC*, 4(4) :453–482, 2001.

[27] S. Schwoon, S. Jha, T. Reps, and S. Stubblebine. On Generalized Authorization Problems. In *IEEE CSFW'03*, pages 202–218, 2003.

[28] P. Syverson. The Use of Logic in the Analysis of Cryptographic Protocols. In *IEEE SS& P-91*, pages 156–170, 1991.

[29] S. Weeks. Understanding Trust Management Systems. In *IEEE SS&P-2001*, pages 94–105, 2001.