UNIVERSITÉ DE BLIDA 1
Faculté des Sciences
Département d'Informatique

THÈSE DE DOCTORAT

Option : Génie des Systémes Informatique

UN FORMALISME DE SÉCURITE POUR LE CLOUD COMPUTING

par
Khalida GUESMIA

devant le jury composé de:

N. Benblidia	Professeur	U. de Blida 1	Présidente
N. Boustia	Maître de conférences A	U. de Blida 1	Directrice de thèse
N.F. Chikhi	Maître de conférences A	U. de Blida 1	Examinateur
A. Mokhtari	Professeur	U.S.T.H.B.	Examinatrice
Y. Challal	Professeur	E.S.I.	Examinateur
H. Abed	Professeur	U. de Blida 1	Invitée

Blida, Juin 2018

À ma très chère famille.

RÉSUMÉ

Le problème de confidentialité des données continue de se poser dans l'informatique en nuage. Les règles d'accès et d'utilisation des données sensibles dans ces services doivent donc être clairement définies et respectées. Et vu que l'environnement cloud est dynamique, les règles doivent être également flexibles afin de s'adapter temporairement au changement de contexte. Cette thèse se propose d'apporter sa contribution pour faire face à ce défi, en proposant un nouveau formalisme logique $TL - JClassic_{\delta\epsilon}$ qui permet d'une part, d'exprimer des règles de sécurité temporelles sous une forme prototypique admettant des exceptions, et d'autre part, il supporte le raisonnement non monotone afin de gérer implicitement les conflits entre les règles et déduire automatiquement les actions autorisées à effectuer par l'utilisateur qui demande l'accès aux données sensibles selon le contexte de la requête.
Pour montrer l'intérêt de ce formalisme, nous avons proposé deux modèles de contrôles d'accès aux données hébergés en cloud, à savoir, un modèle de contrôle d'accès dynamique basé sur le contexte temporel ($TBAC_{\delta\epsilon}$) et une extension du modèle de contrôle d'accès basé sur l'organisation ($P - OrBAC_{\delta\epsilon}$) afin d'introduire l'objectif d'usage sous forme d'une séquence d'action dans ses règles d'autorisations.

Mots clés : confidentialité des données, cloud computing, contrôle d'accès dynamique, raisonnement non monotone, logique de description temporelle.

ABSTRACT

The data privacy issue continues to arise in the services offered by cloud computing. The rules for accessing and using sensitive data in these services need to be clearly defined and respected, and since the cloud computing environment is dynamic, they must also be flexible in order to adapt temporarily with context change. This thesis aims to contribute to this challenge by proposing a new formalism, $TL-JClassic_{\delta\epsilon}$, which allows on the one hand, to specify temporal policies in a prototypical form admitting exceptions, and on the other hand, it supports non-monotonic reasoning to implicitly handle conflicts between rules and automatically deduce the actions allowed to be performed by user who requests access to sensitive data depending on the context of the access request. To show the interest of our proposed formalism, we proposed two access controls models to data outsourced into the cloud, a dynamic access control model based on the temporal context ($TBAC_{\delta\epsilon}$) and an extension of the organization-based access control model ($P-OrBAC_{\delta\epsilon}$) to introduce the purpose as a sequence of actions in its authorization rules.

Keywords : data privacy, cloud computing, dynamic access control, non-monotonic reasoning, temporal description logic.

ملخص

لاتزال خصوصية البيانات مشكلة قائمة في الخدمات التي تقدمها الحوسبة السحابية. لذلك يجب أن تكون قواعد التحكم للوصول واستخدام البيانات الحساسة الموجودة في هذه الخدمات محددة ومحترمة بشكل واضح. وبما أن بيئة الحوسبة السحابية ديناميكية ، يجب أن تكون تلك القواعد أيضًا مرنة من أجل التكيف مؤقتًا مع تغيير السياق. تهدف هذه الأطروحة إلى المساهمة في هذا التحدي من خلال اقتراح لغة شكلية جديدة التي تسمح من جهة ، للتعبير عن السياسة الأمنية على أساس السياق الزمني في شكل نموذجي يقبل بالاستثناءات ومن ناحية أخرى ، تدعم الاستدلال غير الرتيب من أجل إدارة التضارب بين القواعد بشكل ضمني و استنتاج الإجراءات المسموح بها تلقائيًا ليتم تنفيذها من قبل المستخدم الذي يطلب الوصول إلى البيانات الحساسة وفقًا لسياق الطلب.

لإظهارأهمية شكليتنا المنطقية المقترحة، اقترحنا نموذجين لضبط الوصول للبيانات المستضافة في الحوسبة السحابية ، نموذج التحكم في الوصول الديناميكي على أساس السياق الزمني وتمديد للنموذج التحكم في الوصول على أساس المنظمة لإدخال الغرض من الاستخدام كعامل أساسي في قواعد الوصول إلى البيانات الحساسة ،وذلك بتمثيله في شكل خطة عمل.

الكلمات الرئيسية: خصوصية البيانات، الحوسبة السحابية، التحكم الديناميكي في الوصول، التفكير غير الرتيب، وصف المنطق الزمني.

REMERCIEMENTS

Tout d'abord, je remercie Allah, le tout puissant, de m'avoir donné la volonté et la patience pour mener à bien ce présent travail.

Je remercie très sincèrement ma directrice de thèse, Dr.Boustia Narhimene, de m'avoir encadré durant cette thèse, pour ses précieux conseils, ses critiques constructives et ses encouragements. Qu'elle trouve ici l'expression de ma très grande gratitude.

Je tiens à remercier chaleureusement l'ensemble des membres du jury qui me font le grand honneur d'avoir accepté d'évaluer mon travail.

Finalement, j'adresse un grand merci à ma mère qui m'a soutenu et motivé durant toutes ces années, à mon père, à mes frères ainsi que le reste de ma famille pour leurs encouragements constants. Je tiens aussi à exprimer ma gratitude à mes collègues de doctorat, ainsi que tout ceux qui ont contribué de près ou de loin pour la réalisation de cette thèse.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GÉNÉRALE

Contexte de l'étude

Vu l'évolution très rapide des technologies de l'information et de la communication au cours de ces dernières années et leurs adoption par le grand public, de nombreuses entreprises, organisations et particuliers apprécient de plus en plus l'importance du modèle d'informatique en nuage, connu communément sous l'appellation cloud computing [1]. Le cloud computing propose une nouvelle approche qui consiste à fournir l'infrastructure matérielle et logicielle comme un service accessible depuis n'importe quel dispositif connecté à internet et cela sans avoir besoin d'investir au préalable dans des serveurs, des logiciels ou encore dans la maintenance. Toutes ces activités seront sous la responsabilité des fournisseurs de services cloud (service provider), qui sont les nouveaux intervenants dans l'industrie informatique. Ces fournisseurs de services cloud se chargent de garantir une disponibilité et une qualité de services et cela à travers un contrat nommé SLA (Service Level Agreement) conclu avec le consommateur du service. En outre, les services cloud sont évolutifs et peuvent être adaptés à tout moment aux besoins des clients, notamment pour supporter les montées de charges importantes et cela avec un minimum de temps et sans coûts fixes, le paiement se fait uniquement en fonction de la consommation.

Le cloud computing offre de nombreux avantages par rapport aux serveurs et aux applications déployés et gérés en interne mais soulève également un certain nombre de questions relatives à la protection des données et aussi à la disponibilité et la continuité des services. En adoptant le cloud, les entreprises et les particuliers ne seront plus propriétaires de leurs ressources informatiques, les applications et les données se retrouvent chez les fournisseurs dans les centres de données qui regroupent un nombre important des serveurs qui sont reliées au réseau internet grâce à des connexions à très haut débit et elles sont distribuées souvent dans des zones géographiques différentes afin de garantir une meilleure disponibilité de services.

A cet effet, la sécurité des données est l'un des objectifs prioritaires à atteindre dans le

cloud, de meilleures mesures de sécurité ont déjà été mises en place par les fournisseurs de cloud [2]. D'une part, l'accès physique aux centres de données est extrêmement sécurisé. Des systèmes complexes de climatisation, d'alimentation et de protection contre les incendies ont été conçus et mis en œuvre pour assurer le fonctionnement continu de leurs équipements informatiques même en cas de défaillance et de panne. D'autre part, les fournisseurs de cloud emploient d'importantes équipes d'experts pour contrôler régulièrement la sécurité de leurs plateformes. En plus, ils répliquent automatiquement les données à plusieurs endroits différents ce qui assure qu'en cas de défaillance matérielle, les données pourront facilement être récupérées.

Malgré tous les efforts fournis de la part des fournisseurs du cloud notamment en termes de sécurité, et selon un certain nombre d'études qui ont été réalisés au cours des dernières années, les entreprises et les particuliers hésitent à migrer vers le cloud, ils font plus confiance aux systèmes existants géré en interne que les systèmes basés sur le cloud.

Problématique

Pour les entreprises comme pour les particuliers, lorsqu'ils externalisent leurs données aux fournisseurs de services cloud, c'est avec l'espoir que leurs données ne seront utilisées que dans le cadre prévu surtout pour les données à caractère personnel, mais dans l'environnement cloud, il y a également la possibilité que les fournisseurs ou leurs employés utilisent ces données pour leurs propres objectifs. Donc, si les clients n'ont pas confiance en le fournisseur de cloud, ils n'auront pas recours à ses services. Afin d'encourager les entreprises et les particuliers à migrer vers le cloud, les fournisseurs de services cloud doivent s'engager à mettre en œuvre des mesures de sécurité appropriée afin de préserver le caractère confidentiel des données sensibles hébergées dans leurs plateformes. A cet égard, il faut adopter des solutions techniques comme le hachage et le chiffrement des données, et des solutions organisationnelles comme les politiques de contrôle d'accès et les anonymisations des données. Dans ce travail, nous nous intéressons particulièrement aux politiques de contrôle d'accès logique.

Une politique de contrôle d'accès logique est une forme de politique de sécurité spécialisée pour la gestion des permissions d'accès aux données dans un système afin de garantir leurs confidentialité et leurs intégrités, elle permet donc de limiter l'accès aux données aux seuls utilisateurs autorisés [3]. Dans la littérature, il existe divers modèles de contrôle d'accès qui ont été proposés (DAC, MAC, RBAC, OrBAC, ...). Un modèle de

contrôle d'accès sert à élaborer une représentation formelle afin de décrire de manière non ambiguë les différentes entités qui constituent le cadre d'expression d'une politique de contrôle d'accès dans un système. Les modèles de contrôle d'accès existant sont enrichis et ils ont évolué parallèlement avec le développement des technologies. Cependant, l'environnement cloud computing se caractérise par des changements de contexte ou d'environnement qui se produisent fréquemment, et qui sont ou non prévues au préalable. Les règles d'autorisation dans ce type des services sont généralement sujettes à une mise à jour continue en réponse aux changements dans les exigences. Les modèles proposés dans la littérature ne sont pas assez flexible pour spécifier des règles de sécurité contextuelle et qui permettent d'attribuer des droits d'accès de manière dynamique aux entités autorisées, ce qui est nécessaire pour protéger les données à caractère personnelles stockées dans de tels environnements. De plus, la mise en œuvre des politiques qui incluent à la fois des règles positives et négatives peuvent conduire éventuellement à des conflits potentiels dans la prise de décisions d'accès, et cela peut affecter négativement les performances du système. Le problème se situe au niveau du formalisme logique utilisé pour spécifier les règles de sécurité, il est souvent inadapté pour représenter les connaissances contextuelle et il est basé sur le raisonnement monotone qui manque de flexibilité et d'adaptabilité au changement de contexte. En effet, le raisonnement monotone signifie que si les connaissances augmentent (resp. diminuent) alors les déductions augmentent (resp. diminuent) et qu'une conclusion obtenue n'est jamais remise en cause alors que dans le raisonnement de sens commun ou non monotone, les conclusions déduites peuvent être révisables à la lumière de nouvelles informations. Par conséquent, pour spécifier un modèle de contrôle d'accès dynamique adéquat pour l'environnent cloud, il faut avoir un nouveau formalisme expressif qui permet d'exprimer des règles de sécurité contextuelles et flexibles sous une forme prototypique admettant des exceptions et permet de gérer implicitement les conflits lors de déduction d'une décision d'accès.

Contributions

Dans le cadre de notre étude, nous nous somme intéressés aux données partagées dans les réseaux sociaux et les données sensibles utilisées dans les entreprises et les organisations qui adoptent le modèle d'informatique en nuage dans leurs systèmes. Le formalisme logique utilisé pour mieux représenter les connaissances est celui de la logique de description. Cette logique a pour avantage de disposer d'un langage de

représentation expressif et d'avoir des algorithmes de raisonnement décidables qui se terminent toujours avec les bonnes réponses. Boustia et all. [4] ont déjà proposé et développé un raisonneur pour la logique de description non-monotone $JClassic_{\delta\epsilon}$ qui permet de représenter des connaissances strictes, défauts ou exceptionnelles et de pouvoir réaliser des raisonnements à partir de ces informations avec une complexité polynomiale. Néanmoins, elle ne permet pas de représenter les connaissances temporelles, ces dernières peuvent être parfois d'une grande utilité pour exprimer des règles de sécurité contextuelle dans l'environnement cloud. La représentation de la connaissance temporelle a reçu une attention considérable dans le domaine de l'intelligence artificielle. La logique temporelle [5] permet de définir un plan d'action ou d'événement comme se produisant sur des intervalles temporels, et elle permet d'exprimer des contraintes temporelles sur ces intervalles en utilisant l'algèbre d'Allen [6], qui est un formalisme bien connu utilisé pour représenter et raisonner sur les connaissances temporelles qualitatives. Pour cela, nous proposons d'augmenter l'expressivité du $JClassic_{\delta\epsilon}$ en ajoutant des connecteurs de la logique temporelle tout en maintenant une complexité de calcul polynomiale afin de l'utiliser en pratique comme un formalisme de sécurité pour assurer la confidentialité des données sensible sauvegardées dans le cloud [7] .

Les réseaux sociaux sont parmi les services cloud les plus utilisés, aujourd'hui les utilisateurs passent plus de temps sur les réseaux sociaux à partager des informations sur leurs vie personnelle, leurs activités, leurs opinions, leurs envies, leurs centres d'intérêt. . . etc. et cela sous différents formes (texte, photo, vidéo). Dans ce genre du système, l'utilisateur est sensé définir les droits d'accès pour les données qu'il partage en utilisant les paramètres de confidentialité fournis par le fournisseur du système. Cependant, plusieurs études montrent que de nombreux utilisateurs ont des difficultés à configurer correctement leurs paramètres de confidentialité [8, 9]. L'une des causes principales du problème est liée à l'expressivité limitée des mécanismes de contrôle d'accès offert par le système. En effet, avec le temps, les utilisateurs se retrouvent avec une énorme quantité d'informations sensibles qu'ils ont été déjà postées sur leurs comptes et ces informations restent toujours disponibles en ligne sauf s'ils ont été supprimés par leurs propriétaires [10]. Dans ce cas, les utilisateurs ont besoin de spécifier sous quel contexte temporel leurs données partagées doivent être accédées. En utilisant notre logique de description temporel non monotone $TL - JClassic_{\delta\epsilon}$, nous proposons un modèle de contrôle d'accès dynamique basé sur le contexte temporel, qui permet aux

utilisateurs d'introduire des contraintes temporelle générales sujette à l'exception afin de limiter l'accès à leurs données partagées dans les réseaux sociaux et par conséquent, la décision d'accès dans ce modèle sera déduite de façon intelligente en fonction du contexte actuel de la requête d'accès.

La grande majorité des entreprises et organisations stockent et traitent des données sensibles et confidentielles des utilisateurs. La plupart des pays ont déjà pris des mesures législatives pour protéger les données à caractère personnel en vue d'assurer la sécurité de leurs citoyens et de garantir le droit au respect de leurs vie privée. Pour cela, les entreprises et les organisations qui gèrent ces données doivent les protéger face aux risques et aux menaces qui pèsent sur elles. Le modèle de contrôle d'accès basé sur l'objectif d'usage est introduit récemment pour protéger l'accès et l'utilisation des données privées qui sont hors contrôle de leur propriétaire, il permet de garantir que ces données ne peuvent être utilisées que pour l'objectif auquel elles sont destinées. La mise en œuvre d'une telle politique dépend totalement de comment déterminer l'objectif de l'usage lorsqu'un utilisateur demande l'accès aux données, ce qui est considéré comme un des principaux défis de recherche dans le domaine de la protection des données.

De nombreuses solutions ont été proposées dans la littérature. Plusieurs travaux supposent que l'utilisateur lui-même déclare de manière explicite son objectif d'accès [11]. Cette approche comporte le risque de contournement par des utilisateurs malintentionnés afin d'avoir l'accès. D'autres chercheurs ont suggéré que l'objectif de l'usage est déduit en fonction du rôle de l'utilisateur [12, 13]. Une telle solution ne permet pas d'identifier avec certitude l'objectif réel de l'accès. Tschantz et all. [14] proposent que l'objectif de l'usage est déterminé dynamiquement par le système, en se basant sur la situation actuelle de l'action demandé, i.e., en fonction des actions qui précèdent ou suivent cet accès. Donc, il est important de représenter l'objectif de l'usage des données sous forme d'un workflow ou d'un plan d'action car il permet au système d'assurer que l'utilisateur qui veut accéder aux données sensibles, n'effectuera que les actions nécessaires pour réaliser l'objectif qu'il veut atteindre. Vu que les entreprises et les organisations commencent déjà à fournir leurs services sous forme des services cloud, il est intéressant d'adapter ce modèle au contexte dynamique de l'environnement cloud computing. Pour cela, nous proposons d'introduire l'objectif de l'usage dans le modèle OrBAC, vu que ce dernier facilite l'expression des règles de sécurité par rapport aux

autres modèles notamment le modèle RBAC. Il permet de définir une politique de sécurité totalement indépendante de son implémentation. De plus, il prend en considération le contexte dans les règles d'autorisation.

Dans ce nouveau modèle, les règles d'autorisation seront définies comme un ensemble ordonné de permissions élémentaires, et la requête d'accès au système sera différente, elle devrait inclure une demande d'exécuter une séquence d'action. A partir de cette séquence, le système déduira l'objectif de l'accès de l'utilisateur et déterminera ensuite si la séquence d'action est autorisée ou non. Pour rendre notre modèle adéquat et suffisamment flexible pour assurer la confidentialité des données sensible sauvegardés dans le cloud sous différents contextes, il est nécessaire de représenter les règles d'autorisations basé sur l'objectif de l'usage sous une forme prototypique admettant des exceptions. En général, lorsqu'on définit un processus ou un plan d'action, toutes ses actions sont implicitement obligatoires. On propose de faire une distinction entre une action obligatoire et une action facultative. On entend par une action facultative, l'action qui peut être annulée ou ignorée dans des cas particuliers afin de s'adapter temporairement aux changements de contexte, sans que cela n'affecte la définition du plan. Notre formalisme proposé, nous permet de décrire formellement une telle politique de sécurité, et grâce à ses services d'inférences, il est possible de déduire automatiquement les actions autorisées à effectuer par les utilisateurs qui demandent l'accès aux données sensibles sauvegardées dans le cloud.

Plan du manuscrit

Ce manuscrit est structuré en cinq chapitres répartis en deux parties. La première partie consiste à étudier à travers la littérature le paradigme du cloud computing, les différents modèles de contrôle d'accès existants et les travaux d'extension de la logique de description par la dimension temporelle et cela pour fournir la base terminologique nécessaire pour la bonne compréhension de nos travaux. La deuxième partie sera consacrée à la présentation de nos contributions.

Chapitre 1 donne un aperçu sur le fonctionnement général du cloud computing, ses caractéristiques essentielles, ces principaux services et ces différents modèles de déploiement. On terminera par une analyse de l'aspect de la sécurité des données dans le cloud qui est cité souvent comme le frein principale qui bloque son adoption.

Chapitre 2 présente les généralités sur les différentes notions relatives aux modèles

de contrôle d'accès. Une synthèse des principaux modèles de contrôle d'accès existant y est décrite afin de choisir la façon la plus adéquate pour exprimer une politique de sécurité pour les données sauvegardées en cloud computing. Dans ce chapitre, on met en évidence la nécessité d'avoir un langage formel qui permet de représenter les connaissances temporelles et supporte le raisonnement non monotone afin de formaliser, classer les règles d'autorisation, évaluer les différentes requêtes d'accès et enfin déduire les autorisations appropriées selon le contexte.

Le chapitre 3 présente quelques travaux d'extension de la logique de description par la dimension temporelle. Ce chapitre commence par introduire les principes fondamentaux de la logique de description, leur formalise de base et les services d'inférence standards. Deux types d'algorithmes de raisonnement existant sont décrits, à savoir les algorithmes structurels et les algorithmes des tableaux. On introduit par la suite, les logiques temporelles et les différentes approches pour étendre la logique de description par l'aspect temporel.

Chapitre 4 détaille notre nouveau formalisme logique $TL - JClassic_{\delta\epsilon}$. La syntaxe, la sémantique et les différents services d'inférences associées à ce nouveau formalisme seront exposés en détails dans ce chapitre suivi de l'étude de la complexité des algorithmes d'inférences.

Chapitre 5 est consacré aux modèles de contrôle d'accès proposés afin de garantir la confidentialité des données privées sauvegardées dans le cloud. En premier lieu, on présente un modèle dédié à la protection des données personnelles dans les réseaux sociaux en introduisant la notion du contexte temporel. En second lieu, on introduit les entités principales du modèle de contrôle d'accès à base de l'objectif d'usage sous forme d'un plan. On décrit les différentes règles de sécurité qui nous permettent de déduire de manière dynamique la séquence d'actions appropriée qui peut être effectuées par l'utilisateur qui demande l'accès aux données sensibles. Un exemple d'illustration est présenté par la suite ainsi que les tests d'évaluation de performances.

Le manuscrit se termine par une conclusion générale qui synthétise nos contributions et les principaux choix que nous avons faits. Quelques perspectives futures concluront ce chapitre.

CHAPITRE 1
LA SÉCURITÉ DES DONNÉES DANS LE CLOUD COMPUTING

1.1 Introduction

Le cloud computing a réussi à concrétiser l'idée d'abstraire l'espace physique dans lequel les calculs et les données sont gérés. Aujourd'hui avec le cloud computing, il est possible de consommer des ressources informatiques en tant que services, sans besoin d'avoir la moindre idée de ce qui se passe dans l'infrastructure sous-jacente à ces services. Le mot nuage (cloud) a tout d'abord été utilisé pour représenter de manière simple et abstraite l'interconnexion de réseaux informatique notamment l'internet dans les schémas. En effet, l'internet est un réseau informatique étendu reliant des terminaux (un router, un serveur, une station de travail,..) du monde entier entre eux. Lorsqu'on utilise le réseau internet pour accéder à une application web, nous ne savons pas exactement le chemin emprunté pour relier l'appareil sur lequel on s'est connecté et les serveurs hébergeant l'application [15], donc on simplifie par schématiser le réseau internet sous forme d'un nuage. Le mot nuage a été repris plus tard pour masquer la complexité des divers éléments composant le nouveau paradigme de l'informatique (la virtualisation, la mutualisation des ressources, les mécanismes d'élasticités et d'automatisation).
Dans ce chapitre, nous allons définir clairement le concept cloud computing en exposant d'abord quelques définitions, puis décrire ses caractéristiques, parcourir les modèles de service et les modèles de déploiement spécifiques à ce paradigme. Ensuite, nous examinons les principaux enjeux liés à la sécurité des données dans le cloud computing.

1.2 Émergence du cloud computing

Les chiffres relatifs à l'usage du web sont toujours en croissance nette et cela grâce aux géants du web comme Google, Apple, Facebook, Amazon et d'autres qui ont contribué à révolutionner l'usage du web par la mise en place des services de qualité en ligne comme le moteur de recherche, le commerce électronique, le courrier électronique, le réseau social, etc. Aujourd'hui, ces géants gèrent des millions d'utilisateurs à travers

le monde avec un volume de données gigantesques (Google par exemple stocke un volume considérable de vidéos avec Youtube, d'emails avec Gmail). Ces acteurs sont toujours confrontés à gérer la croissance de leurs utilisateurs potentiels qui ne cesse d'augmenter, ils sont confrontés à un problème de passage à l'échelle. Pour atteindre une performance élevée en terme de disponibilité et d'accessibilité de leurs services, ces acteurs ont été obligés d'améliorer et perfectionner leurs ressources informatiques, ils ont développé des environnements logiciels (frameworks) mais aussi physiques via la construction de centres de calcul spécifiques, ce que nous appelons les centres de données, qui sont dotées de plus d'un million de serveurs, notamment pour pouvoir absorber les pics de charge imprévisibles. En effet, la possession d'un million de serveurs pour l'absorption des pics était un premier pas vers le cloud, la mise en œuvre d'un si grand nombre de machines représente un défi technique mais surtout économique. Comment faire face aux coûts engendrés par ces millions de serveurs qui ne fonctionnent pas tout le temps ? Comment gérer le coût des réseaux ? Comment gérer le coût de l'énergie ? Comment gérer le coût du personnel ? etc. [16]. Depuis, les acteurs du web, chacun à sa manière a tenté de permettre à des sociétés tierces d'exploiter leurs ressources informatiques inutilisées afin de les rentabiliser en hébergeant leurs applications.

En 2006, Amazon, le leader mondial du commerce électronique, a proposé aux entreprises et aux particuliers les premiers services d'infrastructure cloud. En effet, en constatant que les pics d'utilisation de leurs ressources informatiques étaient saisonniers (fêtes de fin d'années, soldes, etc.), ils ont réussi a concevoir des technologies permettant d'offrir des services accessibles via internet et avec une adaptation en temps réel de la capacité de traitement, le tout facturé à la consommation. Ce qui est nouveau avec cette approche est que les serveurs ne sont pas dédiés spécialement aux consommateurs mais plutôt une multitude de serveurs virtuels qui peuvent fonctionner sur n'importe quel serveur en fonction de la disponibilité. De cette manière, les ressources inutilisées peuvent être momentanément vendues comme un service et constituer des sources de revenus. Nous en concluons donc que les immenses volumétries d'utilisateurs et de données que manipulent les géants du web ont conduit à l'émergence du cloud computing [17].

1.3 Définition du cloud computing

La définition du concept cloud computing a été largement discutée par la communauté scientifique sans parvenir à s'entendre sur une définition unanimement acceptée. Pour cela, on trouve plusieurs définitions de cloud dans la littérature, le Tableau 1.1 [18] présente les principales définitions du cloud computing. En effet, il existe une énorme divergence de vues portant sur ce qu'est le cloud computing. Certains chercheurs pensent que l'accent devrait être mis sur le passage à l'échelle et la mutualisation de l'usage des ressources informatiques [19] alors que d'autres privilégient le principe de virtualisation et introduisent dans leur définition les notions de l'accord de prestation de service (SLA) et la qualité de service (QoS)[20, 21]. Cependant, la définition proposée par L'Institut National des Standards et de la Technologie (NIST) [1] est l'une des définitions les plus connues et qui est souvent utilisé comme référence dans plusieurs travaux.

1.4 Caractéristiques essentielles du cloud computing

Il y a cinq caractéristiques essentielles de services de cloud qui permettent de définir en quoi consiste un service cloud et permettent de le différencier d'un système conventionnel [24, 2, 1].

1.4.1 Libre-service à la demande

Cette caractéristique du cloud marque un grand changement par rapport aux modèles précédents. Les clients peuvent disposer des ressources informatiques sans forcément passer par un vendeur, un simple clic à travers une interface déclenche la mise à disposition de la demande (serveurs, applications, etc.). La caractéristique "à la demande" signifie que les clients peuvent modifier leurs demandes à la hausse comme à la baisse, en fonction de leurs besoins. Il n'est alors plus nécessaire de posséder des infrastructures ou des ressources informatiques non utilisées.

1.4.2 Accès global à travers le réseau

Grâce à des mécanismes standardisés, tels que les APIs (Application Programming Interface), il est possible d'accéder aux ressources informatiques à partir de tout appareil (smartphone, PC, tablette, terminal, etc.) ayant la capacité de se connecter à internet, de n'importe où et à n'importe quel moment.

Tableau 1.1 – Les différents définitions de cloud computing

Références	Définition en anglais	Interprétation
R. Buyya [22]	A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service level agreements established through negotiation between the service provider and consumers.	L'informatique en nuage est une solution de virtualisation de machines interconnectées et approvisionnées à la demande comme un ensemble de ressources unifiées en se basant sur le SLA établi entre le fournisseur et le consommateur
J. Geelan [19]	Cloud computing is one of those catch all buzz words that tries to encompass a variety of aspects ranging from deployment,load balancing, provisioning, business model and architecture (like Web2.0). It's then logical step in software (software 10.0). For me the simplest explanation for Cloud Computing is describing it as, "internet centric software"	L'informatique en nuage est une solution d'approvisionnement de ressources, de déploiement, d'équilibreur de charge, de modèle économique et d'architecture Web 2.0
K. Sheynkman	Clouds focused on making the hardware layer consumable as on-demand, compute and storage capacity. This is an important first step, but for companies to harness the power of the Cloud, complete application infrastructure needs to be easily configured, deployed, dynamically-scaled and managed in these virtualized hardware environnements.	L'informatique en nuage est une solution de virtualisation à la demande permettant à tout type d'entreprise d'approvisionner, de déployer et de mettre à l'échelle de manière dynamique son application sans se soucier des configurations des infrastructuctures sous-jacentes
L. Vaquero [23]	Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services).	L'informatique en nuage est une solution de virtualisation de ressources matérielles et logicielles accessibles et utilisables facilement.
W. Lizhe [21]	A computing Cloud is a set of network enabled services, providing scalable, QoS guaranteed, normally personalized, inexpensive computing infrastructures on demand, which could be accessed in a simple and pervasive way	L'informatique en nuage est un ensemble de services accessibles de partout, qui fournit le passage à l'échelle, garantit la qualité de service. Elle est basée sur un modèle économique attractif.
NIST [1]	Cloud computing is a model for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources(e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.	L'informatique en nuage est un modèle ubiquitaire accessible à la demande qui fournit un pool de ressources configurables qui peuvent être approvisionnées et libérées avec un minimum d'efforts de gestion ou d'interaction avec le fournisseur.

1.4.3 Mutualisation des ressources

Le cloud computing repose sur un modèle de ressources mutualisées. Cette caractéristique est rendue possible par la technologie de virtualisation. La virtualisation couvre l'ensemble des techniques permettant de reproduire le comportement d'une machine physique (Physical Machine PM) dans un environnement logiciel appelé machine virtuelle (Virtual Machine VM). Ainsi, l'utilisateur a l'illusion de manipuler une PM alors qu'il s'agit en réalité d'un environnement logiciel. Tout comme une PM, une VM contient un processeur, une mémoire RAM ainsi qu'un disque dur et une carte d'interface réseau qui lui sont propres bien que virtuels [25]. Chaque VM exécute son propre système d'exploitation (Operating System OS) ce qui permet à l'utilisateur d'héberger des logiciels. On parle alors d'OS invité pour le distinguer de celui de la PM qualifié d'OS hôte. Les VMs sont créées et gérées par des logiciels appelés hyperviseurs. L'hyperviseur gère les accès aux ressources physiques de PM, ainsi que leurs partages entre les machines virtuelles en concurrence. La Figure 1.1 présente une architecture simplifiée permettant la création des VMs dans une PM.

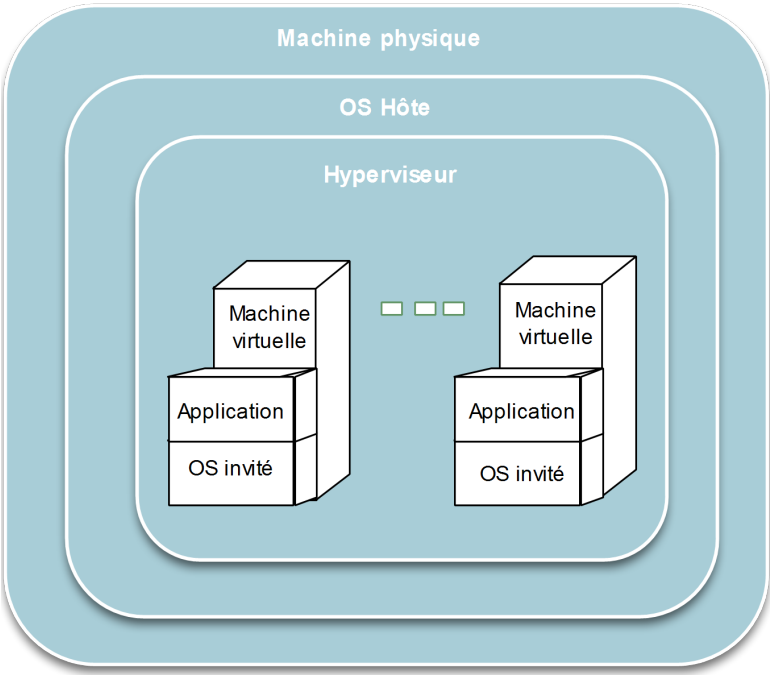

Figure 1.1 – Exemple d'architecture de virtualization

Grace à la virtualisation, on peut donc faire fonctionner sur un même serveur plusieurs applications qui appartiennent à une ou plusieurs entreprises (mutualisation). Ce

partage des ressources est la caractéristique qui différencie le cloud computing de l'infogérance. La mutualisation du matériel permet d'optimiser l'utilisation des machines physique et de ce fait elle permet d'optimiser les coûts par rapport aux systèmes conventionnels et de développer des applications partagées sans avoir besoin de posséder ses propres machines dédiées au calcul. C'est également grâce à cette caractéristique que l'on peut faire l'abstraction de la localisation des ressources informatiques, donc, la machine virtuelle ou l'application de l'utilisateur est quelque part dans les centres de données de fournisseur.

1.4.4 Élasticité

Le critère d'élasticité du cloud computing est lié à la caractéristique «à la demande». Les ressources informatiques peuvent être approvisionnées et libérées avec un minimum de temps et cela pour supporter des montées en charge importantes. La capacité du système paraît donc illimitée pour les utilisateurs, ils peuvent accéder aux ressources demandées à n'importe quel moment. Avant le cloud computing, les entreprises avaient besoin de plusieurs mois entre le passage de la commande et la mise à disposition des ressources informatiques. Le recours au cloud permet d'avoir à leurs dispositions les ressources nécessaires dans un délai court. Même si certains fournisseurs prennent un peu plus de temps que d'autres parce que les technologies ne sont pas les mêmes, mais le temps d'attente avant cette mise à disposition reste considérablement diminué.

1.4.5 Facturation en fonction de l'utilisation

Le cloud utilise un modèle de paiement de type payez ce que vous utilisez, au lieu d'un paiement forfaitaire où peu importe l'ampleur de la consommation, le cloud permet un paiement variable selon l'utilisation des capacités. Par exemple, une entreprise qui a besoin d'un millier de serveurs pour un test de deux heures peut avoir recours à un fournisseur de services cloud. Elle ne paiera que le service rendu pendant les deux heures, et elle n'aura pas besoin d'acheter les milliers de serveurs. L'utilisation des ressources est quantifiée par des moyens de mesure à certains niveaux d'abstraction appropriés au type de service (par exemple, les heures pour le CPU). Néanmoins, les entreprises peuvent fixer un taux d'usage ou un montant maximum à ne pas dépasser. Donc l'utilisation des ressources dans le cloud est surveillée, contrôlée, et rapportée, afin d'offrir

de la transparence pour le fournisseur et le consommateur du service utilisé. Cependant il existe des services cloud qui sont gratuits comme les courriers électroniques, les réseaux sociaux, etc. Ces services gratuits ne peuvent se financer que par les publicités ou par l'utilisation du profil de leurs utilisateurs comme un capital commercialisable.

1.5 Les principaux modèles de services du cloud computing

Des solutions en mode (*X as a service*) sont de plus en plus nombreuses, X définit le type de service offert. Il peut s'agir, entre autres, de stockage de données, de capacité de calcul, d'applications,... etc. Ces services de cloud peuvent être classés en trois catégories fondamentales selon la nature des ressources informatiques fournies, ces classifications sont souvent appelées modèle SPI, où SPI se réfère respectivement au software (logiciel), plateforme et à l'infrastructure en tant que service [26].
Dans cette section, on expose ces trois principaux modèles de services de clouds et on montre les parties gérées par le fournisseur de service cloud et de celles gérées par l'utilisateur (le consommateur de service). Il faut noter que, la gestion et la maintenance de l'infrastructure physique dans les trois modèles de services de cloud est gérée par le fournisseur de service, comme montre la Figure 1.2 [27], en comparent avec le cas classique, où l'utilisateur gère lui-même tout l'ensemble des éléments (matériel, réseau ... etc.).

1.5.1 Logiciel en tant que service (SaaS)

La capacité fournie aux clients dans ce modèle est d'utiliser les applications du fournisseur qui s'exécutent sur leur plateforme cloud. En effet, dans le modèle traditionnel de fourniture de logiciels, il fallait installer le logiciel sur les postes de travail des utilisateurs, effectuer les mises à jour en permanence et acheter les licences. Cependant, le logiciel de type SaaS est prêt à être utilisé, il n'existe pratiquement aucune contrainte technique, l'application est accessible via un navigateur web ou une interface de programme. L'utilisateur pourra éventuellement configurer certains paramètres de l'application, mais il ne peut pas gérer l'infrastructure ni la plateforme sous-jacente sur laquelle s'exécute l'application. Exemple : les réseaux sociaux.

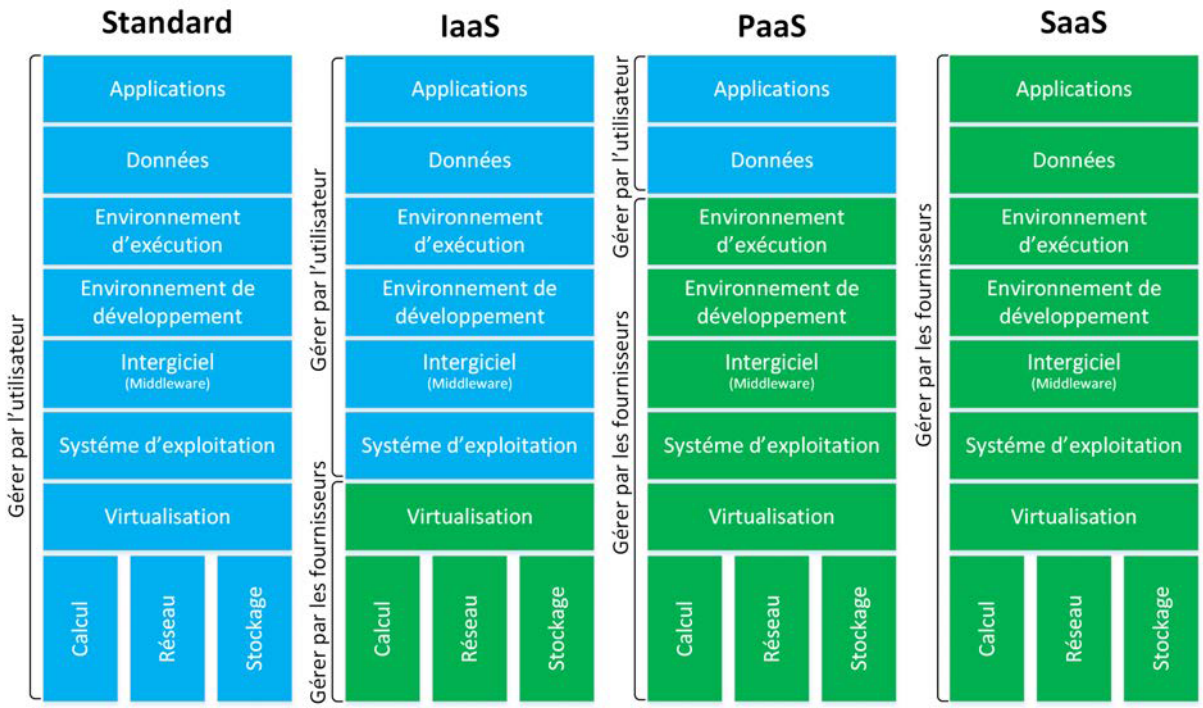

Figure 1.2 – Les différents niveaux de contrôle du fournisseur et d'utilisateur dans les modèles de services cloud

1.5.2 Plateforme en tant que service (PaaS)

Dans ce modèle, le fournisseur offre aux développeurs un environnement prêt à l'emploi pour développer des applications et les tester, ainsi que pour les déployer. Le fournisseur de service prend en charge l'administration et la maintenance de l'infrastructure informatique et les logiciels de base telle que le système d'exploitation et des logiciels qui permettent l'existence de la plateforme et ses composants, tels que les logiciels d'exécution, les bases de données, et d'autres composants middleware. Les utilisateurs ont le contrôle sur les applications qui sont déployées et ils peuvent modifier leurs configurations. L'accès à la plateforme de développement se fait aussi par un simple navigateur, sans installation spécifique. La différence entre le SaaS et le PaaS est que le SaaS inclut des applications achevées tandis que le PaaS offre une plateforme où sont hébergées des applications en cours de développement. Le coût et la complexité du développement et de déploiement des applications peuvent être largement réduits lorsque les développeurs utilisent ce modèle de service cloud [28]. Exemple : AppEngine de Google et Azure de Microsoft.

1.5.3 Infrastructure en tant que service (IaaS)

Dans ce modèle, les utilisateurs ont à leur disposition une infrastructure telle que les serveurs, les réseaux, et le stockage sous forme de services accessibles en utilisant un navigateur. Ces offres s'adressent aux services informatiques puisque l'utilisation des services d'infrastructure cloud nécessite des compétences en architecture ou en administration des systèmes d'information. Pour se faire, le fournisseur s'occupe de toute la partie infrastructure physique, du lieu d'hébergement des infrastructures, pour en faire un centre de données. Les fournisseurs se chargent également des logiciels nécessaires pour donner à ces infrastructures physiques les propriétés des infrastructures cloud alors que le consommateur possède un contrôle sur les systèmes d'exploitation, le stockage, les applications déployées et, éventuellement, le contrôle limité de certains composants réseau. Exemple : Amazon EC2 (Amazon Elastic Compute Cloud)

1.6 Les modèles de déploiement

Les modèles de déploiement indiquent où se trouve l'infrastructure cloud et comment elle est gérée pour la livraison des services décrits précédemment. Il existe quatre modèles de déploiement du cloud, chacun avec ses avantages et ses inconvénients.

1.6.1 Cloud privé

L'infrastructure matérielle de ce type de cloud est dédiée à l'usage exclusif (monolocataire) d'une seule organisation ou entreprise et elle est accessible par des liaisons louées privées ou des connexions sécurisées via les réseaux publics, ceci dans le but de répondre à des contraintes de sécurité ou de confidentialité particulières. Cependant, le cloud privé peut être interne ou externe. Le cloud privé est interne lorsque les infrastructures sont hébergées et géré par l'entreprise elle-même. Dans ce cas, le service informatique de l'entreprise garde la maîtrise complète des ressources informatiques. Le cloud privé est externe lorsque les ressources informatiques (tout ou partie) sont externalisées tout en étant dédiées à une seule entreprise. En effet, le cloud privé peut s'avérer coûteux à mettre en place, en plus le cloud privé n'a que très peu d'intérêt, il ne bénéficie pas des économies d'échelles permises par la mutualisation des ressources, qui est parmi les caractéristiques essentielles des services cloud. Pour cette raison, il y en a qu'ils ne le considèrent pas comme un service cloud mais une utilisation optimisé des ressources informatiques. Néanmoins, la sécurité supplémentaire que fournit

le modèle de cloud privé est idéale pour tout type d'entreprise ayant besoin de stocker et de traiter des données privées, ou alors d'exécuter des tâches sensibles [29].

1.6.2 Cloud public

L'infrastructure physique de ce type de cloud est possédée et contrôlée par une entreprise commerciale, un organisme de recherche ou encore une organisation gouvernementale. Le principe du cloud public peut être défini par opposition à celui du cloud privé. Le cloud public fournit des services virtuels au grand public à travers une plateforme multi-locataire. Le cloud public est plus avantageux au niveau des coûts que le clouds privé mais généralement, les entreprises ne veulent pas déplacer leurs données et leurs applications critiques dans le cloud public pour des raisons de sécurité et de contrôle [29, 24].

1.6.3 Cloud communautaire

L'infrastructure cloud est provisionnée pour une utilisation exclusive par plusieurs organisations d'une même communauté qui ont des préoccupations et intérêts communs. Il peut être détenu, géré et exploité par un ou plusieurs organismes de la communauté, ou alors externalisées, gérées par un tiers ou une combinaison des deux.

1.6.4 Cloud hybride

Le cloud hybride consiste à combiner plusieurs clouds (privé, public ou communautaire) qui restent des entités à part entière mais qui sont liées par une technologie standardisée ou propriétaire qui permet d'assurer la portabilité des données et des applications. Les entreprises peuvent déporter des applications vers un cloud public qui consommera des données stockées et exposées dans un cloud privé, ou bien faire communiquer deux applications hébergées dans deux cloud privés distincts, ou encore consommer plusieurs services hébergés dans des cloud publics différents. Cependant, les clouds hybrides introduisent une complexité supplémentaire à savoir déterminer la façon de distribuer les applications à travers les différents clouds.

A partir de ces modèles de déploiement présentées précédemment, on peut constater que les modèles se diffèrent selon les besoins d'utilisation, le client peut obtenir un meilleur contrôle sur l'infrastructure cloud, lorsqu'il passe d'un cloud public à un cloud communautaire et plus encore d'un cloud communautaire à un cloud privé.

1.7 La sécurité dans le cloud computing

Comme nous l'avons vu dans les sections précédentes, le cloud offre de nombreux avantages par rapport aux systèmes gérés en interne. Cependant, les entreprises hésitent à migrer vers le cloud. Selon une étude récente [30] réalisée par le CSA (Cloud Security Alliance) [1] sur les obstacles et les défis actuels qui empêchent les différentes entreprises à l'adoption du cloud, 73% des participants à cette étude ont déclaré qu'ils s'inquiètent pour la sécurité de leurs données. La sécurité des données dans le cloud est comme dans n'importe quel système informatique vise à garantir la confidentialité, l'intégrité, et la disponibilité des données. La sécurité peut également inclure l'authentification, et la non-répudiation.

- **La disponibilité :** il s'agit de garantir que les données et les services du système sont disponibles à tout moment aux utilisateurs autorisés. Il faut pour cela s'assurer que le système fonctionne correctement et empêcher un utilisateur malveillant d'arrêter ou de bloquer un service (Denial of Service attack).
- **L'intégrité :** il s'agit de la protection des données contre la destruction, l'altération et la modification de façon non autorisée. Pour assurer l'intégrité des données en transit par exemple, on peut utiliser les protocoles standards qui permettent le transfert des données entre le client et le fournisseur du cloud.
- **La confidentialité :** il s'agit de garantir qu'aucune donnée dite sensible ne peut être divulguée à autrui, à un service ou un matériel non autorisé i,e. les données sont uniquement accessible par ceux qui sont autorisés et tout accès indésirable doit être nié ou refusé. La confidentialité doit être assurée techniquement par des mécanismes de chiffrement et de contrôle d'accès.
- **L'authenticité :** garantit qu'un sujet (utilisateur, processus, système, etc.) est celui qu'il prétend être et que les informations reçues de ce sujet sont identiques à celles fournies, l'identification des utilisateurs est cruciale pour gérer les accès pertinents et maintenir la confiance dans les relations d'échange.
- **La non-répudiation et l'imputation :** aucun utilisateur ne doit pouvoir contester les opérations qu'il a réalisé dans le cadre de ses actions autorisées, et aucun tiers ne doit pouvoir s'attribuer les actions d'un autre utilisateur.

1. Le CSA est une organisation dont le but est de promouvoir les bonnes pratiques à utiliser afin de garantir la sécurité dans le cloud computing.

1.8 Les principaux risques liés aux données dans le cloud computing

Plusieurs études menées par des chercheurs du domaine et spécialistes ont permis d'identifier les menaces à la sécurité des données dans le cloud. En effet, les infrastructures cloud sont comme tout système informatique distribuées exposées à des problématiques de sécurité classiques. Néanmoins, la multiplicité et la diversité des intervenants dans ces infrastructures ainsi que la quantité importante de leurs composants matériels et logiciels amplifient ces menaces, le CSA les a répartis en trois grandes catégories :

1. **Architecture du cloud** : qui traite principalement des questions de sécurité autour de la virtualisation des systèmes et du matériel au sein du cloud et à ses risques associés notamment à travers le multi-tenant, l'isolation des machines virtuelles, et les vulnérabilités liées aux hyperviseurs.
2. **Administration du cloud** : qui traite principalement des domaines législatifs et légaux, des audits, de la gestion et de l'interopérabilité entre les clouds.
3. **Exploitation du cloud** : qui traite principalement des problématiques liées à la gestion des accès, du chiffrement de données, des incidents, etc. dans le service cloud.

Les risques à la sécurité des données stockées dans le cloud sont donc nombreux et de différents types, nous détaillons ici certains de ces principaux défis :

— **Confidentialité des données :** confier les données à des fournisseurs de cloud signifie que les utilisateurs et les entreprises perdent la visibilité et le contrôle physique sur leurs données et comptent sur les fournisseurs pour protéger leurs données. Ce n'est pas vraiment surprenant, que certains fournisseurs ou leurs employés outrepassent leurs droits, comme dans le cas de Google, qui a déjà dû licencier des employés pour des accès illégitimes aux données des utilisateurs [31].

— **Localisation des données :** en général, l'emplacement exact des données stockées dans le cloud est inconnu, elles peuvent être situées n'importe où dans le système du fournisseur de services cloud. La plupart des fournisseurs les plus connus ont des centres de données à travers le monde, les applications et les données des utilisateurs pourraient donc être dispersé et stockées dans différents pays. Dans ce cas, les entreprises et les particuliers qui envisagent

d'héberger leurs données sensibles chez ces fournisseurs sont face à une vraie problématique. En effet, si les données des utilisateurs sont stockées dans un pays X, les législations et réglementation du pays X seront appliquées sur ces données. Le Patriot Act est souvent cité en exemple, c'est une loi antiterroriste créée suite aux évènements du 11 septembre 2001 et amendée en 2005. Il permet aux services de sécurité américains d'accéder aux données entreposées sur leur territoire ou à l'étranger si elles sont détenues par des sociétés américaines et vu que la plupart des acteurs du cloud computing sont américains (Microsoft, Amazon, Salesforce, Google), ce qui rend alors la préservation de la vie prive dans les services de cloud offerts par ces fournisseurs plus complexe.

— **Tolérance aux pannes et la continuité du service :** les fournisseurs de services cloud doivent assurer la sécurité des données dans les catastrophes naturelles ou artificielles. Généralement, les données sont répliquées sur plusieurs sites. Toutefois, dans le cas d'un tel événement indésirable, le fournisseur doit garantir la restauration complète et rapide.

— **Enfermement des données :** les utilisateurs qui adoptent le cloud risquent de ne pas pouvoir migrer leurs données vers d'autres clouds. En effet, les utilisateurs peuvent vouloir déplacer leurs données et ainsi quitter un fournisseur qui ne répond plus à leurs besoins et exigences. Cependant, dans leurs formes et fonctionnements actuels, les infrastructures et plateformes cloud n'appliquent pas des méthodes standards pour stocker les données des utilisateurs, ce qui implique qu'ils ne sont pas portables d'un cloud vers un autre.

— **Impact du multi-tenant :** les fournisseurs de service IaaS partagent l'infrastructure physique entre plusieurs clients. Si l'architecture multi-tenant est mal configurée, une faille au niveau de l'hyperviseur peut permettre à des machines virtuelles malveillantes d'obtenir un niveau non approprié de contrôle sur l'infrastructure physique sous-jacente. Donc si une machine virtuelle a réussi à exploiter une ressource physique qui a déjà été allouée aux autres machines virtuelles, et si les données qui sont écrites sur le support physique ne sont pas effacées avant que cette ressource ne soit réaffectée à une autre machine virtuelle, il existe dans ce cas un risque de divulgation des données. Il faut noter aussi que les composants sous-jacents de l'infrastructure cloud comme les caches CPU et les espaces de stockages physiques n'ont pas été conçus en prenant en compte

une propriété d'isolation pour les architectures multi-tenants.

1.9 Conclusion

Nous avons montré dans ce chapitre que le cloud computing offre beaucoup d'avantages aux entreprises et aux particuliers en termes de réduction de coûts, d'adaptation et de passage à l'échelle selon le besoin. En effet, la nouveauté du paradigme cloud computing réside dans la composition des technologies existantes et le nouveau modèle économique qui ne se fonde pas sur la vente du produit qu'il soit logiciel ou matériel mais sur la vente de son usage via internet sous forme d'un service. Cependant, avec ce nouveau mode de consommation de ressources informatiques introduit par le cloud computing, de nombreux problèmes de sécurité se posent. En complément des problèmes classiques de sécurité des systèmes informatiques répartis, le cloud computing présente des facteurs de risque supplémentaire du fait de leurs caractéristiques (la virtualisation, la mutualisation, etc.).

L'une des préoccupations les plus sérieuses concernant les entreprises et les particuliers est la confidentialité de leurs données sensible sauvegardées dans le cloud. En effet, l'hébergement des données sensibles dans le cloud peut augmenter considérablement la possibilité d'une utilisation abusive de ces données, ce qui pourrait endommager gravement la réputation ou les finances des consommateurs de services.

Afin d'encourager les entreprises et les particuliers à migrer vers le cloud, les fournisseurs de services cloud doivent s'engager à mettre en œuvre des mesures de sécurité appropriée afin de préserver le caractère confidentiel des données sensibles hébergées dans leurs plateformes. C'est clair que les données sauvegardées dans le cloud peuvent être invoquées pour différentes raisons, dans ce cas, toute faiblesse au niveau de la gestion des droits d'accès fait augmenter le risque de divulgation de ces données confidentielles. Pour cela, les fournisseurs de cloud doivent avoir des mécanismes d'autorisation flexibles et dynamiques qui permettent de limiter l'accès aux données sensibles et leurs utilisations aux seuls entités autorisées et cela sous différents contextes.

Dans le chapitre suivant, nous allons donner une vue globale sur les différents modèles de contrôles d'accès existants afin de déterminer comment les améliorer et les adapter pour assurer la protection de données privées dans cet environnement émergent.

CHAPITRE 2
EVOLUTION DES MODÈLES DE CONTRÔLE D'ACCÈS

2.1 Introduction

Afin d'assurer la confidentialité et l'intégrité des données stockées dans un système d'information, il est indiscutable que chaque accès à ces données soit contrôlé et bien évidemment tout accès non autorisé devrait être impérativement bloqué et cela peut se faire par l'intermédiaire d'un mécanisme de contrôle d'accès régi par les politiques de sécurité du système. Avec l'évolution constante de l'environnement informatique et des systèmes d'information, des nouvelles exigences de la gestion des droit d'accès apparaissent, il est devenu nécessaire d'exprimer des règles d'autorisation de plus en plus riche et complexe afin de permettre aux organisations et aux entreprises d'exprimer les spécificités de leurs métiers dans leurs politiques de sécurité déployées dans ces nouveaux systèmes. Divers modèles de contrôles d'accès ont été proposé dans la littérature afin de traduire au mieux les politiques de sécurité et de faciliter la gestion des droits d'accès aux utilisateurs du système. Notre but est de déterminer quel modèle de contrôle d'accès pourrait répondre mieux aux besoins de contrôle d'accès aux données sensibles sauvegardées dans le cloud. Pour cela, nous allons étudier dans ce chapitre les principaux travaux de recherche qui ont proposé de nouveaux modèles de contrôles d'accès et les modèles qui introduisent des enrichissements et des contraintes aux modèles de contrôle d'accès existants, en précisant les motivations sous-jacentes à l'introduction de ces modèles mais avant de présenter ces modèles, il est nécessaire de définir au préalable les concepts fondamentaux liés au domaine du contrôle d'accès.

2.2 Généralités sur les politiques de contrôle d'accès

Les politiques de contrôle d'accès sont définies comme étant des règles ou des directives de haut niveau qui spécifient qui a le droit d'effectuer quelle action sur quelle donnée [32]. La mise en œuvre d'une politique de contrôle d'accès dans un système passe en général par trois étapes principales :

- **Définir la politique de sécurité** : cette étape consiste à établir de manière abstraite les règles qui régissent l'accès aux ressources du système.
- **Écrire le modèle de la politique de contrôle d'accès** : cette étape consiste à décrire de manière formelle la politique de sécurité décrite dans l'étape précédente. Le modèle de contrôle d'accès doit être :
  - suffisamment expressif, pour permettre l'expression des règles d'autorisations complexes et de représenter fidèlement la structure et le fonctionnement de l'organisation,
  - décidable, pour évaluer les demandes d'accès à partir des règles d'autorisations. Ils doivent générer une décision qui peut être positive pour permettre l'accès à l'objet ou négative pour y refuser l'accès,
  - facile à gérer, le modèle doit fournir un ensemble de primitives et de fonctionnalités simplifiant les activités des administrateurs.
- **Mise en œuvre de la politique de contrôle d'accès** : cette étape consiste à implémenter la politique de sécurité formellement décrite dans l'étape précédente.

L'expression des règles d'autorisation dans les différents modèles de contrôle d'accès a subi des changements au niveau de sa structure. Néanmoins, il existe des concepts communs qui se trouvent dans tous les règles de contrôle d'accès, qui sont les suivants :

- **Sujet** : entité active qui accède aux données du système. Le sujet peut être un utilisateur, une machine, un processus, un programme, etc.
- **Objet** : entité passive du système susceptible de faire l'objet d'accès. L'objet peut être un fichier, une base de données, une machine, un programme, etc.
- **Droit d'accès** : représente l'action à traiter par le sujet sur l'objet. L'action peut être lire, écrire, exécuter, etc.

Les règles d'autorisation sont exprimées sous forme de permission ou interdiction et même sous forme d'une obligation.

- **Permission** : spécifie l'ensemble des sujets qui peuvent accéder aux objets.
- **Interdiction** : spécifie l'ensemble des sujets qui ne peuvent pas accéder aux objets.
- **Obligation** : permet d'exprimer le fait que deux actions sont liées, donc une action doit être réalisée avant ou après l'exécution d'une action demandé, par exemple, les médecins sont obligés de garder les dossiers médicaux de leurs patients pendant la durée fixée par la loi.

Les politiques de contrôle d'accès peuvent être classées comme ouverte, fermée ou mixte :

- **La politique ouverte** : l'utilisateur a le droit d'accéder à tous les objets du système sauf si une règle d'autorisation a été définie explicitement lui interdisant l'accès aux objets.
- **La politique fermée** : l'accès se fait uniquement en spécifiant d'une façon explicite un ensemble de règles d'autorisation. En d'autres termes, si aucune règle d'autorisation n'est définie pour un utilisateur, l'accès lui sera interdit.
- **La politique mixte** : cette politique permet d'exprimer à la fois des permissions et des interdictions afin d'établir clairement la politique de sécurité d'un système d'information.

Une politique de contrôle d'accès peut être statique ou dynamique, la politique de contrôle d'accès est dite statique si les droits d'accès des utilisateurs ne changent pas par rapport à l'évolution dynamique du système, car les règles de sécurité comporte seulement des contraintes statiques. La politique de contrôle d'accès est dynamique si les droits d'accès sont évaluables au cours d'une requête d'accès et non une fois pour toutes comme les politiques statiques. L'idée de base est que les politiques ainsi que les décisions du contrôle d'accès, soient variables au fil du temps, en fonction de certains nombre de paramètres relatifs au contexte.

Comme nous l'avons mentionné précédemment, les modèles de contrôle d'accès sont mis en œuvre par des mécanismes de contrôle d'accès. Il est généralement recommandé d'organiser ces mécanismes de façon à implémenter la notion de moniteur de référence [33]. Le moniteur de référence est un intermédiaire entre les utilisateurs et les ressources auxquelles ces derniers essaient d'accéder, et qui veille à ce que les ressources ne soient accessibles que par des utilisateurs ayant le droit d'y accéder. Le moniteur de référence doit être :

- **inviolable** : il ne doit pas pouvoir être modifié,
- **incontournable** : il ne doit pas être possible d'accéder à un objet sans être contrôlé par le moniteur de référence,
- **totalement vérifié** : il ne doit comporter aucune faute de conception ou de réalisation.

2.3 Les principaux modèles de contrôle d'accès existants

Dans cette section, on présente les modèles de contrôle d'accès qui proposent des nouveau principes et créent des nouvelles structurations de droits d'accès afin de pouvoir [32] :

— organiser les droits de la façon la plus proche possible de la structure des organisations, et donc permettre aux administrateurs de manipuler les droits d'une façon plus intuitive,

— réduire le nombre de règles de sécurité à définir, pour éviter les erreurs en regroupant certaines de ces entités.

2.3.1 Les modèles discrétionnaires et obligatoires

Le modèle discrétionnaire DAC (Discretionary Access Control) [34] est basé sur la notion de propriétaire, chaque utilisateur qui crée un objet peut propager et manipuler librement les droits d'accès sur cet objet. Ce principe ne semble pas applicable aux niveaux des entreprises et des organisations. Le modèle DAC présente de graves inconvénients vis-à-vis des fuites d'informations, celles-ci peuvent affecter la sécurité du système. En plus, le modèle DAC associe les privilèges directement aux utilisateurs, ce qui rend l'administration des droits d'accès dans ce modèle difficile à gérer pour les grands systèmes. En effet, a chaque fois qu'une nouvelle mise à jour a lieu dans le système, il faut modifier toutes les règles d'accès qui portent sur ces entités, ce qui augmente les risques d'erreurs dans la politique de sécurité.

La gestion des droits d'accès dans le modèle obligatoire est plus sûre par rapport au modèle discrétionnaire. Le modèle MAC (Mandatory Access Control) [35, 36] fixe des niveaux hiérarchiques de sécurité aux ressources (public, confidentiel, secret,...) et des niveaux d'habilitation aux sujets (public, confidentiel, secret, ...). Les règles d'autorisations sont établies et imposées par une autorité centrale non modifiable par les utilisateurs finaux, les permissions sont basées sur une relation de comparaison entre le niveau de sensibilité de la ressource demandée avec le niveau d'habilitation du sujet en question. Ce modèle est strict et non flexible pour être appliqué dans un système dynamique.

2.3.2 Le modèle de contrôle d'accès basé sur le rôle

Le modèle RBAC (Role Based Access Control) [37, 38] a introduit une nouvelle structuration des droits centrée sur le concept rôle, qui est une entité intermédiaire entre les utilisateurs et les permissions. Le rôle représente d'une façon abstraite une fonction ou une autorité au sein d'une organisation qui peut être attribué à un ou plusieurs utilisateurs. Les permissions dans ce modèle sont un ensemble d'actions ou de responsabilités qui peuvent être effectuées par un rôle. Donc tout utilisateur a reçu l'autorisation de jouer un rôle, il hérite alors les privilèges associés à ce rôle mais il faut mentionner que l'utilisateur peut activer ses rôles que dans le cadre de sessions. Une session est un autre nouveau concept introduit par le modèle RBAC. Une session est l'entité qui représente un utilisateur actif au sein du système. La session est liée à une notion temporelle, elle permet de ne pas avoir à authentifier l'utilisateur à chaque accès au système. L'inconvénient connu d'une session est que si un utilisateur oublie de fermer sa session, un autre utilisateur pourra se faire passer pour lui. Pour limiter ce problème, l'une des solutions est de restreindre la session dans le temps. Un utilisateur peut invoquer une multitude de sessions et une session est attribuée à un et un seul utilisateur durant laquelle il peut activer tout ou partie des rôles qui lui sont attribués. Les différents relations entre les éléments du modèle RBAC sont présentées dans la Figure 2.1, et elles seront ensuite détaillées dans le Tableau 2.1.

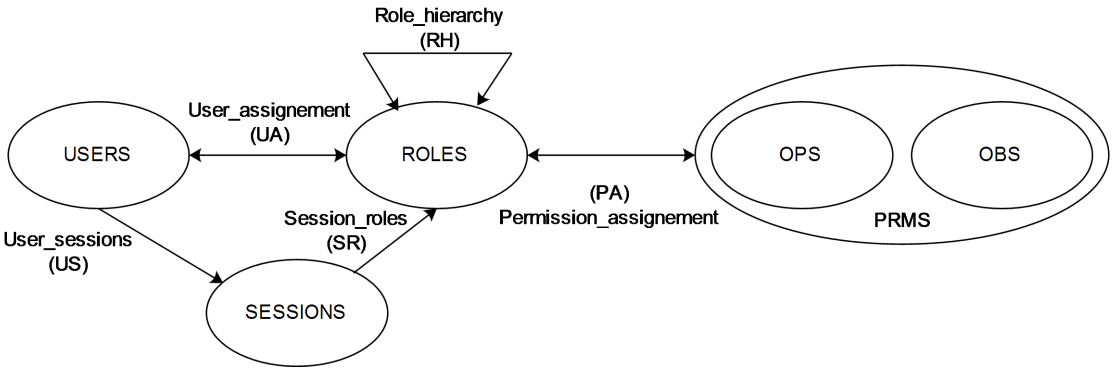

Figure 2.1 – Le modèle RBAC

L'utilisation de la notion de rôle dans un modèle de contrôle d'accès présente de nombreux avantages. Elle permet de faciliter et simplifier le travail des administrateurs. En effet, si un nouvel utilisateur rejoint une entreprise ou bien sa fonction dans l'entreprise change, il suffit juste d'ajouter ou d'enlever un rôle à cet utilisateur, de même, si les permissions liées à un rôle changent, l'administrateur doit uniquement modifier

Notation	Description
$USERS$	ensemble fini d'utilisateurs
$ROLES$	ensemble fini de rôles
OPS	ensemble fini d'opérations
OBS	ensemble fini d'objets
$SESSIONS$	ensemble fini de sessions
$PRMS \subseteq OBS \times OPS$	Une permission signifie l'autorisation d'exécuter une opération autorisé sur un objet. Une opération peut être autorisée sur plusieurs objets et un objet peut être attribuées différentes permissions.
$UA \subseteq USERS \times ROLES$	Affectation plusieurs à plusieurs associant les utilisateurs aux rôles, un utilisateur peut jouer plusieurs rôles dans une seule session et un rôle peut être attribué à plusieurs utilisateurs.
$PA \subseteq ROLES \times PRMS$	Affectation plusieurs à plusieurs entre rôles et permissions. L'attribution d'un rôle garantira plusieurs permissions à l'utilisateur et une permission peut être attribuée à plusieurs rôles.
$US \subseteq SESSIONS \times USERS$	Relation plusieurs à un entre sessions et utilisateurs. Une session ne peut être activée que par un seul utilisateur, qui peut activer une ou plusieurs sessions.
$SR \subseteq SESSIONS \times ROLES$	Relation plusieurs à plusieurs entre sessions et rôles. Dans une session, on peut activer un ou plusieurs rôle Un rôle peut être activé dans une ou plusieurs sessions.
$RH \subseteq ROLES \times ROLES$	Une hiérarchie de rôles qui définit un ordre partiel, noté $\leq$, où $R_1 \leq R_2$ (toutes les permissions accordées à R_1 sont accordées à R_2)

Tableau 2.1 – Les concepts et les relations du modèle RBAC

les droits associés a ce rôle. En plus, les rôles peuvent êtres organisés de manière à former une hiérarchie, où chaque rôle hérite des autorisations des rôles qui lui sont hiérarchiquement inférieurs. A titre d'exemple, le rôle de directeur est hiérarchiquement supérieur aux rôles de sous-chefs qui sont eux-mêmes hiérarchiquement supérieurs aux simples employés, donc le directeur en plus de ses droits, hérite implicitement les privilèges des rôles de sous-chef et d'employé. La hiérarchisation permet donc de réduire le nombre de rôles et limite la redondance des affectations. Néanmoins, le modèle RBAC présente aussi des inconvénients. En effet, dans le modèle RBAC, tous les utilisateurs ayant le même rôle ont les mêmes privilèges, par exemple, il est difficile avec RBAC de gérer des règles de types seuls les médecins traitants peuvent accéder aux informations médicales du dossier d'un patient. En plus, le concept de permission est primitif, ils dépendent de la mise en œuvre concrète du modèle. Il serait préférable d'ajouter au modèle une structure générique de permission. Enfin, il est difficile d'appliquer le modèle RBAC dans un système qui inclut plusieurs organisations qui coopèrent [39].

2.3.3 Le modèle de contrôle d'accès basé sur la vue

C'est un modèle de sécurité proposé par SQL pour les bases de données relationnelles [40, 41]. Dans les systèmes de gestion de bases de données relationnelles, les vues sont destinées aux utilisateurs finaux. Elles sont des représentations construites sur le modèle logique de données sous-jacent. Elles peuvent être soit virtuelle (elles sont calculées sous forme de réécriture de requêtes), soit matérialisée (elles sont stockées puis elles sont mises à jour). Du point de vue du contrôle d'accès, les vues sont des objets abstraits qui masquent les objets concrets sous-jacents. Les vues constituent un moyen efficace pour exprimer les politiques d'autorisation dans les systèmes de gestion de base de données en limitant l'accès au modèle logique en fonction de l'action que l'utilisateur souhaite accomplir et cela à l'aide des instructions GRANT (qui permet d'accorder une nouvelle permission à un utilisateur) et REVOKE (qui permet de supprimer une permission que possédait un utilisateur).

2.3.4 Le modèle de contrôle d'accès basé sur la tâche

De façon générale, le modèle TBAC (Task Based Access Control) [42] fournit un nouvel encadrement visant à ajouter la notion de tâche (action composite) dans des

règles d'autorisation afin de répondre au besoin de contrôler la réalisation des activités dans les applications de type workflow. Pour cela, le modèle TBAC fait une distinction entre l'affectation et l'activation des permissions par rapport à des tâches données aux utilisateurs au sein de l'organisation. L'utilisateur ne doit obtenir une permission que lorsque si nécessaire pour poursuivre l'exécution de l'activité considérée. Par exemple, dans une agence de voyage, la tâche d'achat d'un billet d'avion se décompose en plusieurs actions plus élémentaires telles que la réservation du billet, le paiement du billet et l'édition d'une facture. La permission d'éditer une facture ne doit être activée qu'après la réservation et l'achat du billet. Dans ce modèle, les permissions relatives à une tâche donnée sont groupées dans des autorisations qui représentent les abstractions fondamentales du modèle.
Cependant, le modèle TBAC est basé sur l'avancement des tâches des utilisateurs et il ne prend pas en compte des contraintes sur les horaires ou périodes d'accès pendant lesquels les utilisateurs sont en charge de la réalisation de leurs activités [43].

2.3.5 Le modèle de contrôle d'accès basé sur l'équipe

Dans ce modèle [44], l'entité de base est l'équipe. C'est une abstraction qui encapsule un ensemble d'utilisateurs ayant des rôles différents et qui collaborent dans le but d'atteindre un objectif particulier. Ce modèle a été proposé dans le but de fournir un contrôle d'accès pour les systèmes d'information ayant des activités nécessitant la collaboration de plusieurs acteurs. Dans ce modèle, on trouve deux aspects importants du contexte de collaboration : le contexte des utilisateurs qui permet d'identifier les utilisateurs qui jouent un rôle dans une équipe à un moment donné et le contexte des objets qui permet d'identifier les objets requis dans le processus de collaboration. La particularité dans ce modèle est que l'attribution des rôles et l'activation des permissions sont gérées séparément. Un utilisateur obtient par le biais de son appartenance à une équipe le droit d'accès aux ressources de l'équipe. Les privilèges dont disposent les utilisateurs sont donc le résultat de l'union des permissions accordées aux rôles exercés par les utilisateurs et des permissions accordées aux équipes dont les utilisateurs font partie. Cependant, ce modèle a une imperfection concernant la gestion des droits d'accès, il introduit deux relations binaires, rôle-autorisation et équipe-autorisation, ce qui peut engendrer des conflits potentiels entre les droits d'accès. Cette imperfection du modèle a été corrigée dans le modèle OrBAC [39].

2.3.6 Le modèle de contrôle d'accès basé sur l'organisation

Le but principal du modèle OrBAC (Organization Based Access Control) [45] est d'exprimer la politique de sécurité avec des notions abstraites et de dériver ensuite de manière automatique les droits d'accès concrètes. Le modèle OrBAC définit la notion d'organisation comme une entité principale du modèle et fait l'abstraction des sujets, des objets et des actions respectivement en rôle, activité et en vue. Un rôle est attribué à un groupe d'utilisateurs, une activité est attribuée à une ou plusieurs actions, et une vue est attribuée à un ou plusieurs objets. Les affectations des sujets aux rôles dans

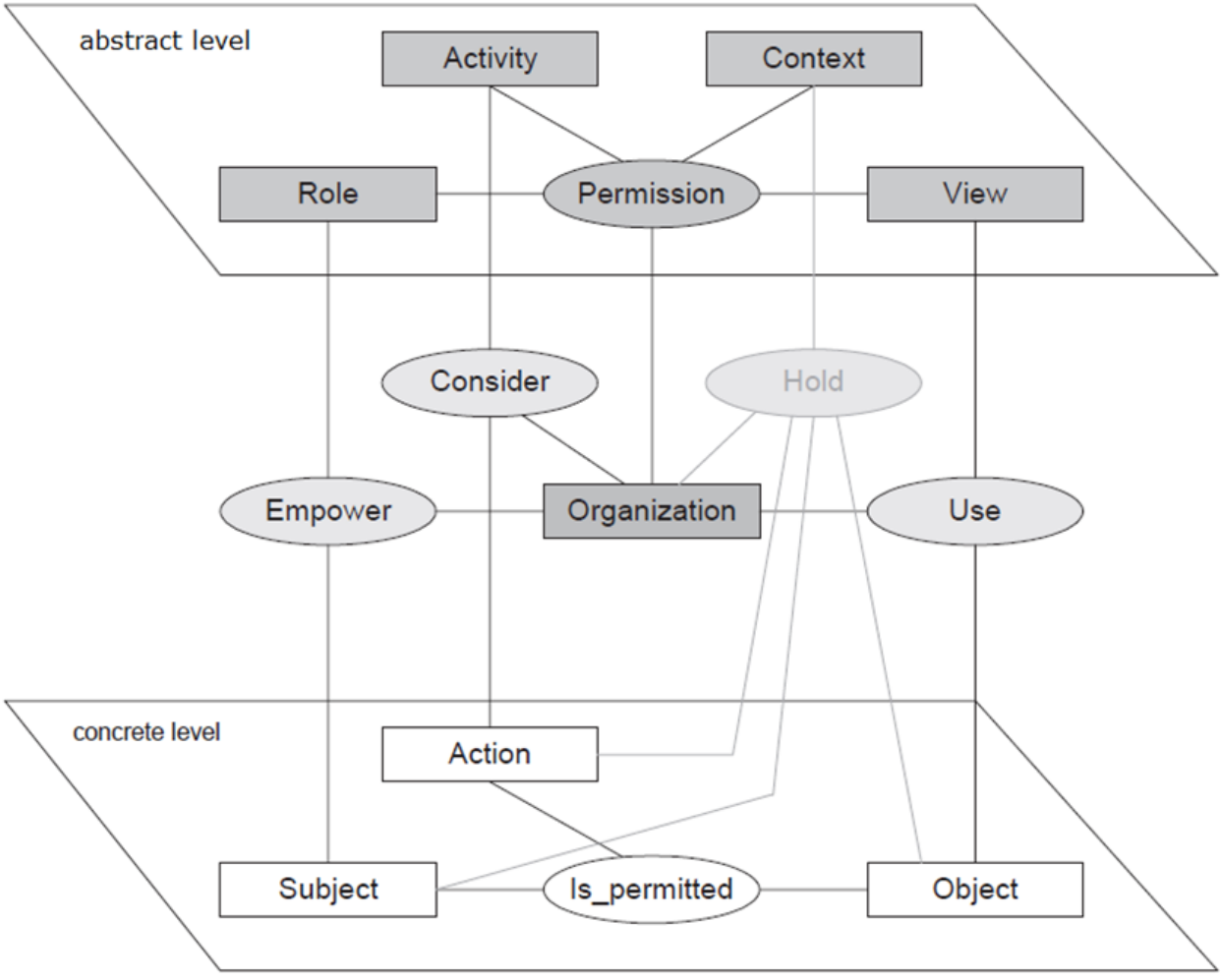

Figure 2.2 – Les concepts et les relations du modèle OrBAC

chaque organisation sont spécifies en utilisant la relation ternaire $Empower$, cette modélisation permet de considérer qu'un même sujet peut jouer des rôles différents suivant l'organisation considérée. De façon similaire, deux autres relations ternaires $Consider$ et Use permettent de spécifier, pour chaque organisation, les relations entre action et activité d'une part, et entre objet et vue d'autre part.

Tableau 2.2 – Le modèle OrBAC

$Empower$	$Organization\times$ $Subject\times$ $Role$	Si org est une organisation, s est un sujet et r est un rôle, alors $Empower(org, s, r)$ signifie que l'organisation org habilite le sujet s à jouer le rôle r.
$Consider$	$Organization\times$ $Action\times$ $Activity$	si org est une organisation, ac est une action et av est une activité, alors $Consider(org, ac, av)$ signifie que l'organisation org considère l'action ac comme faisant partie de l'activité av.
Use	$Organization\times$ $Object\times$ $View$	Si org est une organisation, o un objet et v une vue, alors $Use(org, o, v)$ signifie que l'organisation org utilise l'objet o dans la vue v.
$Hold$	$Organization\times$ $Subject\times$ $Action\times$ $Object\times$ $Context$	si org est une organisation, s est un sujet, ac est une action, o est un objet et c est un contexte, alors $Define(org, s, a, o, c)$ signifie qu'au sein de l'organisation org, le contexte c est vraie entre le sujet s, l'objet o et l'action ac.
$Permission$	$Organization\times$ $Role\times$ $Activity\times$ $View\times$ $Context$	Si org est une organisation, r est un rôle, av est une activité, v est une vue et c est un contexte, alors $Permission(org, r, av, v, c)$ signifie que l'organisation org accorde au rôle r la permission de réaliser l'activité av sur la vue v dans le contexte c.
$Is-permitted$	$Subject\times$ $Action\times$ $Object$	$Permission(org, r, av, v, c) \wedge Empower(org, s, r)$ $Consider(org, a, av) \wedge Use(org, o, v)$ $\wedge\ Define(org, s, a, o, c)$ $\rightarrow Is-permitted(s, a, o)$.

Le modèle OrBAC offre également la possibilité d'exprimer des autorisations contextuelles. Pour cela, OrBAC introduit également la notion de contexte qui est définit comme une condition logique qui permet de conclure que l'on se trouve dans le contexte lorsque cette condition est satisfaite. En effet, le concept contexte dans OrBAC est relié aux concepts concrets (sujet, action et objet) et au concept d'organisation par la relation $Hold$. Le concept de contexte est utilisé ensuite dans la définition des permissions abstraites. Cuppens et all. ont proposé une taxonomie de différents types de contexte qui peuvent être modélisé dans le modèle OrBAC [46] :

— **Le contexte temporel** : permet de contraindre la date et la durée de validité d'une règle d'autorisation.

— **Le contexte spatial** : correspond au lieu depuis lequel un utilisateur peut effectuer une action.

— **Le contexte déclaré par l'utilisateur** : correspond à un contexte dans lequel

l'utilisateur décide de se placer pour effectuer une activité. Par exemple, un professeur peut déclarer un *contexte d'examen* et activer ainsi les permissions associées pour que les étudiants accèdent au document d'examen en ligne.

— **Le contexte prérequis** : correspond aux contraintes spécifiques au domaine d'application

— **Le contexte provisionnel** : permet d'activer des permissions en fonction des actions précédemment réalisées.

Une politique d'autorisation en OrBAC s'exprime en spécifiant, pour chaque organisation, les activités que les rôles ont la permission de réaliser sur les vues dans un contexte. La politique d'autorisation s'exprime donc de façon complètement indépendante des ensembles de sujets, actions et objets gères par le système. Le modèle propose aussi une règle de passage pour dériver automatiquement, les actions concrètes qui peuvent être réalisées par les sujets sur les objets. Les règles exprimées dans le modèle OrBAC peuvent définir non seulement des permissions, mais aussi des interdictions, des obligations et des recommandations. La combinaison entre toutes ces règles pourrait conduire à des situations de conflit qui exige, par conséquent, des mécanismes pour détecter et résoudre les conflits. Une situation de conflits se produit quand la politique d'autorisation dérive à la fois qu'un utilisateur a la permission et l'interdiction, ou l'obligation et l'interdiction, d'effectuer une action sur un objet. Afin de résoudre ces situations de conflits, Benferhat et all. [47] ont proposé d'associer un niveau de priorité à chaque règle de sécurité. En comparant ces valeurs, on déduit la décision de l'accès.
Dans OrBAC, on peut également définir des hiérarchies de rôles, mais aussi d'activités, de vues et d'organisations. Chacune de ces hiérarchies est associée à un mécanisme d'héritage des permissions et des interdictions. La définition de ces hiérarchies permet une expression plus modulaire de la politique d'autorisation organisationnelle.
La Figure 2.2 montre la structure du modèle OrBAC, les deux niveaux abstrait et concret, ainsi que les différentes relations existantes entre les entités de ces deux niveaux. Les prédicats de base utilisés dans OrBAC pour modéliser les relations entre ces entités sont résumés au Tableau 2.2. Bien que l'expressivité dans le modèle OrBAC est meilleure que les autre modèles, mais les règles d'autorisation sont néanmoins limités par la définition des activités simple, alors que les systèmes actuels demande de prendre en considération des activités composées d'une séquence d'action.

2.3.7 Le modèle de contrôle d'accès basé sur l'attribut

Avec l'évolution des architectures orientées service SOA (Services Orienté Architectures) et les services web, le problème de la gestion des droits d'accès devient plus compliqué, il faut gérer des sujets qui n'appartiennent à aucune organisation et qui souhaitent obtenir un service (par exemple, acheter un livre en ligne). Les modèles de contrôle d'accès précédents exigent l'authentification tandis que celle-ci nécessitent un système ouvert, qui permet un accès restreint aux utilisateurs non identifiés ou semi anonymes. Il faut donc un mécanisme de contrôle d'accès très flexible, qui permet aux utilisateurs non identifiés de présenter une preuve nécessaire et suffisante de leurs droits d'accès à une ressource ou un service. Le modèle ABAC (Attribut Based Access Control) [48] a été introduit en proposant une nouvelle vision de l'expression d'une politique de sécurité en se basant sur la notion d'attribut. Un attribut se compose d'un type et d'une valeur pour exprimer une propriété d'une entité dans le système. Le modèle ABAC est basé sur trois entités sujet, ressource et environnement. Un sujet peut donc avoir différents attributs qui définissent son identité et ses caractéristiques tels qu'un identifiant, un nom, un rôle, une adresse, etc. Une ressource peut avoir aussi différents types d'attributs variables selon sa nature, par exemple l'identifiant, le type, la date de modification, etc. Les attributs d'environnement peuvent être décrits par des informations opérationnelles, techniques, liées à la situation ou au contexte dans lequel l'accès à l'information se produit, comme par exemple : la date, le niveau de sécurité du réseau, le débit de la connexion, etc.

Le modèle ABAC définit des règles d'autorisation en spécifiant des conditions à n'importe quel attribut du triplet $(sujet, ressource, environnement)$. L'accès donc est accordé à un sujet S sur une ressource R dans un environnement E s'il existe une fonction F qui sera satisfaite à partir des valeurs liés au triplet (S, R, E), telle que :

$can\_access(S, R, E) \leftarrow F(ATTR(S), ATTR(R), ATTR(E))$ où $ATTR$ représente une combinaison de fonctions booléen prenant en argument les valeurs des attributs d'une entité. Dans les cas où il n'existe pas de fonction F permettant d'accorder l'accès, le sujet se voit refuser l'accès à la ressource.

La simplicité des règles de contrôle d'accès du modèle ABAC offre un avantage en expressivité et en flexibilité, il suffit d'ajouter un attribut que ce soit aux sujets, aux ressources ou à l'environnement pour prendre en compte un nouveau paramètre dans la définition des règles d'autorisations. Néanmoins, le fait de considérer par exemple un

rôle comme un attribut, fait perdre tous les avantages du modèle RBAC, on ne peut plus considérer qu'un rôle représente un ensemble de permissions, le modèle ABAC ne propose pas donc une réelle structuration dans l'expression des règles d'accès, ce qui implique une gestion de politique très complexe. Le langage XACML (eXtensible Access Control Markup Language) a été proposé par OASIS comme un standard pour la définition des politiques de contrôle d'accès basé sur ABAC, OASIS a décrit aussi une architecture de mise en œuvre de ces politiques. Il est à noter que la mise en place d'une politique de sécurité basé sur ABAC nécessitera la mise en œuvre de certains mécanismes pour assurer la gestion des identités, leur certification, ainsi que des mécanismes cryptographiques afin d'assurer l'intégrité des données échangées.

2.3.8 Modèle de contrôle d'usage

Avec l'apparition des nouvelles applications comme la gestion de droits électroniques (DRM, Digital Right Management) qui a pour but de contrôler, d'une manière générale, l'accès et l'usage des contenus numériques protégés par le droit d'auteur. Il est nécessaire de spécifier des conditions qui doivent être satisfaites non seulement avant mais aussi pendant ou après qu'une action soit réalisée. Par exemple, un serveur permettant d'écouter des morceaux de musique doit pouvoir spécifier que le paiement doit s'effectuer avant, pendant ou bien après l'écoute du morceau. Pour exprimer ce type de politique d'autorisation, les modèles de contrôle d'accès présenté précédemment ne sont plus suffisants. C'est la raison pour laquelle des modèles de contrôle d'usage commencent à être proposées. Les modèles de contrôle d'usage sont une généralisation des modèles de contrôle d'accès. Ils ne proposent plus seulement de spécifier quelles sont les conditions d'accès aux objets, ils permettent également de spécifier les contrôles pendant l'usage de ceux-ci. L'un des modèles les plus connues au niveau du contrôle d'usage est UCONabc[49, 50]. Le modèle UCONabc effectue des contrôles avant l'accès, pendant l'utilisation et à la fin de l'utilisation afin d'envisager une capacité de décision plus précise. Il est maintenant possible par exemple de révoquer l'accès à un objet en cours d'utilisation. Toutes les décisions sont prises en tenant compte des composants suivants (voir Figure 2.3) : le sujet et ses attributs, l'objet et ses attributs, un ensemble de droits génériques et une politique de sécurité comprenant des autorisations (a), des obligations (b) et des conditions (c). Les obligations sont des prérequis qui doivent être satisfaits pour la concession d'usage. Les conditions sont des prérequis de

l'environnement ou du système qui sont indépendants des sujets et des objets. Dans le modèle UCONabc, les attributs sont dits mutables, c'est à dire qu'ils peuvent être modifiés. Ces modifications peuvent intervenir à chacune des trois étapes qui caractérisent le contrôle d'usage.

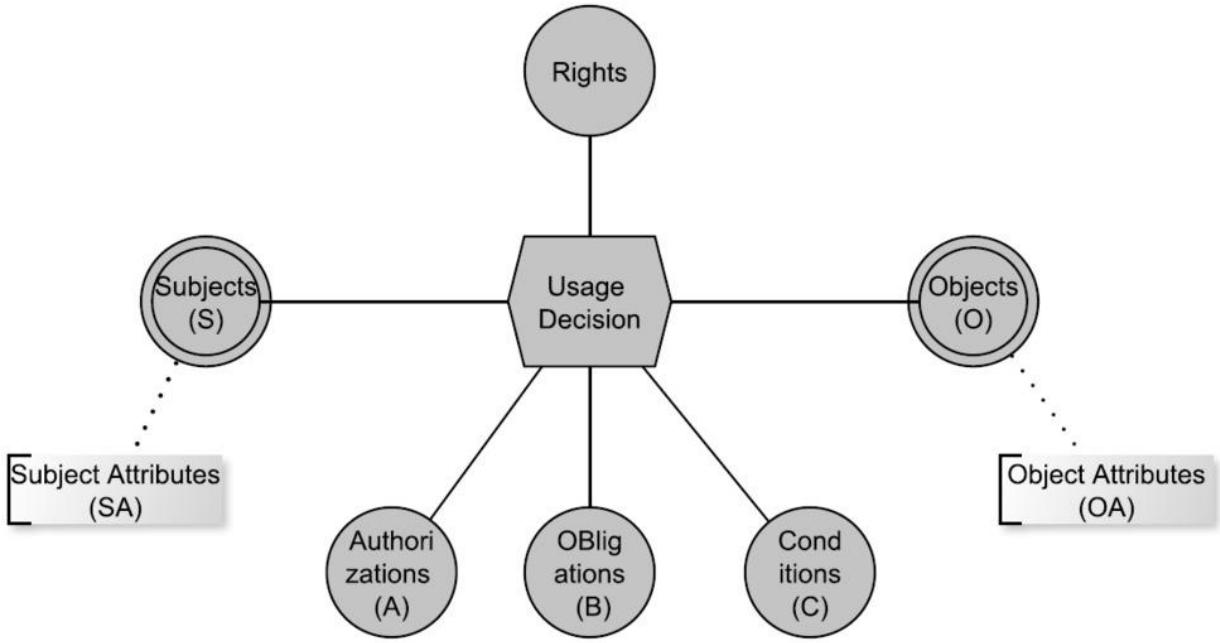

Figure 2.3 – Le modèle UCONabc

2.4 Enrichissements et extensions de modèle RBAC

Comme montre la section précédente, il existe une variété de modèles de contrôle d'accès dans la littérature, néanmoins le modèle RBAC est préférable en général pour la gestion de contrôle d'accès aux données au niveau des entreprises et organisations. Cela est dû au fait que le modèle RBAC a été standardisé [51]. En effet, il existe 4 variantes de RBAC, appelées dans la littérature la famille de RBAC. La première, appelée «RBAC0», représente le noyau de RBAC avec les concepts de base (utilisateur, rôle, opération, objet et session). La deuxième variante, appelée «RBAC hiérarchique» ou «RBAC1», introduit la notion d'héritage de rôle. «RBAC2» ajoute un ensemble de contraintes qui peuvent être appliqués sur les rôles. En 2004, ANSI (The American National Standard Institute) a proposé le standard «RBAC3», qui est la combinaison des deux modèles précédents (RBAC1 et RBAC2). Au fil du temps, ce dernier modèle est devenu insuffisant pour répondre aux besoins de la gestion des droits d'accès dans

les nouveaux systèmes. En effet, l'attribution d'une permission dans les systèmes d'information modernes est devenue de plus en plus complexe et dépendante du contexte. Plusieurs travaux s'orientent vers l'extension du standard RBAC par de nouvelles conditions et paramètres pour prendre en compte l'évolution de la définition du contexte (temps, localisation, caractéristique du système, etc.).
La notion du contexte a été utilisée en linguistique et psychologie avant d'être adoptée en informatique [52], il existe plusieurs définitions du contexte, qui varient selon le domaine d'utilisation et de l'objectif final de son application. La définition communément admise est celle proposée par Dey et all. [53] qui définit le contexte comme étant l'ensemble de toutes les informations qui peuvent être utilisées pour caractériser la situation d'une entité. Une entité pouvant être un acteur, un lieu, ou un objet de l'environnement considéré comme utile à l'interaction entre un utilisateur et une application. Dans cette section, on s'intéresse aux principaux enrichissements introduits au modèle RBAC.

2.4.1 Les contraintes de séparation des tâches

L'un des problèmes du modèle RBAC1 est d'exprimer la séparation des tâches et l'exclusion mutuelle entre les rôles, il est difficile de spécifier que certains tâches doivent être effectuées par des utilisateurs différents, par exemple, dans une entreprise on ne souhaite pas avoir un utilisateur qui puisse à la fois faire une demande d'achat et la valider. Pour limiter cela, le standard ANSI définit les modèles RBAC2 et RBAC3, en introduisant un ensemble des contraintes permettent de limiter les privilèges des utilisateurs et augmentent le pouvoir d'expression des modèles RBAC. Ces contraintes peuvent être appliquées sur l'assignation des rôles et permissions et ainsi sur les hiérarchies de rôles. Les principaux types de contraintes sont [54, 55] :

1. **L'exclusion mutuelle entre rôles** : cette contrainte assure le principe de séparation des privilèges, un même utilisateur ne peut être assigné qu'à un seul rôle dans un ensemble de rôles mutuellement exclusifs. On distingue ainsi deux types :
   (a) **Séparation statique de privilèges (Static Separation of Duty)** : il s'agit d'empêcher un utilisateur de jouer des rôles en conflits, et cela est valable quelque soit la session.
   (b) **Séparation dynamique de privilèges (Dynamic Separation of Duty)** : spécifie que, même si certains rôles peuvent être affectés à un même utilisateur,

ces rôles ne peuvent pas être activés simultanément dans une même session.

En effet, pour respecter une certaine cohérence, les contraintes de séparation doivent être établies de façon à éviter que des rôles en hiérarchie soient également en conflits, ceci a pour conséquence qu'à chaque fois qu'un rôle est affecté à un utilisateur, il faut vérifier que l'utilisateur ne joue aucun autre rôle en conflit avec son nouveau rôle soit directement, soit indirectement à travers la hiérarchie des rôles.

2. **Cardinalité** : il s'agit d'imposer des contraintes sur le nombre d'utilisateur assignés à un certain rôle, par exemple, au plus une personne doit être associe au rôle directeur.
3. **Rôles pré-requis** : dans certains cas, il est intéressant d'imposer des pré-requis en matière d'assignation de rôle ou de permissions. Par exemple, un utilisateur ne peut être assigné au rôle A que s'il appartient déjà au rôle B, et très souvent A sera un rôle senior par rapport à B. De même pour les permissions, un rôle peut se voir attribué une permission donnée uniquement s'il possède déjà une autre permission définie à l'avance.

Il est important de noter que les contraintes, quelles qu'elles soient, ne sont valables que si un contrôle strict est appliqué en ce qui concerne l'assignation des comptes d'utilisateurs aux êtres humains. Si un employé possède déjà deux comptes, eux-mêmes appartenant à deux rôles sensés être mutuellement exclusifs, les contraintes n'ont plus aucune efficacité.

2.4.2 Les contraintes temporelles

L'intégration de l'aspect temporel est le premier élément contextuel qui a été inclus dans le modèle RBAC pour limiter l'activation et la désactivation d'un rôle.

— **Le modèle Temporal-RBAC** : dans le modèle TRBAC (Temporal Role Based Access Control) [56], l'aspect temporel a été introduit dans la structure d'une politique de sécurité. Les rôles se caractérisent souvent par une dimension temporelle. Un rôle peut être actif durant une certaine période temporelle et non actif à d'autres. Un rôle activé est un rôle qu'un utilisateur peut activer durant une session, c'est-à-dire que l'utilisateur peut obtenir ce rôle. Pour cela, dans ce modèle,

des rôles déclencheurs (trigger role) ont été mis en place dans le but de savoir à quel instant ces rôles déclencheurs passent d'une position active à une position passive et inversement. Ce modèle de contrôle d'accès pose des problèmes lorsqu'il est combiné avec la hiérarchie de rôle.

— **Le modèle Generalized Temporal-RBAC** : le modèle Generalized Temporal-RBAC (GT-RBAC) [57] étend le modèle Temporal-RBAC en autorisant de nouvelles formes de contraintes temporelles. Le modèle GT-RBAC prend en compte chacune des relations qui relient les entités dans RBAC et permet la définition de contraintes temporelles. Ainsi, on peut contraindre :
- L'activation et la désactivation périodique de rôle,
- Les affectations de rôles aux utilisateurs et de permissions aux rôles,
- La durée pendant laquelle on peut endosser un rôle.

Cependant, cette multiplication de types de contrainte engendre des conflits. De plus, des activations arbitraires des déclencheurs associés aux contraintes d'interdépendance entre ces déclencheurs peuvent créer des ambiguïtés. On peut facilement créer une boucle en liant deux déclencheurs entre eux afin qu'à tour de rôle ils s'activent et se désactivent. Afin de résoudre ces problèmes, GT-RBAC offre la possibilité de donner des priorités aux règles d'activation. Cependant, chaque type de contrainte possède aussi son propre type de hiérarchie. Afin de résoudre les incohérences, un graphe de dépendance est établi. Celui-ci vérifie la cohérence grâce aux pré et post conditions des contraintes. Bien évidement, tous ces mécanismes rendent l'expression de GT-RBAC complexe. Afin de simplifier leur modèle, Joshi et all. proposent un algorithme [43] pour remplacer les relations d'affectation d'utilisateur aux rôles par des rôles temporels. Ainsi, ils reviennent à un modèle plus flexible et moins complexe.

— **Le modèle Extended-RBAC** : Mossakowski et all [58] proposent d'introduire le temps dans le modèle de contrôle d'accès RBAC afin de manipuler des règles de contrôle d'accès basées sur l'ordre temporel d'exécution des tâches dans les processus métier.

2.4.3 Les contraintes spatiale

Avec l'évolution des dispositifs mobiles, la prise en compte des besoins de mobilité dans les politiques de sécurité est devenue essentielle afin d'exprimer des restrictions

sur la localisation dans les permissions, par exemple, seuls les étudiants qui se trouvent dans la salle A5 ont le droit de passer l'examen en ligne.

— **Le modèle Geographical-RBAC** : le modèle Geographical-RBAC (GEO-RBAC) [59] étend le modèle RBAC en intégrant la localisation géographique comme une contrainte à satisfaire pour activer des rôles. Les rôles donc sont activés dans des places bien déterminées pour limiter géographiquement l'utilisation des rôles. Le modèle GEO-RBAC a introduit la notion de rôle spatial correspondant au couple $< role\_name, extent >$, i,e. un rôle spatial est muni d'informations indiquant les localisations où le rôle peut être activé. Le principe proposé dans ce modèle est de comparer une position physique, supposée obtenue de façon fiable (par exemple la localisation GPS), à des positions logiques (exemples : route, ville, région) auxquelles sont associés des rôles spatiales.

Les représentations utilisées par GEO-RBAC sont mutuellement exclusives, si les conditions liées à une localisation sont satisfaites alors l'ensemble des conditions liées aux autres localisations ne peut pas être satisfait. Afin de valider leur approche, Damiani et all. ont découpé l'étude de GEO-RBAC en quatre parties distinctes tout comme le modèle RBAC : GEO-RBAC core, GEO-RBAC hiérarchique, GEO-RBAC avec contraintes et GEO-HRBAC avec contraintes et hiérarchies.

2.4.4 Les contraintes lies à la confiance

Avec l'évolution de l'ubiquité, différents modèles sensibles au contexte ont été proposés.

— **Le modèle Trust and Context Based Access Control** : dans [60], les auteurs proposent une extension du modèle RBAC avec la notion de confiance et de contexte (TCAC) pour les systèmes ouverts et distribués, donc l'attribution des permissions dans TCAC est basée sur le rôle, la confiance et le contexte de l'utilisateur qui demande l'accès. La notion de confiance est subjective, elle peut être quantifiée en analysant le comportement des utilisateurs, sa valeur est dans l'intervalle réel $[0, 1]$, la valeur 0 signifie absolument pas digne de confiance, et 1 signifie totalement fiable. Chaque utilisateur est associé à une valeur de confiance particulière au cours d'un certain intervalle de temps. La valeur de confiance

d'un utilisateur dépend de son interaction avec les autres utilisateurs. TCAC introduit un mécanisme de réputation, basé sur l'évaluation de la confiance. Le modèle TCAC peut attribuer dynamiquement les rôles en fonction du comportement des utilisateurs et la situation du contexte. Quand la valeur de confiance du demandeur d'accès n'est pas inférieure au seuil de confiance prédéfinie et les informations du contexte de l'utilisateur respectent les contraintes du contexte, l'utilisateur est alors affecté à certains rôles, et peut alors exécuter les autorisations associées à ces rôles.

— **Le modèle Context-Risk-Aware Access Control** : ce modèle est basé sur le modèle RBAC [61]. Les ressources dans ce modèle sont classées dans des groupes d'objets ayant chacun un niveau de confiance minimum pour autoriser l'accès (OLoA : Object Level of Assurance). L'OloA est déterminé en se basant sur le niveau de sensibilité et l'impact d'un accès non autorisé à cet objet. Lorsque le système reçoit une requête d'accès, les informations du contexte sont évaluées pour déterminer le niveau de confiance du demandeur de l'accès (RLoA : Requester's level of Assurance). L'accès n'est autorisé que lorsque $RLoA >= OLoA$.

La valeur du RLoA est calculé en temps réel, donc il est nécessaire de designer un ensemble d'attributs du contexte qui ont un impact sur le niveau du risque d'un accès non autorisé. En effet, les facteurs qui déterminent le risque d'un accès non autorisé sont souvent : les protocoles faibles d'authentification/authentifieur (token), les emplacements d'accès qui ne sont pas dignes de confiance, les canaux de communication non protégés, le manque de fiabilité des systèmes de détection d'intrusions, etc.

Le point faible de ce type des modèles de contrôle d'accès est de trouver comment calculer et évaluer de manière fiable la valeur de confiance ou bien du risque liée à une demande d'accès.

2.4.5 Les contraintes liés à l'objectif de l'usage

Vu l'utilisation massive des technologies de l'information et des communications, un nombre croissant d'individus et d'entreprises délèguent tout ou partie de leurs données sensibles à des hébergeurs de données sur l'internet. Les données personnelles sont

parfois exploitées de façon très peu transparentes à des fins secondaires. La préservation de la vie privée (privacy) est un cas particulier de la propriété de confidentialité où les données à protéger sont à caractère personnel. Dans la littérature, la préservation de la vie privée a été définie par différents chercheurs dont les plus intéressantes sont :

- "La possibilités pour les individus, des groupes et des institutions de déterminer pour eux-mêmes, quand, comment et dans quelle mesure l'information à leur sujet est communiquée aux autres» [62].
- "La capacité d'un individu à contrôler les conditions dans lesquelles ses renseignements personnels sont acquis et utilisés" [63].

Selon l'article [64] intitulé "Privacy is linking permission to purpose", l'idée de prendre en compte l'objectif de l'utilisation (purpose) pour autoriser ou non l'accès aux données privées est très intéressante, par exemple, dans les règlements généraux, on trouve que le médecin est autorisé à accéder aux données d'un patient uniquement à certaines fins, telles que le traitement, et il est interdit d'accéder aux mêmes données à d'autres fins telles que la recherche. Donc l'objectif de l'utilisation ou le bon usage dans un modèle de contrôle d'accès garantit que les données sensibles sont utilisées uniquement dans le but pour lequel elles sont collectées et que ces données ne peuvent pas être divulguées à quelqu'un qui n'a pas besoin de les connaître. Il s'agit des principes de finalité (need to know) et de consentement. Les buts ou les objectifs d'usage sont généralement exprimées en langage naturel, ce qui est difficile à mettre en œuvre au niveau des mécanismes de contrôle d'accès. Différentes solutions ont été proposées pour intégrer le concept de l'objectif de l'usage dans le modèle RBAC :

- **Le modèle Purpose-based access control** : le modèle de contrôle d'accès à base de l'objectif d'usage [65] vise à maintenir la consistance entre la politique de protection et les pratiques déclarées. Les auteurs spécifient des invariants correspondants aux besoins de protection dans une politique de sécurité. L'objectif de représentation de ces invariants est de fournir une interprétation claire et non-ambigüe des politiques. Les objectifs d'usage sont divisés en deux classes, des buts désirés, et des buts d'accès. Les buts désirés sont liés à la donnée sensible, spécifient ainsi les usages pour lesquelles une donnée est accédée. Les buts d'accès sont liés à la requête d'accès aux données et spécifient les intentions pour lesquelles une donnée est accédée. Chaque demandeur doit déclarer explicitement le but d'accès. L'accès est permis si le but d'accès est compatible avec

le but désiré. Cette classification de buts est dérivée du travail présenté dans [11]. Au niveau de l'organisation, les objets sont organisés en utilisant des types d'objets. L'entité rôle dans ce modèle a été étendue afin de prendre en compte le rôle conditionnel, qui est basé sur les attributs du rôle et les attributs du système. En utilisant ces entités d'objets, de rôles, et de buts, les auteurs dans [65] définissent les invariants suivants :

- un objet donnée est crée seulement si nécessaire pour un rôle conditionnel de la requête en cours,

- l'autorisation du rôle conditionnel et les contraintes d'accès aux données sont principalement basés sur la compatibilité entre le but désiré et celui d'accès.

L'utilisation des rôles conditionnels et les types d'objets dans ce modèle augmentent la complexité d'administration de ces politiques.

— **Le modèle Purpose-Aware Role-Based Access Control** : le PuRBAC (Purpose-Aware Role-Based Access Control) [66] introduit le concept de l'objectif d'usage comme concept intermédiaire entre le rôle et la permission. Le modèle définit des hiérarchies de rôles et de buts. Il supporte l'expression des contraintes et des obligations, et les défini comme des conditions sur l'affectation des permissions aux différents buts. Ensuite, les buts sont assignés aux rôles. La requête d'un utilisateur est formée par un identifiant de session, un but, et une permission requise. L'autorisation peut être requise pour des buts liés au rôle actif. Il y a une autre différence majeure avec le modèle RBAC, quand une requête est soumise, il peut soit refuser l'accès, ou définir une autorisation conditionnelle. Dans ce modèle les auteurs supposent qu'il y a une relation entre la notion de l'objectif de l'utilisation dans les politiques de protection et la notion de rôle en RBAC.

— **Le modèle Privacy-aware RBAC** : Qui Ni et all. ont défini une famille de modèles Privacy-aware RBAC [12], où les politiques de protection sont exprimées à travers des affectations de permissions. Ces affectations introduisent aussi les concepts de bon usage et de condition dans les politiques d'accès. Un langage spécifique $LC0$ a été proposé afin de permettre la définition des conditions. Une permission sensible est ainsi modélisée selon des privilèges qui ont la forme générale suivante : $role \times action \times data \times purpose \times condition \times obligation$. Les auteurs ont développé aussi des algorithmes d'analyse de conflits, afin de détecter les conflits entre les différentes affectations de permissions.

— **Le modèle Purpose-based access control via workflows** : Jafari et all. [67] proposent une nouvelle approche qui consiste à utiliser les processus métiers (workflows) pour la mise en place d'un modèle de contrôle d'accès basé sur l'objectif d'utilisation. Cette approche était motivée par le fait que l'objectif de l'utilisation peut conduire à des tâches spécifiques à l'avenir, par exemple, si Alice retire de l'argent de son compte dans le but d'acheter des livres, cela implique que plus tard, elle devrait acheter des livres. En effet, le but est la raison ou l'intention de faire quelque chose. Selon la littérature philosophique sur le sens de l'intention, ils considèrent que l'intention se réfère au plan futur [68], ce plan correspond bien au processus métier défini dans les systèmes des entreprises. De plus, l'objectif de l'accès est généralement visible à une unité de niveau plus élevée qu'une action. Autrement dit, si l'utilisateur demande d'accéder à une donnée, si on considère que cette action est isolée, cela ne révèle pas le but de l'accès, mais si on prend en considération l'ensemble des actions connexes qui précèdent ou suivent cette action, on peut inférer l'objectif d'accès de cet utilisateur. Dans ce cas, l'instance d'un workflow ou les tâches à exécuter peuvent être utilisés pour déduire l'objectif de l'accès. Jafari et all. ont défini leur modèle comme suit : Soient $\mathcal{U}$ l'ensemble des utilisateurs, $\mathcal{R}$ l'ensemble des rôles, $\mathcal{X}$ l'ensemble des données, $\mathcal{C}$ l'ensemble des catégories dans lequel les données sont classées, $\mathcal{A}$ l'ensemble des actions de base dans le système, et $\mathcal{P}$ l'ensemeble des objectifs de l'utilisation défini dans l'organisation. Un système de gestion de workflows contient un ensemble de descriptions des processus métier définis dans le système $\mathcal{W} = \{W_1, ..., W_k\}$, tel que chaque processus métier se compose d'un ensemble de tâches $\mathcal{T}$ et un ensemble d'arcs $\mathcal{E} \subseteq \mathcal{T} \times \mathcal{T}$ qui dénotent une relation de précédence entre les tâches. Chaque tâche est réalisée par un acteur sur un certain nombre de ressources. Les acteurs autorisés à exécuter une tâche sont spécifiées par $TASK\_ROLES : \mathcal{T} \mapsto 2^{\mathcal{R}}$ qui associe chaque tâche à un ensemble de rôles autorisés. Les ressources d'entrée dans une tâche sont définies comme un tableau de taille variable de la forme $\langle I_1, ..., I_{n(t)} \rangle$ où $n(t)$ est le nombre d'entrées à la tâche $t \in \mathcal{T}$. Chaque I_i est un ensemble de catégories qui indique les types de données autorisés pour une i, i.e. $I_i \in 2^C$. Ils ont représenté le tableau de types de ressource d'une tâche spécifique t par $IN\_CATS(t)$ et l'ensemble des catégories de chaque entrée i de cette tableau par $IN\_CATS_i(t)$, où $1 \leq i \leq n(t)$.

Chaque ressource d'entrée dans la tâche peut être soumis à certaines actions, un tableau aussi de taille variable de la forme $\langle A_1, ..., A_{n(t)} \rangle$ est défini, tel que chaque A_i est un ensemble d'actions qui indique les actions qui peuvent etre effectuées sur l'entrée i de la tâche t ($A_i \mapsto 2^{\mathcal{A}}$). Le tableau d'actions d'une tâche spécifique t est dénoté par $IN\_ACTS(t)$ et l'ensemble des actions de chaque entrée i par $IN\_ACTS_i(t)$. Donc, la description de chaque workflow est définie comme $\langle T, E, TASK\_ROLES, IN\_CATS, IN\_ACTS \rangle$, et l'objectif d'utilisation qui correspond à chaque workflow est représenté par $PURP\_OF : \mathcal{W} \mapsto \mathcal{P}$. Chaque utilisateur $u \in U$ est affectée à un ensemble de rôles désignés par $ROLES\_OF(u)$ et chaque données $x \in X$ est associé à un ensemble de catégories de données, désigné par $CATS\_OF(x)$. Un ensemble de règles d'autorisation ($\mathcal{B} \subseteq \mathcal{R} \times \mathcal{A} \times \mathcal{C} \times \mathcal{P}$) est defini, où chaque règle est composé de quadruples $\langle R, A, C, P \rangle$, de role R, action A, catégorie des données C et l'objectif désiré P, pour lesquels l'accès est autorisé. Dans ce cas, une requête d'accès est une demande pour instancier un workflow et elle prend ainsi la forme d'un triple $\langle W, TASK\_USER, TASK\_INPUT \rangle$.

2.5 Discussion

Généralement, les droits d'accès des utilisateurs dans un système dynamique comme dans les services cloud computing peuvent être variés selon la situation présente. Il est tout particulièrement intéressant d'incorporer des contraintes temporelles dans les politiques d'autorisations. En effet, les règles dans ce type de services sont généralement sujettes à une mise à jour continue en réponse aux changements dans les exigences. Après l'étude des différents modèles de contrôle d'accès proposés, on constate que la plupart des modèles évitent de combiner les permissions et les interdictions dans une même politique d'autorisation, ils favorisent le principe d'une politique fermée, ils ne spécifient que des règles positives et ils s'intéressent à introduire des contraintes et des conditions à ces règles pour que la politique d'autorisation soit plus expressives. Cependant, ce principe rend la politique d'autorisation plus complexe à exprimer et à mettre à jour. Alors que les autres modèles comme OrBAC et ABAC permettent de définir à la fois des permissions et des interdictions, il est possible que cela peut entraîner de nombreux problèmes de cohérence et de conflits lors de l'évaluation des requêtes d'accès dans ces modèles, ce qui peut influencer négativement sur les performances

du système. Plusieurs techniques ont été proposées dans la littérature pour résoudre ces conflits et déterminer si un accès est autorisé ou non (par exemple, les interdictions l'emportent toujours sur les permissions, ou les permissions l'emportent toujours sur les interdictions, ou bien associer un niveau de propriété pour chaque règle de sécurité...). Mais en utilisant ces techniques, il est difficile de vérifier si la décision prise est correcte ou non surtout si le nombre de règles dans la politique est important. Une représentation en logique classique n'est pas adaptée car elle donne rapidement des incohérences.
Récemment, des chercheurs commencent à se pencher sur les logiques non monotones [69, 70] pour introduire la propriété par défaut dans les politiques d'autorisation afin d'exprimer les règles par défaut qui peuvent être par la suite sujette à l'exception. Ces logiques possèdent la particularité que lors de l'ajout d'une prémisse, cela peut nous conduire à retirer une conclusion tirée précédemment. Les formalismes logiques non monotones permettent donc de représenter les règles de sécurité sous forme prototypique et de gérer implicitement les conflits et déduire la décision d'accès selon la situation présente. Néanmoins, ces formalismes ne sont pas assez expressifs pour représenter des structures complexes des règles d'autorisation sous forme prototypique. Par exemple, une règle de sécurité qui stipule qu'il est permis d'exécuter le service X du dimanche au jeudi, sauf les jours fériés et les périodes de fêtes ou bien une règle de sécurité basée sur l'objectif d'utilisation sous forme d'une séquence d'action qui peut inclure des actions facultatives qui peuvent être annulées dans certaines situations spécifiques.

2.6 Conclusion

Nous avons vu à travers ce chapitre que plusieurs modèles de contrôle d'accès ont été proposés dans la littérature pour répondre aux exigences de la gestion des droits d'accès dans un système. Chaque modèle tente de résoudre certains des inconvénients des modèles qui l'ont précédé. Il est néanmoins difficile de conclure qu'un modèle de contrôle d'accès est meilleur qu'un autre. Cette décision dépendra totalement des besoins de la sécurité du système concerné.
Afin de proposer et mettre en œuvre un modèle de contrôle d'accès contextuelle et flexible adéquat pour les données sensible dans les services de cloud, il nous semble indispensable d'avoir un formalisme qui, d'une part, permet de représenter explicitement les connaissance temporelles afin de nous permettre de décrire formellement une

politique de sécurité contextuelle, et d'autre part, il supporte le raisonnement non monotone afin de nous permettre de déduire automatiquement les actions autorisé à effectuer par les utilisateurs qui demandent l'accès aux données sensibles selon le contexte de la requête.

Dans le chapitre suivant, nous allons présenter les principaux langages formelles qui ont été développés pour représenter et raisonner avec des connaissances temporelles pour choisir le formalisme le plus approprié à spécifier une politique de sécurité flexible afin d' assurer la confidentialité des données sensibles sauvegardées dans le cloud.

CHAPITRE 3
APPROCHES LOGIQUES POUR LA REPRÉSENTATION DE CONNAISSANCES

3.1 Introduction

Les logiques de description (Description Logics (DLs)) [71] forment une famille de langages de représentation de connaissance qui sont utilisées pour représenter les connaissance terminologiques d'un domaine d'application d'une façon structurée et formelle, par l'utilisation des descriptions basé sur les notions de concepts, de rôles et d'individus, tout en leurs associant une sémantique basée sur une logique formelle et cela pour qu'elles puissent être traitées et manipulées par les machines. Ces langages disposent de mécanismes d'inférences efficaces qui permettent de déduire des connaissances implicites à partir des connaissances explicites. Nombreux travaux ont proposé d'étendre ces logiques afin de représenter et de manipuler des notions plus complexes. Dans ce chapitre, nous introduisons de manière générale les logiques de descriptions en se basant sur leur expressivité et leurs mécanismes de raisonnement. Après, nous étudions les formalismes mis au point pour représenter et raisonner sur les connaissances temporelles. Nous terminerons par étudier les différents travaux existants dans la littérature qui ont proposé d'étendre les logiques de description par la dimension temporelle.

3.2 Les logiques de description

Un système basé sur une logique de description (voir la Figure 3.1 [72]) est généralement composé d'une base de connaissances et d'un moteur d'inférence qui implante les services d'inférence. Une base de connaissances est classé en deux niveaux : niveau terminologique, appelé TBox (Terminological BOX), et un niveau factuel, appelé ABox (Assertionnel BOX).

— **Niveau terminologique (TBox) :** introduit le vocabulaire d'un domaine d'application, il comprend la définition des concepts et des rôles. Les concepts sont des définitions intensives représentant un ensemble d'individus. Les rôles décrivent des relations binaires entre les concepts. Les concepts et les rôles dans une

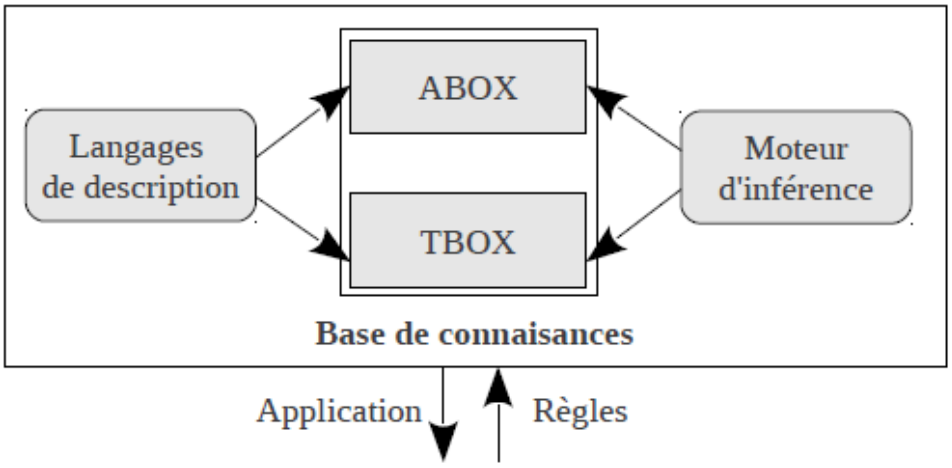

Figure 3.1 – Structure générale d'un système basé sur une logique de description

logique de description (LD) sont définies à partir d'un ensemble des noms de concept N_C, d'un ensemble des noms de rôle N_R en utilisant certain ensemble de constructeurs que ce langage possède, chaque logique a son ensemble de constructeurs.

Une TBox $\mathcal{T}$ est un ensemble fini de définitions de concept sous forme d'un axiome terminologique qui peut être écrit $A \sqsubseteq C$ ou bien $A \equiv C$ où $A \in N_C$ et C est une description de concept. Un nom de concept A est appelé nom défini s'il apparaît une seule fois du côté gauche dans toutes les définitions de concept du $\mathcal{T}$, sinon il est appelé nom primitif.

— **Niveau factuel (ABox) :** permet de décrire les individus du domaine (en leur donnant des noms) qui sont des instances des concepts et des rôles en utilisant des assertions. Les assertions peuvent être sous la forme :
  — $C(I)$, où C est un nom de concept et I un nom d'un individu, désigne une instanciation de concept C.
  — $r(I_1, I_2)$, permet d'indiquer que l'individu I_1 est en relation avec l'individu I_2 par le rôle r.

  Une ABox $\mathcal{A}$ est définie comme un ensemble fini de telles assertions.

3.3 Syntaxe et sémantique formelle

Comme nous l'avons déjà souligné, les différentes LDs se distinguent par les constructeurs spécifiques qu'elles proposent. Les langages de (Frame Logic) $\mathcal{FL}^-$ [73] et $\mathcal{FL}$ [74]

Constructeur	Syntaxe	Langage			
Concept	A	FL0	FL-	AL	S
Rôle	R				
Intersection	$C \cap D$				
Restriction de valeurs	∀R.C				
Quantification existentielle	∃R				
Universel	T				
Vide	⊥				
Négation atomique	¬A				
Négation	¬C	C			
Union	$C \cup D$	U			
Restriction existentielle	∃R.C	E			
Restriction numérique	$(\geq n\ R), (\leq n\ R)$	N			
Nominaux	$\{a_1, ..., a_n\}$	O			
Hiérarchie de rôles	R ⊆ S	H			
Rôle inverse	R ⁻	I			
Restriction numérique qualifiée	$(\geq n\ R.C), (\leq n\ R.C)$	Q			

Figure 3.2 – Constructeurs des logiques de description

sont les premiers langages pour lesquels ont été établis les premiers résultats théoriques sur les logiques de descriptions. Le langage $\mathcal{AL}$ (Attributive Langage) a été introduit dans [75] comme un langage minimal ayant un intérêt pratique. Ce dernier est ensuite enrichie progressivement par de nouveaux constructeurs [76] pour avoir des langages de plus en plus expressifs (on dit qu'un langage $\mathcal{A}$ est plus expressif qu'un langage $\mathcal{B}$ si tout concept exprimable en $\mathcal{B}$ est aussi exprimable en $\mathcal{A}$ mais pas l'inverse). Les noms de ces langages sont identifiés par les lettres associées à chaque ensemble des constructeurs ajoutés, i.e. si on ajoute la négation complète repérée par la lettre $\mathcal{C}$, à la logique $\mathcal{AL}$ on obtient la logique nommé $\mathcal{ALC}$. La Figure 3.2 présente les constructeurs les plus courants avec leur nom de langage [77].

Parmi les constructeurs les plus utilisés, on trouve

— Les deux constantes ($\top$) dénote le concept le plus général, son extension inclut tous les instances possibles et ($\bot$) dénote le concept le plus spécifique, son extension est l'ensemble vide.

— Le constructeur de la conjonction ($\sqcap$) permet de définir qu'un concept est construit à partir d'une intersection des concepts.

— Le constructeur de la négation primitive ($\neg A$) correspond à la négation qui porte que sur les concepts primitifs, alors que la négation complète ($\neg C$) est étendue aux concepts définis.

— Le constructeur de restriction universelle ($\forall r.C$) qui précise le co-domaine du rôle r, autrement dit, les individus dont toutes les relations de type r se font avec des

individus de type C.

— Le constructeur de la restriction de cardinalité supérieure ($\geq nr$) (resp inférieure ($\leq nr$)) précise qu'il existe au plus (resp. au moins) n individus dans le co-domaine de rôle r.

— La restriction existentielle ($\exists r.C$) affirme l'existence d'au moins un couple d'individus en relation par l'intermédiaire de rôle r.

— Les nominaux $\{I_1, ..., I_n\}$, où $I_1, ..., I_n$ sont des individus, utilisé pour définir un domaine de valeurs à la façon d'un type énuméré, en donnant la liste exhaustive des instances qui constituent le domaine de valeurs élémentaires possibles de ce concept.

La sémantique de LD est définie par les interprétations. Une interprétation I est un couple $(\Delta_I, .^I)$ où Δ_I est un ensemble non vide, appelé domaine d'interprétation, $.^I$ est une fonction d'interprétation. Cette fonction associe :

— un concept C à un sous ensemble $C^I \subseteq \Delta_I$

— un role r à un sous ensemble $r^I \subseteq \Delta_I \times \Delta_I$

— un nom d'un individu I_1 à un element ${I_1}^I \in C^I$

— une assertion de rôle $r(I_1, I_2)$ à un élément de $({I_1}^I, {I_2}^I) \in r^I$, on suppose que des noms d'individus distincts dénotent des objets différents. Par conséquent, l'interprétation doit satisfaire l'assomption de nom unique UNA (Unique Name Assumption), i.e, si I_1 et I_2 sont des noms différents alors ${I_1}^I \neq {I_2}^I$.

La sémantique est étendue aux différents constructeurs, elle est présenté par le Tableau 4.1.

Constructeur	Syntaxe	Interprétation
Concept universel	$\top$	Δ_I
Concept vide	$\bot$	$\varnothing$
Concept primitive	A	$A^I \subseteq \Delta_I$
Négation atomique	$\neg A$	$\Delta_I - A^I$
Négation complète	$\neg C$	$\Delta_I - C^I$
Restriction de cardinalité	$\geq nr$	$\{x \in \Delta_I, \mid y, r(x,y) \in r^I \mid \geq n\}$
Restriction de cardinalité	$\leq nr$	$\{x \in \Delta_I, \mid y, r(x,y) \in r^I \mid \leq n\}$
Conjonction de concept	$C \sqcap D$	$C^I \cap D^I$
Disjonction de concept	$C \sqcup D$	$C^I \cup D^I$
Quantificateur universel	$\forall r.C$	$\{x \in \Delta_I \mid \forall y : (x,y) \in r^I \rightarrow y \in C^I\}$
Quantificateur existentiel	$\exists r.C$	$\{x \in \Delta_I \mid \exists y : (x,y) \in r^I \wedge y \in C^I\}$
Nominaux	$\{I_1, ..., I_n\}$	$\{I_1^I, ..., I_n^I\}$

Tableau 3.1 – La sémantique des constructeurs des logiques de description

Si une interprétation I satisfait un axiome alors I est appelé modèle de cet axiome. Deux axiomes ou deux ensembles d'axiomes sont équivalents s'ils ont les mêmes modèles.

L'interprétation I est un modèle d'une TBox $\mathcal{T}$ si elle satisfait chacun des axiomes de $\mathcal{T}$. La sémantique d'une TBox est définie par l'ensemble de ses modèles. Deux TBox sont donc équivalents s'ils ont les mêmes modèles.

L'interprétation I satisfait une ABox $\mathcal{A}$ si I satisfait toute assertion dans $\mathcal{A}$, dans ce cas, I est un modèle de $\mathcal{A}$.

L'interprétation I est un modèle d'une base de connaissance $\mathcal{K}$ = ($\mathcal{T}$, $\mathcal{A}$) si I est un modèle de $\mathcal{T}$ et de $\mathcal{A}$.

3.4 Les services d'inférences standards

Par ailleurs, les systèmes de LDs offrent des capacités de raisonnement. Le raisonnement permet d'obtenir les connaissances qui ne sont pas représentées explicitement dans la base de connaissances. Il existe deux classes de services d'inférences :

3.4.1 Inférences élémentaires sur TBOX

Les différents services d'inférences qui peuvent être effectués sur une TBox $\mathcal{T}$ sont :

Subsomption La détermination de subsomption entre deux concepts C et D, typiquement écrit $C \sqsubseteq D$ (D subsume C), permet de vérifier si un concept D (le subsumer) est plus général que C (le subsumé). Formellement, un concept C est subsumé par un concept D par rapport à $\mathcal{T}$ si $C^I \subseteq D^I$ pour tout modèle I de $\mathcal{T}$, dans ce cas on écrit $C \sqsubseteq_{\mathcal{T}} D$ ou $\mathcal{T} \vDash C \sqsubseteq D$.

L'application de la relation de subsumption sur tous les couples possible de concepts dans la TBOX, permet de faire une classification des concepts (calcul de la taxonomie et des hiérarchies). Cette tâche détermine l'emplacement d'un concept par rapport aux autres concepts dans la hiérarchie en fonction de leur généralité. Cette hiérarchie fournie l'information utile pour la connexion entre différentes concepts, et elle peut être utilisée pour accélérer les services d'inférences.

Satisfiabilité Un concept C est satisfaisable par rapport à $\mathcal{T}$ s'il existe au moins un modèle I de $\mathcal{T}$ tel que C^I n'est pas vide.

Equivalence Deux concepts C et D sont équivalent par rapport à $\mathcal{T}$ si $C^I = D^I$ pour tout modèle I de $\mathcal{T}$, dans ce cas on écrit $C \equiv_{\mathcal{T}} D$ ou $\mathcal{T} \vDash C \equiv D$.

Disjonction Deux concepts C et D sont disjoints par rapport à $\mathfrak{T}$ si $C^I \cap D^I = \varnothing$ pour tout modèle I de $\mathfrak{T}$.

3.4.2 Réduction de tests

Il y a une relation étroite entre la subsomption et la satisfiabilité, pour cela il existe des systèmes LD qui considère la subsumption des concepts comme le service d'inférence de base, tandis que d'autres, considèrent la satisfiabilité. Selon l'expressivité d'un langage LD (supporte ou non la négation complète), on peut réduire les services d'inférences présenté précédemment au test de subsomption ou bien au test de satisfiabilité :

- **Au test de subsomption**
  - C n'est pas satisfaisable si seulement si C est subsumé par $\perp$.
  - C et D sont équivalents si seulement si C est subsumé par D et D est subsumé par C.
  - C et D sont incompatibles si seulement si (C $\sqcap$ D) est subsumé par $\perp$.
- **Au test de satisfiabilité**
  - C est subsumé par D si seulement si $(C \sqcap \neg D)$ n'est pas satisfaisable.
  - C et D sont équivalents si seulement si $(C \sqcap \neg D) \sqcup (D \sqcap \neg C)$ n'est pas satisfaisable.
  - C et D sont incompatibles si seulement si $(C \sqcap D)$ n'est pas satisfaisable.

3.4.3 Inférences élémentaires sur ABOX

Les différents services d'inférences qui peuvent être effectuées sur une ABox $\mathcal{A}$ sont :

- **La vérification d'instance** : consiste à vérifier si un individu dans $\mathcal{A}$ est une instance d'un concept donné.
- **La recherche d'instance** : trouver toutes les instances dans $\mathcal{A}$ qui appartiennent au concept donné.
- **La reconnaissance d'instance** : trouver pour un individu donné, les concepts (notamment le ou les plus spécifiques) dont il est une instance.

3.5 Les algorithmes de raisonnement

Dans LD, un algorithme de raisonnement est une procédure de décision pour un problème d'inférence, qui vérifie les 3 propriétés suivantes :

Arrêt l'algorithme doit donner un résultat à un temps fini.

Correction les inférences produites sont en accord avec la sémantique associée.

Complétude toutes les formules valides peuvent être démontrées sur le plan syntaxique.

Comme nous l'avons déjà mentionné, les moteurs d'inférence dans les systèmes LD n'ont besoin que d'un seul algorithme d'inférence pour raisonner au niveau terminologique. Selon la technique utilisée, on distingue les algorithmes de subsomption de type normalisation-comparaison(NC), des fois appelé des algorithmes de subsomption structurelle et les algorithmes de vérification de satisfiabilité à base de tableaux. L'idée principale de ces algorithmes peut être décrite comme suit :

3.5.1 Raisonnement basée sur la comparaison structurelle

Le principe de ces algorithmes consiste à comparer les structures syntactiques de deux concepts. Cet algorithme passe par deux étapes :

La normalisation les descriptions de concepts sont réécrites dans une forme normale dans laquelle les parties comparables entre les descriptions de concept de chaque description de concept sont regroupées. Cette transformation préserve la subsumption car les concepts réécrits sont équivalents logiquement avec les précédents.

La comparaison des structures syntaxiques des formes normales la subsomption entre les descriptions de concept est récursivement vérifiée en comparant les formes normales correspondantes.

Exemple 1. *On présente cette approche dans le cadre du langage simple* FL_0 *:*
La description d'un concept FL_0 *est sous forme normale si et seulement si elle est de la forme,* $A_1 \sqcap ... \sqcap A_m \sqcap \forall r_1.C_1 \sqcap ... \sqcap \forall r_n.C_n$*, où* $A_1...A_m$ *sont des concepts de noms distincts,* $r_1...r_n$ *sont des noms de rôles distincts, et* $C_1...C_n$ *sont des descriptions de concepts* FL_0 *sous forme normale.*
Il est a noter que toute description peut être transformée en une forme normale en utilisant l'idempotence, la commutativité, et l'associativité de $\sqcap$*.*

- *Idempotence :* $C \sqcap C = C$
- *Commutativité :* $C \sqcap D = D \sqcap C$
- *Associativité :* $C \sqcap (D \sqcap E) = (C \sqcap D) \sqcap E$
- $\forall r.C \sqcap \forall r.D = \forall r.(C \sqcap D)$
- *Si* $C \sqsubseteq D$ *et* $C \sqsubseteq E$ *alors* $C \sqsubseteq D \sqcap E$
- *Si* $C \sqsubseteq D$ *alors* $C \sqcap X \sqsubseteq D$ *pour tout description X*

Soient C,D deux concepts FL_0 *sous forme normale, où C =* $A_1 \sqcap ... \sqcap A_m \sqcap \forall r_1.C_1 \sqcap ... \sqcap \forall r_n.C_n$ *et D =* $B_1 \sqcap ... \sqcap B_k \sqcap s_1.D_1 \sqcap ... \sqcap s_l.D_l$*,* $C \sqsubseteq D$ *si et seulement si les deux conditions suivantes sont vérifiées :*

1. pour tout i, $1 \leq i \leq k$*, il existe j,* $1 \leq j \leq m$ *tel que* $A_j \sqsubseteq B_i$

2. pour tout i, $1 \leq i \leq l$*, il existe j,* $1 \leq j \leq n$ *tel que* $s_j = r_i$ *et* $C_j \sqsubseteq D_i$

Cette caractérisation de subsomption est correcte et complète. Elle permet de définir un algorithme récursif polynomial [74].

Soient quatre concepts A, B, C et D où $A \sqsubseteq B$ *et on veut montrer que :*

$A \sqcap \forall r.C \sqcap \forall r.D \sqsubseteq B \sqcap \forall r.C$

On doit procéder comme suit :

1. ***Normalisation*** *:* $A \sqcap \forall r.C \sqcap \forall r.D \equiv A \sqcap \forall r.(C \sqcap D)$ *(factorisation des co-domaines de r). Donc ça revient à montrer que :* $A \sqcap \forall r.(C \sqcap D) \sqsubseteq B \sqcap \forall r.C$
2. ***Comparaison structurelle*** *: on vérifie que pour chaque sous expression X dans le concept (*$B \sqcap \forall r.C$*) s'il existe une sous expression Y dans l'autre concept (*$A \sqcap \forall r.(C \sqcap D)$*) tel que* $Y \sqsubseteq X$*, on met :*
   - *X = B, Y = A est verifié (*$A \sqsubseteq B$*, selon l'énoncé).*
   - *X =* $\forall$ *r.C, Y =* $\forall r.(C \sqcap D)$*, pour qu'il soit égal, il faut montrer que* $C \sqcap D \sqsubseteq C$*.*
   - *On applique le même algorithme : X = C, Y =* $C \sqcap D$ *est verifié parce que* $C \sqsubseteq C$*, et* $C \sqcap Z \sqsubseteq C$ *pour tout description Z. on deduit que X =* $\forall$ *r.C, Y =* $\forall r.(C \sqcap D)$ *est verifié.*
   - *On conclut donc que* $A \sqcap \forall r.C \sqcap \forall r.D \sqsubseteq B \sqcap \forall r.C$ *est vrai.*

Les algorithmes de subsomption structurelle sont facile à mettre en œuvre et sont efficaces pour les logiques de descriptions simples avec une moindre expressivité mais ils sont généralement incomplets, ils peuvent ne pas détecter toutes les relations valides de subsumption pour les logiques de descriptions expressives, en particulier, les logiques de descriptions avec la disjonction, la négation complète, etc. Le problème réside dans

la difficulté de définir une forme normale appropriée pour ces logiques expressives [78].

3.5.2 Raisonnement basé sur l'algorithme de tableaux

L'idée de cette approche est fondée sur la méthode spécialisée du calcul des tableaux sémantiques en logique classique. La méthode des tableaux sémantiques est une méthode de réfutation analytique [79], i,e. pour prouver une formule ϕ, il est supposé que ϕ est fausse et les conséquences de cette hypothèse sur les constituants de ϕ sont examinées. La méthode s'appuie sur un ensemble de règles de décomposition qui permettent d'assigner une valeur de vérité aux constituants de la formule ϕ, une fois fixée la valeur de vérité de la formule ϕ elle-même. Le premier algorithme de tableaux

Règle	Condition	Action
$\rightarrow \sqcap$	A contient $C_1 \sqcap C_2(X)$, $C_1(X)$ et $C_2(X)$ ne sont pas dans A	A' = $A \cup \{C_1(X), C_2(X)\}$
$\rightarrow \sqcup$	A contient $C_1 \sqcup C_2(X)$, $C_1(X)$ et $C_2(X)$ ne sont pas dans A	A' = $A \cup \{C_1(X)\}$ ou A'' = $A \cup \{C_2(X)\}$
$\rightarrow \forall$	A contient $\forall r C(X)$ et $r(X,Y)$ mais $C(Y) \notin A$	A' = $A \cup \{C(Y)\}$
$\rightarrow \exists$	A contient $\exists$ r C (X) et $\nexists$ y tq. r(X,Y) et C(Y) soient tous deux dans A	A' = $A \cup \{r(X,Z), C(Z)\}$ où Z est une nouvelle variable

Tableau 3.2 – Les règles de décomposition pour ALC

dans les LDs a été présenté par Schmidt-Schauß et Smolka [75] pour la satisfiabilité des concepts $\mathcal{ALC}$, et ce travail a montré de bon résultat sur la complexité des inférences standards. Depuis, cette approche a été utilisée pour obtenir des algorithmes de satisfiabilité correctes et complètes pour les différents LDs expressif qui étendent $\mathcal{ALC}$. En effet, la plupart des raisonneurs actuels FaCT (Fast Classification of Terminologies), FaCT++ (version en C++), Pellet et Racer (Renamed ABox and Concept Expression Reasoner) implémentent cet algorithme, tout en intégrant des optimisations sophistiquées, car leurs mises en œuvre naïves ne parviennent pas à être pratiques.

De manière générale, un tableau sémantique dans LD est une procédure qui permet de construire une interprétation qui satisfait l'assertion d'un concept donné. Les dérivations peuvent être établies par l'application d'un ensemble de règles de transformation ou de décompositions qui correspondent aux constructeurs dans le langage LD utilisés, elles peuvent être déterministe ou non déterministe, le Tableau 3.2 représente un exemple de règles de décomposition pour ALC. Plus formellement, soit C un concept sous forme normale négative (NNF), i.e., la négation se trouve uniquement devant les

concepts primitifs. Afin de tester la satisfiabilité de ce dernier, l'algorithme de tableaux commence par une ABox avec une assertion de la forme $A_0 = C(I)$ et applique des règles de décomposition qui préservent la consistance jusqu'à ce qu'aucune règle ne puisse être appliquée. Si l'ABox obtenue à la fin ne contient pas de contradiction, alors A_0 est consistante et donc C est satisfiable sinon elle est non consistante et dans ce cas C est non satisfiable. On illustre ce principe via l'exemple suivant dans le cadre de la logique $\mathcal{ALCN}$:

Exemple 2. *Soient A et B deux noms de concepts, r un nom de rôle et on veut savoir si :* $(\exists r.A) \sqcap (\exists r.B) \sqsubseteq \exists r.(A \sqcap B)$

Ceci revient à tester la non satisfiabilité de $C = (\exists r.A) \sqcap (\exists r.B) \sqcap \neg(\exists r.(A \sqcap B))$

Le concept C doit être en NNF. En utilisant les règles du processus de normalisation suivantes :

$$\begin{aligned}
&\neg\top \rightarrow \bot \\
&\neg\bot \rightarrow \top \\
&\neg(C \sqcap D) \rightarrow \neg C \sqcup \neg D \\
&\neg(C \sqcup D) \rightarrow \neg C \sqcap \neg D \\
&\neg(\neg C) \rightarrow C \\
&\neg(\forall r.C) \rightarrow \exists r.\neg C \\
&\neg(\exists r.C) \rightarrow \forall r.\neg C \\
&\neg(\geq nr) \rightarrow \forall r.\bot \text{ si } n = 1 \\
&\neg(\geq nr) \rightarrow (\leq n-1\ r) \text{ si } n > 1 \\
&\neg(\leq nr) \rightarrow (\geq n+1\ r)
\end{aligned}$$

On obtient, $C = (\exists r.A) \sqcap (\exists r.B) \sqcap \forall r.(\neg A \sqcup \neg B))$.

Ensuite, on essaye de construire une interprétation finie I telle que $C^I \neq \varnothing$. Autrement dit, vérifier s'il existe un individu de Δ_I qui soit un élément de C^I.

Soit I_1 un tel élément et donc I_1 doit satisfaire les contraintes suivante :

$I_1 \in (\exists r.A)^I$ *,* $I_1 \in (\exists r.B)^I$ *et* $I_1 \in (\forall r.(\neg A \sqcup \neg B)^I$.

De $I_1 \in (\exists r.A)^I$, on déduit qu'il doit exister un élément I_2 tel que $(I_1, I_2) \in r^I$ et $I_2 \in A^I$.

Il est de même, $I_1 \in (\exists r.B)^I$, on déduit qu'il doit exister un élément I_3 tel que $(I_1, I_3) \in r^I$ et $I_3 \in B^I$.

Pour toute restriction existentielle, on introduit un nouvel individu comme remplisseur du rôle. En plus, cet individu doit satisfaire les contraintes correspondantes.

On a déjà introduit I_1, I_2 et I_3, ces individus doivent satisfaire les contraintes suivantes :

- $I_1 \in (\forall r.(\neg A \sqcup \neg B)^I$
- $I_2 \in (\neg A \sqcup \neg B)^I$

— $I_3 \in (\neg A \cup \neg B)^I$

Si $I_2 \in (\neg A \sqcup \neg B)^I$ *signifié que* $I_2 \in (\neg A)^I$ *ou* $I_2 \in (\neg B)^I$ *.*

Toutefois, $I_2 \in (\neg A)^I$ *contredit l'autre contrainte* $I_2 \in (A)^I$*, si bien qu'on doit opter pour* $I_2 \in (\neg B)^I$*. D'une manière similaire, on prend* $I_3 \in (\neg A)^I$

Pour les contraintes disjonctives, on choisit les possibilités successivement en faisant des retours arrière à chaque fois qu'une contradiction est découverte. Donc là on vient de satisfaire toutes les contraintes en générant une interprétation I *telle que :*

— $\Delta_I = \{I_1, I_2, I_3\}$,

— $r^I = \{(I_1, I_2), (I_1, I_3)\}$,

— $A^I = \{I_2\}$,

— $B^I = \{I_3\}$.

On déduit que C est satisfiable et donc $(\exists r.A) \sqcap (\exists r.B) \not\equiv \exists r.(A \sqcap B)$*.*

3.6 Les logiques temporelles

Aujourd'hui, plusieurs applications dans le domaine d'intelligence artificielle exigent de représenter et de raisonner sur les connaissances temporelles. Vu l'aspect multi-forme du temps, on trouve des logiques temporelles de toutes natures dans la littérature. Nous en distinguons trois catégories [80, 81] : la logique classique du premier ordre, la logique modale temporelle, et la logique réifiée.

3.6.1 La logique temporelle classique

Dans cette logique, le temps est introduit aux prédicats du premier ordre comme un argument qui représente l'information temporelle durant lesquels ces prédicats doivent être interprétés. Cette représentation ne donne aucun statut particulier au temps. Elle ne permet pas de réaliser des inférences ou de déduire des propriétés liées au temps tel que la transitivité de la relation de précédence. La méthode TA des arguments temporels (Temporal Arguments) [82] est un cas particulier de cette logique, où les atomes sont de la forme $P(t, x_1, ..., x_n)$ avec t qui désigne un terme temporel, et les x_i sont des termes non-temporels. Les différentes entités temporelles sont représentés soit par des instants, soit par des termes fonctionnels qui correspondent aux différentes formes d'intervalles : $closed(t_1, t_2)$, $open(t_1, t_2)$, etc. Certains prédicats prennent en argument plus d'un terme temporel (< et =). Un prédicat particulier $Exists(x)$ décrit l'existence de l'individu dénoté par x. La sémantique de ce langage est donnée par des structures

d'interprétation de la forme (W, O, D, E), où W est un ensemble de points de temps, O une relation d'ordre sur W, D est le domaine d'interprétation des termes non temporels, et E est une relation entre des sous ensembles de W et des éléments de D. La relation E matérialise les périodes d'existence des éléments de D. La fonction d'interprétation V réalise alors classiquement le lien entre termes temporels et éléments de W, entre termes non-temporels et éléments de D, entre le symbole $<$ et la relation O, entre le prédicat $Exists()$ et la relation E, et enfin associe à un prédicat P un ensemble de n-uplet $(t, x_i, ..., x_n)$ pris dans $W \times D \times ... \times D$. Cette approche malgré ces défauts reste fiable pour le raisonnement temporel.

3.6.2 Les logiques modales temporelles

Dans cette logique, le temps est représenté comme une modalité associée au domaine d'interprétation d'une formule et sa manipulation se fait via des règles d'inférence de la logique [83]. Le langage utilisé est une extension du langage propositionnel ou des prédicats avec les opérateurs suivants :
Soit ϕ une formule propositionnelle :

$F\phi$: ϕ sera vraie dans le futur.
$P\phi$: ϕ a été vraie dans le passé.
$G\phi$: ϕ sera toujours vraie dans le futur.
$H\phi$: ϕ a toujours été vraie dans le passé.

Le lien entre ces modalités se fait avec les opérateurs de possibilité ($\Diamond$) et de nécessité ($\Box$) :

— suivant la conception Diodoréenne, ce qui est nécessaire est vrai maintenant et le restera dans l'avenir :

$$\Box\phi \equiv \phi \wedge G\phi$$
$$\Diamond\phi \equiv \phi \vee F\phi$$

— suivant la conception Aritotélicienne, ce qui est nécessaire a été, est, et sera toujours vrai :

$$\Box\phi \equiv H\phi \wedge \phi \wedge G\phi$$
$$\Diamond\phi \equiv P\phi \vee \phi \vee F\phi$$

L'interprétation des formules de logiques modales se fait par rapport à des structure, dites de Kripke, qui définissent un ensemble d'univers ou de mondes, une relation entre

ces mondes, dite relation d'accessibilité, et ce qu'il faut pour interpréter dans chaque monde une formule sans opérateur modal, en général ce sont des interprétations classiques. Dans le cas de la logique temporelle modale, les mondes représentent différents états de l'univers à différents moments (instants ou intervalles), et la relation d'accessibilité correspond à la relation de précédence entre les moments. La logique modale temporelle a donné lieu à des développements dans différents domaines tel que la preuve de programmes ou la vérification de circuits logiques.

3.6.3 Les logiques réifiées

Contrairement aux approches présentées précédemment, les logiques temporelles réifiées considère le temps comme une entité primitive du langage. Le principe général d'une logique réifiée consiste de passer à partir d'un langage propositionnel ou de prédicat à un métalangage, où une formule du langage initial devient un simple terme dans le nouveau langage. De ce fait, il est possible de raisonner sur des aspects particuliers de la validité des expressions du langage initial à travers l'utilisation du prédicat de vérité $TRUE$.

Dans le cas du raisonnement temporel, le prédicat $TRUE$ prend comme argument une assertion logique atemporelle (une formule du langage propositionnel ou de premier ordre) et sa référence temporelle. Cette dernière consiste en un ensemble d'entités temporelles reliées entre elles par un ensemble de relations temporelles.

$$TRUE(\ formule\ atemporelle, qualification\ temporelle).$$

La logique réifiée permet donc d'accorder un statut spécial au temps et d'offrir une puissance d'expression des différents aspects temporels. Nous pouvons par exemple :

— représenter l'axiome d'homogénéité du prédicat $TRUE$ qui stipule qu'une proposition p est vraie sur un intervalle T donné si et seulement si elle est vraie sur tous ses sous intervalles, de la manière suivante :

$$TRUE(p,T) \Leftrightarrow \forall\ t\ in(t,T) \Rightarrow TRUE(p,t)$$

p étant une proposition, t et T des intervalles et in un prédicat exprimant le fait qu'un intervalle est contenu dans un autre intervalle,

— exprimer des faits incompatibles, par exemple :

$$\forall f_1, f_2, HOLDS(f_1, time_1) \wedge HOLDS(f_2, time_2) \wedge INCOMPATIBLE(f_1, f_2) \Rightarrow \neg overlap(time_1, time_2)$$

Cet axiome stipule que deux faits incompatibles (f_1, f_2) ne peuvent se chevaucher.

— exprimer la causalité : la propriété de causalité représente une connaissance très importante dans plusieurs domaines d'intelligence artificielle tel que le raisonnement qualitatif. Pour exprimer, la règle générale de causalité suivante : "Les effets ne peuvent pas précéder leur causes", nous écrivons :

$$\forall event, fact, time_1, time_2\ CAUSES(event, time_1, fact, time_2) \Rightarrow time_1 < time_2.$$

Suivant les travaux, la représentation primitive du temps se fait, soit par l'instant (la logique de McDermott), soit par l'intervalle (la logique d'Allen) et on peut aussi avoir les deux types d'entités.

— **La logique de McDermott** : cette logique est basé sur la notion d'état défini comme une photographie instantanée de l'univers à une date précise, McDermott représente le temps par un ensemble infini et dense d'instants, c'est à dire qu'entre deux états s_1 et s_2 tels que $s_1 < s_2$ il existe toujours un autre état s_3 tel que $s_1 < s_3 < s_2$. L'idée étant de capturer l'aspect continu du temps. Les seules relations possibles sur une ramification de la ligne temporelle sont les relations de précédence (<, >, =). Pour représenter les histoires possibles de l'univers, McDermott définit la notion de chronique comme un ensemble complètement ordonné d'états qui s'étend infiniment dans le temps. Un fait, qui intuitivement correspond à l'assertion de la vérité d'une proposition à un instant ou durant une période, est décrit dans cette logique comme un ensemble d'états : cet ensemble est celui où la proposition est vraie. Un événement, dont l'objet est de représenter des actions sur l'univers, est défini comme un ensemble d'intervalles d'états. Ceci permet d'assimiler l'événement avec ses intervalles d'occurrence, sachant qu'un intervalle est toujours un sous-ensemble d'une chronique. Un événement a donc comme caractéristique de « prendre du temps », ce qui le distingue de la notion équivalente dans le calcul de situations par exemple, où un événement est un changement instantané entre deux situations.

— **La logique d'Allen** : contrairement à l'approche précédente, Allen [6] considère l'intervalle comme la notion appropriée pour simuler de la manière dont l'être humain perçoit le temps dans son raisonnement.

Un intervalle temporel est une période de temps débutant à un certain instant ou point temporel et se terminant à un instant ultérieur, avec le premier point inférieur

Relation de base	Symbole	Relations entre les points d'extrémité
X before Y	b	$X^- < Y^-, X^- < Y^+,$
Y after X	a	$X^+ < Y^-, X^+ < Y^+$
X meets Y	m	$X^- < Y^-, X^- < Y^+,$
Y met-by X	mi	$X^+ = Y^-, X^+ < Y^+$
X overlaps Y	o	$X^- < Y^-, X^- < Y^+,$
Y overlapped-by X	oi	$X^+ > Y^-, X^+ < Y^+$
X during Y	d	$X^- > Y^-, X^- < Y^+,$
Y includes X	di	$X^+ > Y^-, X^+ < Y^+$
X starts Y	s	$X^- = Y^-, X^- < Y^+,$
Y started-by X	si	$X^+ > Y^-, X^+ < Y^+$
X finishes Y	f	$X^- > Y^-, X^- < Y^+,$
Y finished-by X	fi	$X^+ > Y^-, X^+ = Y^+$
X equals Y	=	$X^- = Y^-, X^- < Y^+,$
		$X^+ > Y^-, X^+ = Y^+$

Tableau 3.3 – Les relations entre les extrémités des relations temporelles de base.

au second. Un intervalle peut être éventuellement décomposé en sous-périodes. Les propriétés, les événements et les processus sont des entités temporelles fondamentales dans cette logique. Toutes sont associées à un intervalle. La différence entre ces trois entités réside dans leur propriété de "divisibilité". Une propriété, notée $HOLD(p,t)$ telle que (la voiture est bleue) est vraie sur un intervalle si et seulement si elle est vraie sur tous ses sous-intervalles. Par contre, un événement tel que (la balle va de X_1 à X_2) ne peut être vrai sur deux intervalles dont l'un contient l'autre, le fait qu'un événement e se produit à un intervalle t est noté $OCCUR(e,t)$, dans ce cas l'événement e couvre l'intervalle t entièrement et ne peut survenir sur aucun de ses sous-intervalles. L'intervalle t est donc considéré comme étant le plus petit des intervalles dans lequel l'événement e se produit. Un processus se produisant sur un intervalle, noté $OCCURRRING(p,t)$ comme (je mange) peut être vrai sur certains de ses sous intervalles. Contrairement aux événements, les processus peuvent se dérouler durant certains sous intervalles de leur intervalle, nous pouvons ainsi compter le nombre d'occurrences d'un événement mais pas celui d'un processus. En utilisant ses trois entités fondamentales, Allen introduit aussi les notions de causalité et d'action. La causalité stipule qu'un événement peut être la cause d'un autre événement à condition que ce dernier ne le précède pas. D'un autre coté, une action est définie comme un événement ou un processus causé par un agent. À partir de la notion d'action, Allen introduit la notion de planification définie comme une séquence d'actions.

Allen définit enfin un calcul sur les intervalles, communément appelée l'algèbre d'intervalles, elle est basé sur treize primitives (sont illustrées dans le Tableau 3.3 [6]) qui sont des relations temporelles binaires qui permettent d'exprimer la position relative d'un intervalle par rapport à un autre, elles sont mutuellement exclusives, i.e. une seule relation est possible entre deux intervalles dans le cas où les extrémités des deux intervalles sont complètement définis, sinon, la relation peut être exprimée comme une disjonction de l'ensemble de relations de base. Il a introduit aussi les opérations inverse ($.^{-1}$), intersection ($\cap$), et composition ($\circ$) entre les relations, qui sont définies comme suit :

— $\forall\ X,Y : (X\ Rel\ Y) = (Y\ Rel^{-1}\ X)$

— $\forall\ X,Y : (X\ Rel_1 \cap Rel_2\ Y) = (X\ Rel_1\ Y)\ \&(X\ Rel_2\ Y)$

— $\forall\ X,Y : (X\ Rel_1 \circ Rel_2\ Y) = \exists\ \mathsf{Z} : (X\ Rel_1\ Z\ \&\ Z\ Rel_2\ Y)$

$\circ$	$\equiv$	p	p^{-1}	m	m^{-1}	o	o^{-1}	d	d^{-1}	s	s^{-1}	f	f^{-1}
$\equiv$	$\equiv$	p	p^{-1}	m	m^{-1}	o	o^{-1}	d	d^{-1}	s	s^{-1}	f	f^{-1}
p	p	p	$\top$	p	ρ	p	ρ	ρ	p	p	p	ρ	p
p^{-1}	p^{-1}	$\top$	p^{-1}	λ^{-1}	p^{-1}	λ^{-1}	p^{-1}	λ^{-1}	p^{-1}	λ^{-1}	p^{-1}	p^{-1}	p^{-1}
m	m	p	ρ^{-1}	p	θ	p	β	β	p	m	m	β	p
m^{-1}	m^{-1}	λ	p^{-1}	σ	p^{-1}	γ^{-1}	p^{-1}	γ^{-1}	p^{-1}	γ^{-1}	p^{-1}	m^{-1}	m^{-1}
o	o	p	ρ^{-1}	p	β^{-1}	α	ν	β	λ	o	γ	β	α
o^{-1}	o^{-1}	λ	p^{-1}	γ	p^{-1}	ν	α^{-1}	γ^{-1}	ρ^{-1}	γ^{-1}	α^{-1}	o^{-1}	β^{-1}
d	d	p	p^{-1}	p	p^{-1}	ρ	λ^{-1}	d	$\top$	d	λ^{-1}	d	ρ
d^{-1}	d^{-1}	λ	ρ^{-1}	γ	β^{-1}	γ	β^{-1}	ν	d^{-1}	γ	d^{-1}	β^{-1}	d^{-1}
s	s	p	p^{-1}	p	m^{-1}	α	γ^{-1}	d	λ	s	σ	d	α
s^{-1}	s^{-1}	λ	p^{-1}	γ	m^{-1}	γ	o^{-1}	γ^{-1}	d^{-1}	σ	s^{-1}	o^{-1}	d^{-1}
f	f	p	p^{-1}	m	p^{-1}	β	α^{-1}	d	ρ^{-1}	d	α^{-1}	f	θ
f^{-1}	f^{-1}	p	ρ^{-1}	m	β^{-1}	o	β^{-1}	β	d^{-1}	o	d^{-1}	θ	f^{-1}

$\alpha = (\text{pmo})\ \beta = (\text{ods})\ \gamma = (\text{od}^{-1}\text{f}^{-1})\ \sigma = (\equiv \text{ss}^{-1})\ \theta = (\equiv \text{ff}^{-1})\ \rho = (\text{pmods})$
$\lambda = (\text{pmod}^{-1}\text{f}^{-1})\ \nu = (\equiv \text{oo}^{-1}\text{dd}^{-1}\text{ss}^{-1}\text{ff}^{-1})\ \top = (\equiv \text{pp}^{-1}\text{mm}^{-1}\text{oo}^{-1}\text{dd}^{-1}\text{ss}^{-1}\text{ff}^{-1}).$

Figure 3.3 – Table de composition pour les relations temporelles de base

L'opération de composition est calculé en utilisant la table de composition, des fois appelé table de transitivité (est présenté dans la Figure 3.3[84]), afin de déduire la position relative de deux intervalles en fonction des relations les liant à un troisième intervalle. Exemple : Soit I_1, I_2, I_3 trois intervalles, Si I_1 avant (before) I_2 et I_2 rencontre (overlaps) I_3 Alors I_1 avant (before) I_3.

Allen définit alors un réseau de contraintes dans lequel les nœuds représentent les intervalles et les arcs sont étiquetés par une disjonction de relations de base

d'Allen pour représenter la relation temporelle qualitative entre les nœuds qui les concernent. La vérification de la cohérence du réseau est assurée par l'algorithme de fermeture transitive [6], qui est un algorithme de propagation de contraintes qui utilise la table de transitivité pour déduire toutes les relations possibles entre les intervalles et éliminer les relations qui rendraient le réseau incohérent. Lorsque les contraintes sont choisies à partir de l'algèbre complète d'Allen, la calcul de la fermeture transitive dans ce cas sera un problème NP-complet[85], il est donc peu probable qu'il existe des algorithmes efficaces pour raisonner dans l'algèbre complète. Plusieurs travaux se sont penchées sur la restriction de l'algèbre d'intervalles et ils s'intéressent à déterminer les sous classes de l'algèbre d' Allen dans lesquelles l'algorithme est polynômial [86]. Dans [84], les auteurs ont montré que cette algèbre contient exactement dix-huit sous-algèbres maximales traitables (sont présentées dans la Figure 3.4 [84]), et le raisonnement dans tout fragment non entièrement contenu dans l'une de ces sous-algèbres est NP-complet.

$$\mathcal{S}_{\mathsf{p}} = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{p})^{\pm 1} \subseteq r\}$$
$$\mathcal{S}_{\mathsf{d}} = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{d}^{-1})^{\pm 1} \subseteq r\}$$
$$\mathcal{S}_{\mathsf{o}} = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{o})^{\pm 1} \subseteq r\}$$
$$\mathcal{A}_1 = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{s}^{-1})^{\pm 1} \subseteq r\}$$
$$\mathcal{A}_2 = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{s})^{\pm 1} \subseteq r\}$$
$$\mathcal{A}_3 = \{r \mid r \cap (\mathsf{pmodf})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{s})^{\pm 1} \subseteq r\}$$
$$\mathcal{A}_4 = \{r \mid r \cap (\mathsf{pmodf}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{s})^{\pm 1} \subseteq r\}$$
$$\mathcal{E}_{\mathsf{p}} = \{r \mid r \cap (\mathsf{pmods})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{p})^{\pm 1} \subseteq r\}$$
$$\mathcal{E}_{\mathsf{d}} = \{r \mid r \cap (\mathsf{pmods})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{d})^{\pm 1} \subseteq r\}$$
$$\mathcal{E}_{\mathsf{o}} = \{r \mid r \cap (\mathsf{pmods})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{o})^{\pm 1} \subseteq r\}$$
$$\mathcal{B}_1 = \{r \mid r \cap (\mathsf{pmods})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{f}^{-1})^{\pm 1} \subseteq r\}$$
$$\mathcal{B}_2 = \{r \mid r \cap (\mathsf{pmods})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{f})^{\pm 1} \subseteq r\}$$
$$\mathcal{B}_3 = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{s}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{f}^{-1})^{\pm 1} \subseteq r\}$$
$$\mathcal{B}_4 = \{r \mid r \cap (\mathsf{pmod}^{-1}\mathsf{s})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{f}^{-1})^{\pm 1} \subseteq r\}$$
$$\mathcal{S}^* = \left\{ r \;\middle|\; \begin{array}{l} 1)\, r \cap (\mathsf{pmod}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{f}^{-1})^{\pm 1} \subseteq r, \text{ and} \\ 2)\, r \cap (\mathsf{ss}^{-1}) \neq \emptyset \Rightarrow (\equiv) \subseteq r \end{array} \right\}$$
$$\mathcal{E}^* = \left\{ r \;\middle|\; \begin{array}{l} 1)\, r \cap (\mathsf{pmods})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{s})^{\pm 1} \subseteq r, \text{ and} \\ 2)\, r \cap (\mathsf{ff}^{-1}) \neq \emptyset \Rightarrow (\equiv) \subseteq r \end{array} \right\}$$
$$\mathcal{H} = \left\{ r \;\middle|\; \begin{array}{l} 1)\, r \cap (\mathsf{os})^{\pm 1} \neq \emptyset \;\&\; r \cap (\mathsf{o}^{-1}\mathsf{f})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{d})^{\pm 1} \subseteq r, \text{ and} \\ 2)\, r \cap (\mathsf{ds})^{\pm 1} \neq \emptyset \;\&\; r \cap (\mathsf{d}^{-1}\mathsf{f}^{-1})^{\pm 1} \neq \emptyset \Rightarrow (\mathsf{o})^{\pm 1} \subseteq r, \text{ and} \\ 3)\, r \cap (\mathsf{pm})^{\pm 1} \neq \emptyset \;\&\; r \not\subseteq (\mathsf{pm})^{\pm 1} \Rightarrow (\mathsf{o})^{\pm 1} \subseteq r \end{array} \right\}$$
$$\mathcal{A}_{=} = \{r \mid r \neq \emptyset \Rightarrow (\equiv) \subseteq r\}$$

Figure 3.4 – Les 18 sous-algèbres maximales traitables de l'algèbre d'Allen

3.7 Les extensions temporelles de la logique de description

Plusieurs travaux ont été proposé pour étendre la logique de description avec une dimension temporelle afin de représenter et raisonner sur les connaissances temporelles [87, 88]. Ces extensions temporelles diffèrent les unes des autres de différentes manières. En effet, il existe deux façons d'ajouter la dimension temporelle, soit implicitement (la notion du temps est implicite dans le langage) ou explicitement (les opérateurs temporels sont ajoutés au langage), aussi sur la façon d'envisager la notion de temps, comme un ensemble d'instances ou d'intervalle. Ces choix ont un impact considérable sur l'expressivité et la décidabilité de la logique qui en résulte. Dans le cadre de notre étude, on s'intéresse aux extensions temporelles de logique de description basé sur l'intervalle afin de représenter et raisonner sur les actions et les plans. Schmiedel [89] fut le premier à proposer une extension des logiques de description avec la logique temporelle basée sur l'intervalle. Une représentation uniforme des états, des actions et des plans est fournie et les nouveaux opérateurs temporels introduits dans la logiques sont le qualificateur (@ at) et les quantificateurs temporels existentiel ($\diamond$ sometime) et universel ($\Box$ alltime). Ces quantificateurs temporels introduisent des variables temporelles avec un ensemble de contraintes. Il existe trois types de contraintes temporelles : les relations qualitatives entre deux intervalles en utilisant l'algèbre Allen, les contraintes métriques sur un seul intervalle afin de traiter les durées et les contraintes de granularité où un intervalle est requis pour prendre des valeurs multiples d'une unité de temps. Néanmoins, Schmiedel n'a proposé aucun algorithme pour la subsomption, il ne donne que quelques indications préliminaires mais plusieurs travaux ont montré que la logique de Schmiedel est indécidable car elle contient la logique temporelle à base d'intervalles indécidable de Halpern et Shoham [83]. Weida et Litman [90] ont proposé $T-Rex$, une intégration hybride entre la logique de description et les réseaux de contraintes. $T-Rex$ utilise deux langages différents pour représenter les actions et les plans, une logique de description non temporelle ($K-Rep$) pour décrire des actions et un autre langage pour composer des plans en ajoutant des informations temporelles. Les plans sont donc définis comme des collections d'étapes associées à des contraintes temporelles entre leurs durées. Chaque étape est associée à un type d'action, représenté par un concept générique dans $K-Rep$. Un algorithme de subsomption structurel de plan est défini, caractérisé en termes de correspondance de graphes, et basé sur la subsomption terminologique pure entre les types d'actions étiquetant les nœuds, et la subsomption

temporelle pure entre les relations d'intervalle étiquetant les arcs. Inspiré par les travaux précédant, Artale et all [5]. ont proposé $\mathcal{TL}-\mathcal{ALCF}$ une logique de description temporelle décidable avec une syntaxe formelle et une sémantique pour représenter et raisonner sur les actions et les plans. Cette logique est une extension de la logique de description non-temporelle $\mathcal{ALCF}$ par la logique temporelle TL qui permet d'introduire des variables temporelle et d'exprimer des contraintes temporelles entre eux en utilisant l'algèbre d'Allen. Le plan des actions ou des événements peut être représenté en $\mathcal{TL}-\mathcal{ALCF}$ comme un concept complexe qui décrits comme un modèle général dans lequel un ensemble de concepts ou d'individus sont disposés dans un certain ordre au moyen de réseau de contraintes temporelles qualitatives. Bettaz et all. [91] ont proposé récemment une extension temporelle pour la logique de description non monotone $JClassic_{\delta\epsilon}$ [4] dans le cadre des modèles de contrôle d'accès mais elles n'ont pas traité les contraintes temporelles dans la définition des concepts temporels, ce qui est important pour une représentation expressive des politiques d'autorisation contextuelles.

3.8 Conclusion

Nous avons abordé dans ce chapitre la logique de description, il existe toute une famille de logique descriptive allant des logiques les moins expressives au plus expressives, nous avons constaté que l'efficacité de raisonnement dépend de l'utilisation ou non de certain constructeurs spécifiques. Nous avons étudié les différents formalismes mis au point pour représenter et raisonner sur les connaissances temporelles, les logique de description temporelle sont inclue. Les extensions temporelles existantes de la logique de description basée sur l'intervalle offrent une richesse d'expression intéressante pour représenter formellement des politiques d'autorisation temporelles. Néanmoins, elles ne sont pas suffisamment expressives pour formuler des structures hiérarchiques d'actions permettant le raisonnement temporel et l'héritage défavorable, ce qui est nécessaire pour spécifier un modèle de contrôle d'accès flexible pour les données sensibles dans les services cloud. Une approche évidente est d'étendre la logique de description non monotone par la logique temporelle TL.

CHAPITRE 4
LA LOGIQUE DE DESCRIPTION TEMPORELLE NON MONOTONE
$TL-JCLASSIC_{\delta\epsilon}$

4.1 Introduction

Le développement d'un nouveau formalisme logique commence souvent par analyser ce qui doit pouvoir être exprimé. Il convient ensuite de définir la syntaxe, la sémantique et les algorithmes d'inférence appropriés. L'étude théorique de la complexité des algorithmes d'inférence est une étape requise avant d'envisager l'implémentation de ces algorithmes et de mettre en œuvre l'outil de raisonnement. Dans le cadre de notre travail, il est nécessaire d'avoir un nouveau formalisme non monotone qui permet d'une part de représenter de manière expressive les connaissances temporelle surtout les plans d'action qui sont composés d'actions ordonnées strictes ou génériques sujettes à exception, et d'autre part de fournir des services de raisonnement capable de manipuler ces connaissances temporelles et d'inférer de nouvelles connaissances dans des délais de temps acceptables. Dans ce chapitre, nous proposons d'étendre la logique de descriptions non monotone $JClassic_{\delta\epsilon}$ en ajoutant une dimension temporelle, la logique temporelle $\mathcal{TL}$, tout en maintenant une complexité polynomiale de la subsumption. Le nouveau formalisme obtenu sera renommé $TL-JClassic_{\delta\epsilon}$. Nous allons présenter au cours de ce chapitre la syntaxe de $TL-JClassic_{\delta\epsilon}$, sa sémantique et décrire en détail les algorithmes d'inférences qui permettent de calculer la subsomption et l'héritage des connaissances temporelles dans $TL-JClassic_{\delta\epsilon}$, et étudier aussi la complexité de leurs algorithmes d'inférences.

4.2 Syntaxe

La logique $TL-JClassic_{\delta\epsilon}$ comprend l'ensemble des connecteurs de $JClassic_{\delta\epsilon}$ [69] et de la logique temporelle $\mathcal{TL}$, il est défini donc à l'aide d'un ensemble de concepts primitifs P, d'un ensemble de rôles atomiques R, un ensemble d'individus Ind, et un ensemble de variables temporelles $(\overline{X})$. Les nouveaux connecteurs temporels qui ont été ajoutés sont présentés comme suit :

— Le qualificatif temporel @ spécifie qu'un concept est vrai durant une certaine variable temporelle (X) ou à un intervalle temporal.

— Le qualificatif temporel de substitution $(C[Y]@X)$ permet de renommer dans le concept temporel C la variable temporelle Y à X.

— Le quantificateur existentiel temporel $\diamond$ introduit un ensemble de variables temporelles $(\overline{X})$ dans la description d'un concept temporel et les relie par un ensemble de contraintes temporelles $(\overline{Tc})$, les variables temporelles utilisées dans les contraintes temporelles doivent être déjà déclarées dans le même quantificateur temporel, à l'exception de la variable temporelle spéciale ($\sharp$) qui est considéré comme un intervalle de référence (le temps actuel)

— Les variables temporelles indiquent les intervalles liés par les contraintes temporelles où chaque contrainte temporelle est une disjonction arbitraire des relations d'Allen (Rel).

La syntaxe de $TL-JClassic_{\delta\epsilon}$ est présenté dans le Tableau 4.1.

C,D→	$\top$	Le concept le plus général
	$\mid \bot$	Le concept le plus spécifique
	$\mid P$	Concept primitif
	$\mid C \sqcap D$	Conjonction de concepts
	$\mid \neg P$	Négation de concept primitif
	$\mid \forall r$: C	Restriction sur tous les rôles r
	$\mid Min_u$	ensemble des réels $> u$ (u est un nombre réel)
	$\mid Max_u$	ensemble des réels $\leq u$
	| ONE-OF$\{I_1,..,I_n\}$	Concept en extension ($\{I_1,..,I_n\}$ sont des individus)
	| r FILLS $\{I_1,..,I_n\}$	Sous-ensemble de valeur pour r
	| r AT-LEAST n	Restriction minimum de nombre pour r (n est un entier)
	| r AT-MOST n	Restriction maximum de nombre pour r
	$\mid \delta$ C	Concept par défaut
	$\mid C^{\epsilon}$	Exception sur le concept
	| C @ X	Qualificatif temporel
	| C [Y]@ X	Qualificatif temporel de substitution
	$\mid \diamond(\overline{X})\overline{Tc}.C$	Quantificateur existentiel temporel
Tc →	(X(Rel)Y)	Contrainte temporelle
	| (X(Rel)$\sharp$)	
	|($\sharp$(Rel)Y)	
$\overline{Tc}$→	Tc | Tc $\overline{Tc}$	
Rel →	Rel,S	Disjonction
	|s|m|f|o...	Relations d'Allen
$\overline{X}$ →	X|X $\overline{X}$...	Variables temporelles

Tableau 4.1 – La synatxe de $TL-JClassic_{\delta\epsilon}$

Un plan d'action est défini dans $TL-JClassic_{\delta\epsilon}$ comme un concept temporel qui peut être représenté sous la forme d'un graphe orienté où les arcs sont des contraintes temporelles qualitatives entre les variables temporelles qui sont associés à des nœuds. Les nœuds sont des concepts de $TL-JClassic_{\delta\epsilon}$, ils représentent les actions ou tâches d'un plan, de sorte que chaque action est décrite comme un concept temporel qualitatif. La description d'un plan d'action est satisfaite si leurs contraintes temporelles sont cohérentes. Il est à noter que les plans d'action peuvent être incorporés dans d'autres plans mais pas de manière récursive dans eux-mêmes. Cependant, $TL-JClassic_{\delta\epsilon}$ a la particularité de permettre de spécifier explicitement les actions strictes, facultatives ou ignorées dans la description d'un plan. Nous précédons le concept qualificatif temporel par (δ) pour représenter une action facultative $(\delta ACTION@X)$, par (ϵ) pour une action ignorée $(ACTION^{\epsilon}@X)$ sinon le concept est considéré comme une action stricte. Notons que, nous pouvons ignorer seulement les actions facultatives.

Exemple 3. *En général, le processus de soins à l'hôpital est une séquence des actions suivantes : inscrire un patient* $(Registration)$*, consulter leurs données médicales* $(ConsultPatientMedicalData)$*, effectuer un examen physique* $(PhysicalExamination)$ *pour pouvoir bien identifier la nature de la douleur, effectuer un test d'imagerie qui est souvent une action facultative* $(\delta PerformImagingTest)$*, déterminer le diagnostic* $(DetermineDiagnosis)$*, appliquer le traitement* $(ApplyTreatment)$*, et enfin signer une décharge pour sortir de l'hôpital* $(Discharge)$*. Pour définir ce plan dans* $TL-JClassic_{\delta\epsilon}$*, on doit d'abord introduire des variables temporelles* $(\overline{X})$ *, spécifier des relation temporelles qualitatives entre les variables temporelles* $(\overline{Tc})$ *et enfin associer chaque action du processus de soin à une variable temporal comme suit :*

$GeneralCareProcess \sqsubseteq \Diamond\ (X_1, X_2, X_3, X_4, X_5, X_6)$

$(X_1\{p\}X_2)$, $(X_2\{p \mid d\}X_3)$,$(X_2\{p \mid d\}X_4)$, $(X_4\{p\}X_5)$, $(X_5\{p\}X_6)$

$Registration@X_1 \sqcap ConsultPatientMedicalData@X_2 \sqcap PhysicalExamination@X_3 \sqcap$

$\delta PerformImagingTest\ @X_4 \sqcap (DetermineDiagnosis \sqcap ApplyTreatment)\ @X_5$

$\sqcap\ Discharge@X_6$.

Lorsqu'un patient est en situation critique, nous devons passer ou ignorer l'action facultative effectuer un test d'imagerie $(\delta PerformImagingTest)$*, alors, dans ce cas, le processus de soins d'urgence* $(EmergencyCarePprocess)$ *sera défini comme suit :*

$EmergencyCarePrococess \sqsubseteq \Diamond\ Y\ (Y\{d\}\sharp)$

$(GeneralCarePrococess[X_4]@Y)\ @\sharp \sqcap (PerformImagingTest^{\epsilon}@Y)$.

La présence de l'action ($PerformImaging - Test^{\epsilon}$ @Y) dans la définition de $EmergencyCareProcess$ permet d'exprimer le fait que l'action ($performimagingtest$) à la période Y est annulée mais elle devrait être dans la définition de $EmergencyCareProcess$ puisqu'il s'agit d'un $GeneralCareProcess$. Cette représentation donc permet de classer $EmergencyCareProcess$ sous le concept $GeneralCareProcess$ mais elle ne permet pas au concept $EmergencyCareProcess$ d'hériter du concept $GeneralCareProcess$ l'action ($PerformImagingTest$) à l'intervalle Y car elle devrait être ignorée.

4.3 Sémantique intentionnelle

Dans cette section, nous définissons $TL - CL_{\delta\epsilon}$ un cadre algébrique qui est utilisé pour donner une sémantique intentionnelle aux concepts de $TL - JClassic_{\delta\epsilon}$. $TL - CL_{\delta\epsilon}$ recouvre les caractéristiques formelle de la définition des concepts de $TL - JClassic_{\delta\epsilon}$ pour nous permettre de calculer la subsomption entre les concepts de $TL - JClassic_{\delta\epsilon}$ et de déduire leurs propriétés temporelle héritées et cela par l'intermédiaire de l'algorithme de subsumption ($TL - Sub_{\delta\epsilon}$) et l'algorithme d'héritage qui seront utilisés ensuite dans le développement d'un outil de raisonnement pour $TL - JClassic_{\delta\epsilon}$.

La subsomption entre deux concepts temporels C et D [5, 90] est généralement réduite à la comparaison de leurs réseaux de contraintes temporelles et à la comparaison de leurs concepts non temporels (un concept non temporel est le concept C dans le concept temporel qualitatif ($C@X$) qui ne contient pas des quantificateurs temporels existentiels), mais comme les concepts temporels sont composés principalement des variables temporels, la comparaison se fait sous un mappage de variables temporelles de D (le subsumer) vers C (le subsumé).

Le mappage de variables $\mathcal{M}$ est une fonction totale $\mathcal{M} : \overline{Y} \mapsto \overline{X}$ qui mappe chaque variable temporelle dans l'ensemble des variables temporelles de concept D ($\overline{Y}$) à une variable temporelle distincte dans le concept C ($\overline{X}$) [5]. Il est à noter qu'il est important de calculer la fermeture transitive des contraintes temporelles de deux concepts car la comparaison de leurs réseaux de contraintes temporelles est liée à la mesure dans laquelle ses contraintes sont explicitées dans leurs descriptions. La relation de subsomption dans $TL - JClassic_{\delta\epsilon}$ est considérée comme dans $JClassic_{\delta\epsilon}$ [4], d'un point de vue descriptif et structurel. Dans les sections suivantes nous détaillons chaque point de vue.

4.3.1 Point de vue descriptif du cadre algébrique

Nous construisons un système équationnel (EQ) qui présente les principales caractéristiques et propriétés des connecteurs de $TL-JClassic_{\delta\epsilon}$, à partir duquel nous définissons des relations d'équivalence entre les contraintes temporelles et entre les concepts non temporels, et nous formalisons ainsi la relation de subsomption entre les concepts temporels de $TL-JClassic_{\delta\epsilon}$.

Un système équationnel pour la logique $TL-JClassic_{\delta\epsilon}$:

$\forall\ C \in TL-JClassic_{\delta\epsilon}$, s'écrit sous la forme $\Diamond\ (X_0,..,X_n)_{n\epsilon N}\ (X_0\ Rel_{(0,0)}\ X_0)$,..,$(X_i\ Rel_{(i,j)}\ X_j)_{(i,j\leq n)}\ (C_0@X_0) \sqcap ... \sqcap (C_n@X_n)$ où $C_0...C_n$ sont des concepts non temporals, $\forall\ r \in$ **R**, et $\forall\ I_{z(z\epsilon N)} \in$ **Ind** :

01 : $(X_i\ Rel\ X_j) \equiv (X_j\ Rel^{-1}\ X_i)$

02 : $(X_i\ Rel_1 \cap Rel_2\ X_j) \equiv (X_i\ Rel_1\ X_j)(X_i\ Rel_2\ X_j)$

03 : $(X_i\ Rel_1 \circ Rel_2\ X_j) \equiv (X_i\ Rel_1\ X_k)(X_k\ Rel_2\ X_j)_{(k\leq n)}$

04 : $C_i@X_1 \sqcap C_j@X_1 \equiv (C_i \sqcap C_j)@X_1$

05 : $(C_i@X_1 \sqcap C_j@\sharp)@X_2 \equiv C_i@X_1 \sqcap C_j@X_2$

06 : $\Diamond(X_0,..,X_n)_{n\epsilon N}\overline{Tc_1}(C_1 \sqcap (\Diamond(X'_0,..,X'_m)_{(m\epsilon N)}\overline{Tc_2}\ (C'_0@X'_0) \sqcap ... \sqcap (C'_m@X'_m)$ $[X'_0]@X_0...[X'_i]@X_{j(i\leq m,j\leq n)}))\ @X_1 \equiv \Diamond((X_0,..,X_n) \cup (X'_0,..,X'_m))_{([X'_0/X_0]..[X'_i/X_j])}$ $\overline{Tc_1 \cup Tc'_2}(C_1 \sqcap (C'_0@X_0) \sqcap ... \sqcap (C'_m@X_j))@X_1$

07 : $C \sqcap \Diamond(X_0,..,X_n)_{n\epsilon N}\overline{Tc}(C_0@X_0) \sqcap .. \sqcap (C_n@X_n) \equiv \Diamond(X_0,..,X_n)\overline{Tc}(C \sqcap (C_0@X_0) \sqcap .. \sqcap (C_n@X_n))@\sharp$

08 : $(C_i \sqcap C_j) \sqcap C_k \equiv C_i \sqcap (C_j \sqcap C_k)$

09 : $C_i \sqcap C_j \equiv C_j \sqcap C_i$

10 : $C_i \sqcap C_i \equiv C_i$

11 : $\top \sqcap C_i \equiv C_i$

12 : $\bot \sqcap C_i \equiv \bot$

13 : $Min_{u_1} \sqcap Min_{u_2} \equiv Min_{max(u_1,u_2)}$

14 : $Max_{u_1} \sqcap Max_{u_1} \equiv Max_{min(u_1,u_2)}$

15 : $ONE-OF\ \{I_1,..,I_z\} \sqcap ONE-OF\ \{I_1,..,I_{z'}\} \equiv ONE-OF\ \{I_1,..,I_z\} \cap \{I_1,..,I_{z'}\}$

16 : $r\ FILLS\ \{I_1,..,I_z\} \sqcap r\ FILLS\ \{I_1,..,I_{z'}\} \equiv r\ FILLS\ \{I_1,..,I_z\} \cup \{I_1,..,I_{z'}\}$

17 : $r\ FILLS\ \varnothing \sqcap \top \equiv \top$

18 : $(\forall r : C_i) \sqcap (\forall r : C_j) \equiv \forall r : (C_i \sqcap C_j)$

19 : $\forall r : \top \equiv \top$

20 : $r\ AT-LEAST\ m \sqcap r\ AT-LEAST\ n \equiv r\ AT-LEAST\ max(m,n)$

21 : $r\ AT-LEAST\ 0 \equiv \top$

22 : $r\ AT-MOST\ m \sqcap r\ AT-MOST\ n \equiv r\ AT-MOST\ min(m,n)$

23 : $r\ AT-MOST\ 0 \equiv \forall\ r : \bot$

24 : $(\delta C_i)^{\epsilon} \equiv {C_i}^{\epsilon}$

25 : $\delta(C_i \sqcap C_j) \equiv \delta C_i \sqcap \delta C_j$

26 : $C_i \sqcap \delta C_i \equiv C_i$

27 : ${C_i}^{\epsilon} \sqcap \delta C_i \equiv {C_i}^{\epsilon}$

28 : $\delta\delta C_i \equiv \delta C_i$

Les sept premiers axiomes correspondent aux propriétés de nouveaux connecteurs temporels qui ont été ajoutés à notre logique [5].

Les axiomes de 01 à 03 correspondent respectivement à l'inverse ($.^{-1}$), à l'intersection binaire ($\cap$) et à la composition binaire ($\circ$). Ces opérations sont définies sur les relations entre les intervalles dans l'algèbre d'Allen.

L'axiome 04 indique que l'opérateur $@$ est distributive sur la conjonction.

L'axiome 05 exprime que le concept temporel qualifié est valide au premier intervalle lié à celui-ci.

L'axiome 06 simplifie l'expression temporelle en développant leurs concepts temporels qualifiés qui incluent les quantificateurs temporels existentiels. L'expression $((X_0,..,X_n) \cup (X'_0,..,X'_m))$ est l'union des ensembles de variables temporel de deux concepts où chaque occurrence de X'_0,..,X'_i est substitué par $X_0,...,X_j$ respectivement, alors que pour les autres éléments de $\overline{X'}$, si leurs noms existent déjà dans $\overline{X}$, alors ils seront renommées avec de nouveaux identificateurs, sinon ils seront rajoutés directement à l'ensemble. La même substitution des variables ou le renommée des noms aura lieu dans les contraintes temporelles $\overline{Tc2'}$ et dans les concepts temporels qualifiés $(C_i@X_j)$.

L'axiome 07 correspond à la conjonction d'un concept non temporel et d'un concept temporel.

Les axiomes de 08 à 28 représentent les propriétés des connecteurs de $JClassic_{\delta\epsilon}$, les axiomes de 08 à 23 sont classiques, elles concernent les propriétés de connecteurs de la logique de description, les axiomes 24 à 28 correspondent aux propriétés des connecteurs δ et ϵ [69].

La subsumption descriptive :

Dans la logique $TL - JClassic_{\delta\epsilon}$, la subsomption descriptive entre deux concepts temporels C et D écrits sous la forme simplifié (signifie qu'il existe une contrainte temporelle entre chaque paire de leurs variables temporelles et tous leurs concepts temporels qualifiés sont des concepts non temporels) s'est basée sur la comparaison entre leurs contraintes temporelles et sur la comparaison entre leurs concepts non temporels sous un mappage de variables temporelles de D vers C. On note $\sqsubseteq_d$ pour une subsomption descriptive. $\sqsubseteq_d$ est une relation d'ordre partiel permettant d'ordonner les termes de $TL - JClassic_{\delta\epsilon}$. L'égalité sous le mappage de variables $\mathcal{M}$ entre deux termes de $TL - JClassic_{\delta\epsilon}$ est notée $=_{EQ_{\mathcal{M}}}$. $=_{EQ_{\mathcal{M}}}$ est une relation de congruence qui partitionne l'ensemble des termes de $TL - JClassic_{\delta\epsilon}$, i.e., $=_{EQ_{\mathcal{M}}}$ permet de former des classes d'équivalence entre les termes sous $\mathcal{M}$. Nous définissons la relation de congruence sous le mappage de variables comme suit :

Définition 1 (Équivalence des contraintes temporelles)**.** *Soient* $Tc_1 = (X\ Rel_{(x,x')}\ X')$ *et* $Tc_2 = (Y\ Rel_{(y,y')}\ Y')$*, deux contraintes temporelles où* $\mathcal{M}(Y) = X$ *et* $\mathcal{M}(Y') = X'$*.* $Tc_1 =_{EQ_{\mathcal{M}}} Tc_2$*, i.e.,* Tc_1 *est équivalente à* Tc_2 *sous le mappage de variables* $\mathcal{M}$*, ssi* $Rel_{(x,x')} \sqcap Rel_{(y,y')} =_{EQ} Rel_{(x,x')}$

Définition 2 (Équivalence des concepts non temporels)**.** *Soient* $(C_1@X_1)$ *et* $(D_1@Y_1)$*, deux concepts temporels qualifiés, où* C_1 *et* D_1 *sont des concepts non temporal et* $\mathcal{M}(Y_1) = X_1$*.* $(C_1@X_1) =_{EQ_{\mathcal{M}}} (D_1@Y_1)$*, i.e.,* $(C_1@X_1)$ *est équivalente à* $(D_1@Y_1)$ *sous le mappage de variables* $\mathcal{M}$*, ssi* $C_1 \sqcap D_1 =_{EQ} C_1$*.*

Maintenant, nous pouvons définir la subsomption descriptive entre deux concepts temporelle simplifiés de $TL - JClassic_{\delta\epsilon}$ sous le mappage de variables $\mathcal{M}$ et la conjonction des concepts comme suit :

Définition 3 (A descriptive subsumption)**.** *Soient C=*$\diamond\ (X_0,..,X_n)_{(n\in\mathbb{N})}\ (X_0\ rel_{(0,0)}\ X_0),..,(X_n\ rel_{(n,n)}\ X_n)(C_0@X_0) \sqcap..\sqcap(C_n@X_n)$ *et D=*$\diamond\ (Y_0,..,Y_m)_{(m\in\mathbb{N},m\leq n)}\ (Y_0\ rel_{(0,0)}\ Y_0),..,(Y_m\ rel_{(m,m)}\ Y_m)\ (D_0@Y_0) \sqcap..\sqcap(D_m@Y_m)$*, deux concepts simplifies de* $TL - JClassic_{\delta\epsilon}$*.*

C $\sqsubseteq_d$ *D, i.e., D subsume descriptivement C, s'il existe un mappage de variables* $\mathcal{M}$ *:*$(\overline{Y_0,..,Y_m}) \mapsto (\overline{X_0,..,X_n})$ *où* $\forall i, i' \leq n, \exists j, j' \leq m$:

- $(X_i\ R_{(i,i')}\ X_{i'}) =_{EQ_{\mathcal{M}}} (Y_j\ R_{(j,j')}\ Y_{j'})$.
- $(C_i@X_i) =_{EQ_{\mathcal{M}}} (D_j@Y_j)$.

Du point de vue algorithmique, les termes ne sont pas facilement manipulables dans le calcul de la subsomption. Nous adoptons une subsomption structurelle qui consiste à comparer deux concepts temporels en fonction de leur sémantique intentionnelle.

4.3.2 Point de vue structurel du cadre algébrique

Pour définir le point de vue structurel de la subsomption dans $TL-JClassic_{\delta\epsilon}$, nous définissons dans cette section une structure algébrique de concepts temporels $TL-CL_{\delta\epsilon}$ qui permet d'associer les concepts temporels par leurs formes normales fondées sur l'ensemble de leurs propriétés en appliquant un homomorphisme de l'ensemble des termes de $TL-JClassic_{\delta\epsilon}$ vers les éléments de $TL-CL_{\delta\epsilon}$. Le point de vue structurel pour la subsomption entre deux concepts temporels consiste alors à comparer leurs formes normales.

Cadre algébrique $TL-CL_{\delta\epsilon}$

En utilisant le système EQ présenté dans la section 4.3.1, nous calculons pour chaque concept une dénotation structurelle, à savoir une forme normale pour représenter ce concept. Un élément de $TL-CL_{\delta\epsilon}$ correspond à un terme T de $TL-JClassic_{\delta\epsilon}$ est une paire $\langle t_\theta, t_\delta\rangle$ où t_θ est la partie stricte de T et t_δ est la partie par défaut. Les deux ayant la même structure de 7-uplets $(t, dom, min, max, \pi, r, \epsilon)$, les six derniers champs de la 7- uplets sont les mêmes que le 6- uplets défini pour $JClassic_{\delta\epsilon}$, nous avons introduit le premier champ pour représenter les propriétés temporelles d'un terme T de $TL-JClassic_{\delta\epsilon}$. Les éléments de chaque 7-uples sont donc définis comme suit :

- **t** : est représenté sous forme d'un 3-uplets ⟨ var, cst, cpt ⟩ où :
  - **var** : est l'ensemble de variables temporelles introduit par $\diamond$ dans la description de T.
  - **cst** : est l'ensemble de contraintes temporelles définis dans T, chaque contrainte temporelle est présenté sous forme d'un 3-uplets (x, rel, y) où x et y sont des variables temporelles et rel est l'ensemble d'une disjonction de relations d'Allen.
  - **cpt** : est l'ensemble de concepts temporels qualifiés définis dans T, chaque concept qualifié ($C@X$) est défini sous forme (c, x, ren) où c est la forme normale du concept qualifié C, x est la variable temporelle liée au C et ren est l'ensemble de substitution de variables si le concept C contient $[Y]@X$,

chaque substitution est représenté par le couple (y, x) pour indiquer que la variable temporelle y dans c est renommée par x.

- **dom** : est soit l'ensemble d'individus si la description de T contient une propriété ONE-OF, ou sinon le symbole spécial **UNIV**.
- **min**(resp. **max**) : est soit un réel si T contient une propriété MIN (resp.MAX), ou le symbole spécial **MIN-R** (resp. **MAX-R**) sinon.
- π : est l'ensemble des concepts primitifs appartenant à T.
- **r** : est l'ensemble des rôles défini dans T, où chaque rôle présenté sous forme d'un 5-uplets $\langle r, fillers, least, most, c\rangle$ où :
  - **r** est le nom de rôle,
  - **fillers** : est l'ensemble d'individus, si T contient une propriété $rFills$ sinon $\varnothing$
  - **least** (resp. **most**) : est un entier, si T contient une propriété $rATLEAST$ (resp. $rAT-MOST$), else 0 (resp. NOLIMIT).
  - **c** : est la forme normale de C, si T contient une propriété $\forall r : C$.
- ϵ : est l'ensemble de 7-uplets $(t, dom, min, max, \pi, r, \epsilon)$.

Notation : La structure complète de la forme normale est notée comme suit : ⟨ ($t_{\theta t}$, $t_{\theta dom}$, $t_{\theta min}$, $t_{\theta max}$,$t_{\theta \pi}$,$t_{\theta r}$, $t_{\theta \epsilon}$), ($t_{\delta t}$, $t_{\delta dom}$, $t_{\delta min}$,$t_{\delta max}$, $t_{\delta \pi}$,$t_{\delta r}$,$t_{\delta \epsilon}$)⟩

Dans la section précédente, nous avons présenté la structure des éléments de $TL{-}CL_{\delta\epsilon}$ mais on n'a pas montrer comment calculer la forme normale d'un concept temporel défini avec une description T. Pour cela, nous définissons un homomorphisme (h) de l'ensemble des termes de $TL-JClassic_{\delta\epsilon}$ à l'ensemble des éléments $TL-CL_{\delta\epsilon}$. L'application de cet homomorphisme sur un terme T permet de calculer la forme normale de ce concept.

Homomorphisme h

L'interprétation des connecteurs et des constantes de $TL-JClassic_{\delta\epsilon}$ est donnée en détail dans l'algorithme 1.

Pour compléter la définition de la forme normale d'un concept de $TL-JClassic_{\delta\epsilon}$, nous devons définir l'union de deux éléments de $TL-CL_{\delta\epsilon}$ sous un mappage de variables.

Définition 4 (Union de deux formes normales ($\otimes_{\mathcal{M}}$))**.** *Soient C* $\langle c_\theta, c_\delta\rangle$ *et D* $\langle d_\theta, d_\delta\rangle$ *deux formes normales simplifiées de* $TL-CL_{\delta\epsilon}$ *différentes de* b_0[1]*.* $\mathcal{M}$ *est un ensemble de*

1. b_0 *est une constante utilisée comme dénotation de* $\perp$*.*

Algorithm 1 $SimpleTerme(C)$

Require: C is a simple concept description of $TL - JClassic_{\delta\epsilon}$
Ensure: NF_C : normal form of C in $TL - CL_{\delta\epsilon}$

if C is $\top$ **then**
  $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R, \varnothing,\varnothing,\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R, \varnothing,\varnothing,\varnothing)\rangle$
end if
if C is $\bot$ **then**
  $NF\_C \leftarrow b_0$
end if
if C is P **then**
  $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R, P,\varnothing,\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R, P,\varnothing,\varnothing)\rangle$
end if
if C is ONE-OF E **then**
  $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, E, Min - R, Max - R, \varnothing,\varnothing,\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, E, Min - R, Max - R, \varnothing,\varnothing,\varnothing)\rangle$.
end if
if C is Min u **then**
  $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, u, Max - R, \varnothing,\varnothing,\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, u, Max - R, \varnothing,\varnothing,\varnothing)\rangle$.
end if
if C is Max u **then**
  $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, u,\varnothing,\varnothing,\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, u,\varnothing,\varnothing,\varnothing)\rangle$.
end if
if C is r FILLS E (E $\not\equiv \varnothing$) **then**
  $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R,\varnothing,\{\langle r, E, |\ E\ |, NOLIMIT, \top\rangle\},\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R,\varnothing,\{\langle r, E, |\ E\ |, NOLIMIT, \top\rangle\},\varnothing)\rangle$
end if
if C is r FILLS $\varnothing$ **then**
  $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R, \varnothing,\varnothing,\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R, \varnothing,\varnothing,\varnothing)\rangle$.
end if
if C is r AT-LEAST n (n $\geq$ 1) **then**
  $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R,\varnothing,\{\langle r,\varnothing, n, NOLIMIT, \top\rangle\},\varnothing),$
$(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R,\varnothing,\{\langle r,\varnothing, n, NOLIMIT, \top\rangle\},\varnothing)\rangle$.
end if
if C is r AT-LEAST 0 **then**
  $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R, \varnothing,\varnothing,\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min - R, Max - R, \varnothing,\varnothing,\varnothing)\rangle$.
end if

32: **if** C is r AT-MOST n (n ≥ 1) **then**
33: $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing, \{\langle r,\varnothing,0,n,\top\rangle\},\varnothing),$
34: $(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing, \{\langle r,\varnothing,0,n,\top\rangle\},\varnothing)\rangle$.
35: **end if**
36: **if** C is r AT-MOST 0 **then**
37: $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing, \{\langle r,\varnothing,0,0,b_0\rangle\},\varnothing),$
38: $(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing, \{\langle r,\varnothing,0,0,b_0\rangle\},\varnothing)\rangle$.
39: **end if**
40: **if** C is $\forall r : \top$ **then**
41: $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing,\varnothing,\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing,\varnothing,\varnothing)\rangle$.
42: **end if**
43: **if** C is $\forall r : \bot$ **then**
44: $NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing, \{\langle r,\varnothing,0,0,b_0\rangle\},\varnothing),$
45: $(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing, \{\langle r,\varnothing,0,0,b_0\rangle\},\varnothing)\rangle$.
46: **end if**
47: **return** NF_C

paires de variables temporelles (x, y), *où* $x \in c_{\delta t_{var}}$ *et* $y \in d_{\delta t_{var}}$, *qui représentent un mappage de variables de D vers C.* $C \otimes_{\mathcal{M}} D = \langle(c_\theta \oplus_{\mathcal{M}} d_\theta), (c_\delta \oplus_{\mathcal{M}} d_\delta)\rangle$.

Cette définition fait appel à une opération d'union de deux 7-uple de $TL-CL_{\delta\epsilon}$ sous $\mathcal{M}$ qui sera défini comme suit :

Définition 5 (Union de deux 7-uples $\oplus_{\mathcal{M}}$)**.** *Soient c (*c_t, c_{dom}, c_{min}, c_{max}, c_π, c_r, c_ϵ*) et d (*d_t, d_{dom}, d_{min}, d_{max}, d_π, d_r, d_ϵ*) deux 7-uples.* $c \oplus_{\mathcal{M}} d$ *est l'union de chaque champ de* c *et* d *sous un mappage de variables* $\mathcal{M}$*. Ceci se fait à l'aide de la fonction* $union-uple(c, d, \mathcal{M})$*, qui sera détaillé dans l'algorithme 2.*

Subsomption structurelle

La subsomption structurelle entre deux concepts de $TL-JClassic_{\delta\epsilon}$ est réduite à la comparaison entre leurs formes normales. On note par $\sqsubseteq_s$ la subsomption structurelle. $\sqsubseteq_s$ est une relation d'ordre partiel. L'égalité structurelle de deux éléments de $TL-CL_{\delta\epsilon}$ est noté $=_{CL}$. $=_{CL}$ est une relation de congruence comme $=_{EQ}$ dans la subsomption descriptive. Nous définissons la subsomption structurelle en utilisant la relation de congruence et l'union des deux formes normales sous un mappage de variables comme suit :

Définition 6 (Subsomption structurelle)**.** *Soient C* $\langle c_\theta, c_\delta\rangle$ *et D* $\langle d_\theta, d_\delta\rangle$ *deux formes normales de concepts de* $TL-JClassic_{\delta\epsilon}$*. C* $\sqsubseteq_s$ *D, i.e., D subsume structurellement C, ssi il existe un mappage de variables* $\mathcal{M}$ *où :* $C \otimes_{\mathcal{M}} D =_{CL} C$

Algorithm 2 $union - uple(C, D, VarMap)$

Require:
C ($\langle c_{var}, c_{cst}, c_{cpt}\rangle_t, c_{dom}, c_{min}, c_{max}, c_\pi, c_r, c_\epsilon$)
D($\langle d_{var}, d_{cst}, d_{cpt}\rangle_t, d_{dom}, d_{min}, d_{max}, d_\pi, d_r, d_\epsilon$)
VarMap : a set of temporal variable pairs (X, Y) that represent variable mapping from D to C, where $X \in c_{var}$ and $Y \in d_{var}$
Ensure: U($\langle u_{var}, u_{cst}, u_{cpt}\rangle_t, u_{dom}, u_{min}, u_{max}, u_\pi, u_r, u_\epsilon$) : is the union of C and D under the variable mapping VarMap.
$\langle u_{var}, u_{cst}, u_{cpt}\rangle \leftarrow \langle\varnothing, \varnothing, \varnothing\rangle$
if $\langle c_{var}, c_{cst}, c_{cpt}\rangle = \langle\varnothing, \varnothing, \varnothing\rangle$ **then**
  if $\langle d_{var}, d_{cst}, d_{cpt}\rangle = \langle\varnothing, \varnothing, \varnothing\rangle$ **then**
    $u_{dom} \leftarrow c_{dom} \cup d_{dom}$
    $u_{min} \leftarrow max(c_{min} \cup d_{min})$
    $u_{max} \leftarrow min(c_{max} \cup d_{max})$
    $u_\pi \leftarrow c_\pi \cup d_\pi$
    $u_\epsilon \leftarrow c_\epsilon \cup d_\epsilon$
    $u_r \leftarrow \varnothing$
    for $\langle r, fillers1, least1, most1, c1\rangle \in c_r$ **do**
      if $\exists\langle r, fillers2, least2, most2, c2\rangle \in d_r$ **then**
        $u_r \leftarrow u_r \cup \langle r, fillers, least, most, cpt\rangle$ with :
        $fillers \leftarrow fillers1 \cup fillers2$
        $least \leftarrow max(least1, least2, \mid fillers \mid)$
        $most \leftarrow min(most1, most2, \mid cpt_{dom} \mid)$
        $cpt \leftarrow c1 \otimes c2$
      else
        $u_r \leftarrow u_r \cup \langle r, fillers1, least1, most1, c1\rangle$
      end if
    end for
    for all $(\langle r, fillers2, least2, most2, c2\rangle \in d_r)$ **do**
      if $\nexists\langle r, fillers1, least1, most1, c1\rangle \in c_r$ **then**
        $u_r \leftarrow u_r \cup \langle r, fillers2, least2, most2, c2\rangle$
      end if
    end for
  else
    $U \leftarrow D$
    if $\exists(A, \sharp, \varnothing) \in u_{cpt}$ **then**
      $A \leftarrow A \otimes C$
    else
      $u_{cpt} \leftarrow u_{cpt} \cup (C, \sharp, \varnothing)$
    end if
  end if
else
  if $\langle d_{var}, d_{cst}, d_{cpt}\rangle = \langle\varnothing, \varnothing, \varnothing\rangle$ **then**
    $U \leftarrow C$
    if $\exists(A, \sharp, \varnothing) \in u_{cpt}$ **then**
      $A \leftarrow A \otimes D$
    else
      $u_{cpt} \leftarrow u_{cpt} \cup (D, \sharp, \varnothing)$
    end if
  else
    if $VarMap = \varnothing$ **then**
      $U \leftarrow b_0$
    else
      $U \leftarrow C$
      for all $(X_1, Rel, X_2) \in u_{cst}$ **do**
        $Y_1 \leftarrow getY(VarMap, X_1)$
        $Y_2 \leftarrow getY(VarMap, X_2)$
        $Rel \leftarrow Rel \cup getRel(d_{cst}, Y_1, Y_2)$
      end for
      for all $(C, X, \varnothing) \in u_{cpt}$ **do**
        $Y \leftarrow getY(VarMap, X)$
        $C \leftarrow C \otimes getCpt(u_{cpt}, Y)$
      end for
    end if
  end if
end if
return U

Il y a une équivalence entre la subsomption descriptive et la subsomption structurelle. Pour deux concepts C et D de $TL-JClassic_{\delta\epsilon}$, une subsomption descriptive entre C et D implique une subsomption structurelle entre leurs formes normales et s'il existe un mappage de variables $\mathcal{M}$ où l'union des formes normales de C et D est équivalente à C en $TL-CL_{\delta\epsilon}$, alors il existe une correspondance entre leurs réseaux de contrainte temporelle et leurs concepts non temporels sous le même mappage de variables.
L'équivalence entre la subsomption descriptive et la subsomption structurelle reflète la correction et la complétude de la sémantique intentionnelle de $TL-CL_{\delta\epsilon}$ conformément au système EQ défini pour la logique $TL-JClassic_{\delta\epsilon}$. L'équivalence entre la subsomption descriptive et la subsomption structurelle est exprimée formellement par le théorème suivant :

Théorème 1. *Soient C et D deux concepts de* $TL-JClassic_{\delta\epsilon}$. $C \sqsubseteq_d D \Leftrightarrow C \sqsubseteq_s D.$

4.4 Les algorithmes d'inférence de $TL-JClassic_{\delta\epsilon}$

La relation de subsomption et la relation d'héritage sont utilisées pour déduire de nouvelles connaissances temporelles dans notre système déductif, le calcul de la relation de subsomption entre chaque pair de concepts temporels est utilisé pour classer les concepts, alors que la relation d'héritage sert de base pour récupérer les propriétés temporelles héritées, donc ces deux relations sont les plus importantes dans notre raisonneur développé pour $TL-JClassic_{\delta\epsilon}$.

4.4.1 Algorithme de calcul de subsomption

Cet algorithme s'appuie sur le calcul de la subsomption structurelle décrit dans la section 4.3.2. Comme nous avons mentionné précédemment, la subsomption entre deux concepts temporels C et D revient à vérifier s'il y a au moins un mappage de variables qui satisfait l'équivalence entre $C \sqcap D$ et C. Il y a deux approches possibles [92, 93] pour chercher s'il existe ce mappage de variables entre les deux concepts :

— Une approche naïve qui consiste à générer un mappage de variable entre deux concepts et vérifier si l'union de leurs formes normales sous ce mappage permet d'avoir une subsumption entre les deux concepts. Si cela n'est pas le cas, on génère un autre mappage et on répète le processus jusqu'à tester toutes les combinaisons possible de mappage de variables entre les deux concepts. Soient $\mid c_{\delta t_{var}} \mid = n_1$ et $\mid D_{\delta t_{var}} \mid = n_2$, le nombre de mappage de variables potentiel de D à

C est $n_2^{n_1}$, il est clairement un NP problème dans le pire des cas.

— Un prétraitement avant de calculer, selon cette approche, l'ensemble de mappage de variable possible entre deux concepts peut être modélisé comme un problème de satisfaction de contrainte binaire (CSP). Un CSP binaire se compose d'un ensemble fini de variables, chacune d'elles dispose d'un ensemble fini de valeurs possibles (domaine) ainsi qu'un ensemble de contraintes binaires sur ces variables. Résoudre un CSP consiste à chercher une instanciation de toutes les variables à des valeurs dans leurs domaines respectifs qui satisfait toutes les contraintes. Il est peu probable qu'un algorithme polynomial existe pour résoudre un CSP mais un travail considérable a été fait pour tenter de le résoudre dans un délai raisonnable. Les travaux de Mackworth [94, 93] ont prouvé que les algorithmes de consistance (Un algorithme de consistance permet de propager les contraintes dans chaque domaine de variable et d'assurer sa cohérence par supprimer toutes les inconsistances qui ne vérifient pas les contrainte locales) sont des méthodes efficaces pour réduire l'espace de recherche de solutions. C'est vrai que ces algorithmes ne résoudraient pas nécessairement un CSP mais ils éliminent les valeurs qui ne pourraient pas faire partie de toute solution. Il est donc approprié d'adopter cette approche pour calculer la subsomption dans notre système.

Nous définissons $TL-Sub_{\delta\epsilon}(C,D)$, une procédure de calcul de subsomption entre deux concepts temporels de $TL-JClassic_{\delta\epsilon}$ (est présenté dans l'algorithme 3). Elle comprend trois étapes, la normalisation, le prétraitement, et la comparaison, nous détaillons chaque étape dans les sections suivantes.

Algorithm 3 $TL-Sub_{\delta\epsilon}(C,D)$

Require: C and D two concepts of $TL-JClassic_{\delta\epsilon}$
Ensure: Response "Yes" or "No" to question "Is C subsumed by D ?"
{Compute normal forms}
$NF\_C \leftarrow Normalization(C)$
$NF\_D \leftarrow Normalization(D)$
{Treatment of bottom($\perp$)}
if $NF\_C = b_0$ **then**
return "Yes"
else if $NF\_D = b_0$ **then**
return "No"
else
VarMapSet← FindVarMap($NF\_C_{\delta t}$,$NF\_D_{\delta t}$)
$Response \leftarrow "No"$
while $(VarMapSet \neq \varnothing)\&(Response = "No")$ **do**
$Let(\mathcal{M} \in VarMapSet)$
$NF\_CD \leftarrow NF\_C \otimes_{\mathcal{M}} NF\_D$
{Comparison of the normal forms}
$Compar(NF\_C_{\theta}, NF\_CD_{\theta}, Response)$
if $Response = "Yes"$ **then**
$Compar(NF\_C_{\delta}, NF\_CD_{\delta}, Response)$
end if
$VarMapSet \leftarrow VarMapSet - \{\mathcal{M}\}$
end while
return Response
end if

— **1. Normalization** : les formes normales de C et D sont calculées en utilisant la procédure de normalisation (algorithm 4). Cette procédure permet de calculer la forme normale d'un concept à partir de sa description. Si le concept est temporel, il est nécessaire d'appeler les fonctions suivantes pour compléter le calcul de sa forme normale :

$Expand(C_t)$ (presenté dans l'algorithme 6) : vu que le concept temporel peut inclure des concepts macro dans leur ensemble des concepts qualificatifs temporels, alors cette fonction développe chaque concept macro en lui substituant récursivement ses éléments constitutifs dans sa forme normale après avoir fait la substitution des variables temporelles requises.

$TransitiveClosure(C_{t_{cst}})$ Après avoir étendu tout les concepts macro dans le concept temporel, l'ensemble des contraintes temporelles peut être incomplètes ou incohérentes, qui peut influer négativement sur la validité de l'inférence. Pour cela, nous appelons la fonction $TransitiveClosure$ définie dans

l'algèbre d'Allen afin de calculer toutes les relations temporelles implicites qui découle des contraintes temporelles existantes et cette fonction permet également de détecter les incohérences si présente.

Algorithm 4 $Normalization(C)$

Require: C is a concept of $TL - JClassic_{\delta\epsilon}$ ($C \equiv T_1 \sqcap T_2 \sqcap ... \sqcap T_n, n \geq 1$)
Ensure: NF_C : normal form of C in $TL - CL_{\delta\epsilon}$
  if (C is $\top$) | (C is $\bot$) | (C is P) | (C is ONE-OF E) | (C is Min u) | (C is Max u) | (C is r FILLS E (E $\nexists \varnothing$)) | (C is r FILLS $\varnothing$) | (C is r AT-LEAST n (n $\geq$ 1)) | (C is r AT-LEAST 0) | (C is r AT-MOST n (n $\geq$ 1)) | (C is r AT-MOST 0) | (C is $\forall r : \top$) | (C is $\forall r : \bot$) **then**
    return $SimpleTerme(C)$
  else
    return $CompTerme(C)$
  end if

Algorithm 6 $expand(C_t)$

Require: $C_t\langle c_{var}, c_{cst}, c_{cpt}\rangle$ is a temporal part in 7-uplet and c_{cpt} may include a set of temporal concepts written in the form (d, x, ren) where d is a normal form of complex concept, the interval $x \in c_{var}$ and the set ren which represent the temporal variables' substitution list in d.
Ensure: All the temporal concepts in c_{cpt} is written in the form $(a, x, \varnothing)$ where a is a normal form of non temporal concept.
  if $\langle c_{var}, c_{cst}, c_{cpt}\rangle \neq \langle\varnothing, \varnothing, \varnothing\rangle$ **then**
    for all $(a, x, ren) \in c_{cpt}$ **do**
      if $a_t \neq \langle\varnothing, \varnothing, \varnothing\rangle$ **then**
        $a_t \leftarrow expand(a_t)$
        {Temporal variables substitution}
        $a'_t \leftarrow renameVar(a_t, ren)$
        Let $a'_t = \langle a'_{var}, a'_{cst}, a'_{cpt}\rangle$
        $c_{var} \leftarrow c_{var} \cup a'_{var}$
        $c_{cst} \leftarrow c_{cst} \cup a'_{cst}$
        $c_{cpt} \leftarrow c_{cpt} - \{(a, x, ren)\}$
        $c_{cpt} \leftarrow c_{cpt} \cup a'_{cpt}$
      end if
    end for
    if $\exists(x_1, rel1, x_2), (x_1, rel2, x_2) \in c_{cst}$ **then**
      $c_{cst} \leftarrow c_{cst} - \{(x_1, rel1, x_2), (x_1, rel2, x_2)\}$
      $c_{cst} \leftarrow c_{cst} \cup \{(x_1, rel1 \cup rel2, x_2)\}$
    end if
    if $\exists(a1, x, \varnothing), (a2, x, \varnothing) \in c_{cpt}$ **then**
      $c_{cpt} \leftarrow c_{cpt} - \{(a1, x, \varnothing), (a2, x, \varnothing)\}$
      $c_{cpt} \leftarrow c_{cpt} \cup (a1 \otimes a2, x, \varnothing)$
    end if
  end if
  return C_t

— **2. Prétraitement (calcul de l'ensemble de mappage de variables possible)** :

Algorithm 5 $CompTerme(C)$

Require: C is a concept of $TL-JClassic_{\delta\epsilon}$
Ensure: NF_C : normal form of C in $TL-CL_{\delta\epsilon}$

if C is $\forall r : D$ **then**
$NF\_D \leftarrow Normalisation(D)$
if $NF\_D = \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing,\varnothing,\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing,\varnothing,\varnothing)\rangle$ **then**
$NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing,\varnothing,\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing,\varnothing,\varnothing)\rangle$
else if $NF\_D = b_0$ **then**
$NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing, \{\langle r,\varnothing,0,0,b_0\rangle\},\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing, \{\langle r,\varnothing,0,0,b_0\rangle\},\varnothing)\rangle$
else if $NF\_D_{\delta t} = \langle\varnothing,\varnothing,\varnothing\rangle$ **then**
$NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing, \{\langle r,\varnothing,0,NF\_D_{\theta.dom}, NF\_D\rangle\},\varnothing), (\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing, \{\langle r,\varnothing,0,NF\_D_{\theta.dom}, NF\_D\rangle\},\varnothing)\rangle$.
else
$NF\_C \leftarrow b_0$
end if
end if
if C is $\Diamond X_1,..,X_n.(X_1 R_{(1,2)} X_2),..,(X_{n-1} R_{(n-1,n)} X_n)\ D_0@\sharp\sqcap(D_1@[Y_1]@X_1...@[Y_k]@X_k)@X_1 \sqcap ... \sqcap (D_n@[Y_1]@X_1...@[Y_k]@X_k)@X_n$ **then**
for all i **do**
$NF\_d_i \leftarrow Normalisation(D_i)$
end for
$NF\_D \leftarrow \langle (\langle \{x_1,..,x_n\}_{var}, \{(x_1, rel_{(1,2)}, x_2),..,(x_{n-1}, rel_{(n-1,n)}, x_n)\}_{cst}, \{(NF\_d_i, x_i, \{(y_j,x_j)\}_{ren})\}_{cpt} \rangle, Univ, Min-R, Max-R, \varnothing, \varnothing,\varnothing), (\langle \{x_1,..,x_n\}_{var}, \{(x_1, rel_{(1,2)}, x_2),..,(x_{n-1}, rel_{(n-1,n)}, x_n)\}_{cst}, \{(NF\_d_i, x_i, \{(y_j,x_j)\}_{ren})\}_{cpt} \rangle, Univ, Min-R, Max-R, \varnothing, \varnothing, \varnothing)\rangle$
$NF\_D_{\theta t} \leftarrow expand(NF\_D_{\theta t})$
$NF\_D_{\delta t} \leftarrow expand(NF\_D_{\delta t})$
$NF\_D_{\theta cst} \leftarrow TransitiveClosure(NF\_D_{\theta cst})$
$NF\_D_{\delta cst} \leftarrow TransitiveClosure(NF\_D_{\delta cst})$
if $(NF\_D_{\theta cst} = \varnothing) \| (NF\_D_{\delta cst} = \varnothing)$ **then**
$NF\_C \leftarrow b_0$
else
$NF\_C \leftarrow NF\_D$
end if
end if
if C is δ D **then**
$NF\_D \leftarrow Normalisation(D)$
$NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing,\varnothing,\varnothing), NF\_D_\delta\rangle$.
end if
if C is D^ϵ **then**
$NF\_D \leftarrow Normalisation(D)$
$NF\_C \leftarrow \langle(\langle\varnothing,\varnothing,\varnothing\rangle, Univ, Min-R, Max-R, \varnothing,\varnothing,\varnothing, NF\_D_\delta),$
$(NF\_D_{\delta t}, NF\_D_{\delta dom}, NF\_D_{\delta max}, NF\_D_{\delta min}, NF\_D_{\delta\pi}, NF\_D_{\delta r}, NF\_D_{\theta\epsilon} \cup NF\_D_{\delta\epsilon})\rangle$.
end if

if C is $D_1 \sqcap D_2$ **then**
$NF\_D1 \leftarrow Normalisation(D_1)$
$NF\_D2 \leftarrow Normalisation(D_2)$
{$NF\_C \leftarrow NF\_D1 \otimes_{\mathcal{M}} NF\_D2$}
$NF_C_\theta \leftarrow$unionUple(NF_D1_θ,NF_D2_θ,$\mathcal{M}$)
$NF_C_\delta \leftarrow$unionUple(NF_D1_δ,NF_D2_δ,$\mathcal{M}$)
end if
return NF_C

la fonction $FindVarMap$ (présenté dans l'algorithme 7) sert à réduire le nombre potentielles de mappage de variables entre les formes normales obtenues. Elle définis d'abord les domaines de valeurs possibles pour chaque concept non temporel dans $D_{\delta t_{cpt}}$ à partir de l'ensemble des concepts non temporels existants en $C_{\delta t_{cpt}}$, si un des domaines obtenu est un ensemble vide la fonction retourne faux, sinon, elle utilise les contraintes temporelles des concepts pour appliquer la propagation de contraintes afin de réduire les domaines des variables en filtrant les valeurs incohérentes des domaines. Si un domaine de n'importe quelle variable après le prétraitement est vide, la fonction retourne un faux.

Algorithm 7 $FindVarMap(C_t, D_t)$

Require: $C_t\langle c_{var}, c_{cst}, c_{cpt}\rangle$, $D_t\langle d_{var}, d_{cst}, d_{cpt}\rangle$ are two expanded temporal concepts.
Ensure: a set of all possible variable mapping from D to C.
$VarMapSet \leftarrow \varnothing$
if $\langle c_{var}, c_{cst}, c_{cpt}\rangle \neq \langle\varnothing,\varnothing,\varnothing\rangle \wedge \langle d_{var}, d_{cst}, d_{cpt}\rangle \neq \langle\varnothing,\varnothing,\varnothing\rangle$ **then**
if $\mid c_{var} \mid >= \mid d_{var} \mid$ **then**
for all $(a, x, \varnothing) \in c_{cpt}$ **do**
$dom(a) \leftarrow \varnothing$
for all $(b, x2, \varnothing) \in d_{cpt}$ **do**
if $a \sqsubseteq b$ **then**
$dom(a) \leftarrow dom(a) \cup \{(b, x2, \varnothing)\}$
end if
end for
if $dom(a) = \varnothing$ **then**
return $\varnothing$
end if
end for
$c' \leftarrow reduce(c_{cst})$
$d' \leftarrow reduce(d_{cst})$
$VarMapSet \leftarrow pathConsistency(Dom, c', d')$
{Dom is a set of all $Dom(a)_i$}
end if
end if
return $VarMapSet$

— **3. Comparaison syntaxique** : après le calcul préalable des mappages de variables potentielles, la forme normale de C ⊓ D est calculée avec la fonction $union$ $\otimes_{\mathcal{M}}$, où $\mathcal{M}$ représente un des choix possible des mappage de variables potentiel. Après cela, l'étape de la comparaison syntaxique entre les formes normales de C et $C \sqcap D$ commence. Si les parties strictes de leurs formes normales sont égaux, alors l'algorithme continue de comparer leurs parties par défaut. La comparaison syntaxique se fait via la procédure Compar (présentée dans l'algorithme 8) qui vérifie l'égalité entre deux 7-uples. Elle fait appel aux deux procédures suivantes :

— La comparaison des rôles permet de vérifier l'égalité des rôles définis dans les formes normales des deux concepts, cette procédure est donnée dans l'algorithme 9.

— La comparaison des propriétés temporelles, qui permet de vérifier l'égalité des champs temporels dans les formes normales des deux concepts. Cette procédure est donné dans l'algorithme 10.

L'algorithme $TL-Sub_{\delta\epsilon}(C, D)$ retourne «Oui», s'il existe un mappage de variables selon lequel les formes normales de C et $C \sqcap D$ sont égales, sinon il renvoie "Non" après avoir essayer tous les mappages potentiels de variables.

Algorithm 8 Procedure Compar(C,D,Response)

Require: C (c_t, c_{dom}, c_{min}, c_{max}, c_π, c_r, c_ϵ) and D (d_t, d_{dom}, d_{min}, d_{max}, d_π, d_r, d_ϵ) are two 7-uplets.

Ensure: Response "Yes" or "No" to question "C equals D ?"

if $(c_{dom} \neq d_{dom}) \,||\, (c_{min} \neq d_{min}) \,||\, (c_{max} \neq d_{max}) \,||\, (c_\pi \neq d_\pi) \,||\, (c_\epsilon \neq d_\epsilon)$ **then**
  $Response \leftarrow "No"$
else
  {Comparison of roles fields}
  Compar-roles (c_r, d_r, Rep)
  if $Rep =$ "Yes" **then**
    {Comparison of temporal fields}
    Compar-temp(c_t, d_t, Rep)
    $Response \leftarrow Rep$
  else
    $Response \leftarrow "No"$
  end if
end if

Algorithm 9 $Compar - roles(C_r, D_r, Response)$

Require: C_r and D_r are sets of roles $\langle r, fillers, least, most, c\rangle$ in the 7-uplets.
Ensure: Response "Yes" or "No" to question "C_r equals D_r ?"

$Response \leftarrow "Yes"$
if $\mid C_r \mid \neq \mid D_r \mid$ **then**
  $Response \leftarrow "No"$
else
  while $(C_r \neq \varnothing) \& (Rep = "Yes")$ **do**
    let e1$\langle r, fillers1, least1, most1, c1\rangle \in C_r$
    search $e2 \in D_r$ with the same name r
    if $e2 \notin D_r$ **then**
      $Response \leftarrow "No"$
    else
      let $e2\langle r, fillers2, least2, most2, c2\rangle$
      if $(\mid fillers1 \mid \neq \mid fillers2 \mid) \,||\, (least1 \neq least2) \,||\, (most1 \neq most2)$ **then**
        $Response \leftarrow "No"$
      else
        {Comparison of strict and default 7-uples of c1 and c2}
        $Compar(c1_\theta, c2_\theta, Rep)$
        if $Rep = "Yes"$ **then**
          $Compar(c1_\delta, c2_\delta, Rep)$
          $Response \leftarrow Rep$
        else
          $Response \leftarrow "No"$
        end if
      end if
    end if
    $C_r \leftarrow C_r - \{e1\}$
    $D_r \leftarrow D_r - \{e2\}$
  end while
end if

Algorithm 10 $Compar - temp(C_t, D_t, Response)$

Require: $C_t\langle c_{var}, c_{cst}, c_{cpt}\rangle$, $D_t\langle d_{var}, d_{cst}, d_{cpt}\rangle$ are two temporal fields.
Ensure: Response "Yes" or "No" to question "C_t equals D_t ?"

```
Response ← "Yes"
if ⟨c_var, c_cst, c_cpt⟩ ≠ ⟨∅, ∅, ∅⟩ then
  if ⟨d_var, d_cst, d_cpt⟩ = ⟨∅, ∅, ∅⟩ then
    Rep ← "No"
    if (c_var ≠ d_var) || (c_cst ≠ d_cst) then
      Response ← "No"
    end if
    for all (a1, x, ∅) ∈ c_cpt) do
      search (a2, x, ∅) ∈ d_cpt with the same interval x
      if (a2, x, ∅) ∉ d_cpt then
        Response ← "No"
      else
        {Comparison of strict and default 7-uples of a1 and a2}
        Compar(a1_θ, a2_θ, Rep)
        if Rep = "Yes" then
          Compar(a1_δ, a2_δ, Rep)
          Response ← Rep
        else
          Rep ← "No"
        end if
      end if
    end for
  end if
else if ⟨d_var, d_cst, d_cpt⟩ ≠ ⟨∅, ∅, ∅⟩ then
  Rep ← "No"
end if
```

4.4.2 Algorithme d'héritage

La tâche principale de la relation d'héritage dans notre système est de retirer les exceptions de la dénotation d'un concept (les deux parties de ϵ dans la forme normale d'un concept) pour avoir les propriétés temporelles héritées pour ce concept. La procédure est donnée dans l'algorithme 11, il se compose des étapes suivantes :

1. Remplacer chaque exception de niveau pair par un défaut $(C^{\epsilon})^{\epsilon} = \delta C$.
2. Pour chaque variable temporelle dans la forme normale d'un concept, appeler récursivement la procédure d'héritage avec une restriction de valeur de la variable temporelle.
3. Pour chaque rôle dans la forme normale d'un concept, appeler récursivement la procédure d'héritage avec une restriction de valeur de rôle.
4. Supprimer P (resp. $\neg P$) dans $c_{\delta\pi}$ si $\neg P$ (resp. P) est dans $c_{\theta\pi}$.

La dénotation résultat est la forme d'héritage du concept. Les propriétés héritées sont celles trouvées dans le schéma d'héritage, les propriétés strictes (resp. propriétés défauts) sont celles qui se trouvent dans la partie stricte (resp. partie défaut).

Algorithm 11 $inheritance(C)$

Require: C is a normal form of concept $TL-JClassic_{\delta\epsilon}$
  $res \leftarrow \langle(\varnothing, c_{\theta\pi}, \varnothing, \varnothing), (\varnothing, c_{\delta\pi}, \varnothing, \varnothing)\rangle$
  for all $y \in c_{\theta\epsilon} \cup c_{\delta\epsilon}$ **do**
    $res \leftarrow res \cup transform(y, c_{\theta\epsilon})$
  end for
  for all $\langle a, x\rangle \in c_{\theta t}$ **do**
    res$\leftarrow$ res$\cup\langle(\langle inheritance(a), x\rangle, \varnothing, \varnothing, \varnothing), (\varnothing, \varnothing, \varnothing, \varnothing)\rangle$
  end for
  for all $\langle a, x\rangle \in c_{\delta t}$ **do**
    res$\leftarrow$ res$\cup\langle(\varnothing, \varnothing, \varnothing, \varnothing), (\langle inheritance(a), x\rangle, \varnothing, \varnothing, \varnothing)\rangle$
  end for
  for all $\langle r, p\rangle \in c_{\theta r}$ **do**
    res$\leftarrow$ res$\cup\langle(\varnothing, \varnothing, \langle r, inheritance(p)\rangle, \varnothing), (\varnothing, \varnothing, \varnothing, \varnothing)\rangle$
  end for
  for all $\langle r, p\rangle \in c_{\delta r}$ **do**
    res$\leftarrow$ res$\cup\langle(\varnothing, \varnothing, \varnothing, \varnothing), (\varnothing, \varnothing, \langle r, inheritance(p)\rangle, \varnothing)\rangle$
  end for
  return res

4.5 La complexité des algorithmes d'inférence de $TL-JClassic_{\delta\epsilon}$

Bien que la décidabilité indique en théorie que l'on peut toujours répondre correctement au problème d'un raisonnement, mais à elle seule, n'est pas suffisante, le temps nécessaire pour obtenir la solution est aussi primordial, cela se fait en évaluant la complexité de l'algorithme du raisonnement. La complexité d'un algorithme est définie en donnant un ordre de grandeur du nombre d'opérations élémentaires à effectuer pour obtenir la réponse en fonction de la taille des données manipulées. Les ordres de grandeurs sont indiqués par des classes de complexité, par exemple complexité polynomiale, complexité exponentielle, etc. Notre système est principalement basé sur les procédures de subsomption et d'héritage. L'algorithme de subsomption $TL-Sub_{\delta\epsilon}$ repose sur un ensemble de procédures dont nous prouvons leur traçabilité dans cette section et nous montrons aussi la traçabilité de la procédure d'héritage.
La preuve nécessite des preuves de l'autre résultat présenté comme lemme.

Lemme 1. *La taille d'une forme normale d'un terme temporel de* $TL-JClassic_{\delta\epsilon}$ *est polynomiale en fonction de la taille du terme.*

Preuve

1. La taille de la forme normale d'un terme polynomiale sans défaut et exception est polynomiale

Soit C une description du concept temporel sans les connecteurs (δ et ϵ) et FN_C est sa forme normale. Pour montrer que la taille de la forme normale FN_C est polynomiale en utilisant la taille de C, nous représentons FN_C sous la forme d'un graphe complet ($Graph - FN$) auxquels on associe l'ensemble des concepts $c_{t_{cpt}}$ défini dans FN_C aux nœuds et l'ensemble des $c_{t_{cst}}$ aux arcs. Donc chaque nœud dans le graphe ($Graph{-}FN$) est étiqueté par (c_i, x_i, ren_i) où c_i est un concept non temporel, x_i variable temporelle et $ren = \varnothing$ et chaque arc est représenté par ($x_i\ rel_{(i,i')}\ x'_{i'}$). nous représentons chaque concept non temporel dans les nœuds sous la forme d'un arbre $Tree - NF$, la structure de l'arbre et la procédure qui permet de la construire sont présentées en détail dans [4]. La taille d'une forme normale C est égale à la taille du $Graph - NF_C$. La taille d'un $Graph - NF$ est la somme de la taille de toutes les étiquettes dans leurs arcs et leurs nœuds. La taille des étiquettes dans $Graph - NF$ sont définis comme suit :

- La taille de chaque ensemble de relation (rel) est sa cardinalité (≤13).
- La taille de chaque concept non temporel est la taille de son arbre-NF.
- La taille de chaque variable temporelle est 1.
- La taille de l'ensemble de ren est 1.

Donc la taille de chaque nœud est $|Arbre - NF_c|$+ 2 et la taille de chaque arc est 2 + $|rel|$.

Soit nn le nombre des nœuds dans le $Graph - FN$, ac la moyenne de cardinalité de la taille d'un arbre qui représente le concept non temporel c_i dans un nœud, et on sait que la cardinalité maximum de $|rel|$ est 13, alors on obtient :

taille ($Graph - NF_C$) = $(ac + 2) * nn + 15 * nn^2$. ($nn^2$ le nombre des arcs dans un graphe complet)

Pour montrer que la taille de FN_C est polynômiale si la taille de C est polynômiale, on écrit ac et nn en fonction des connecteurs qui existent dans la description de C.

Le nn est limité par le nombre de variables temporelles introduites par le quantificateur existentiel temporelle ($\Diamond$) et les connecteurs qualificateur de substitution ($[Y]@X$) qui ont inclus dans la description de C.

Le ac est limité par le nombre des connecteurs ($\forall r : A$) inclus dans les concepts non temporels qui sont qualifiés dans la description de C.

Si C est polynomiale, cela signifie que le nn et ac sont limitées et par conséquent la taille de sa forme normale (FN_C) est polynomiale.

2- La taille de la forme normale d'un terme polynomiale de $TL - JClassic^{+}_{\delta\epsilon}$ est polynomiale

La forme normale d'un concept avec la description δC est ⟨ (⟨ ∅,∅,∅ ⟩, $Univ$,$Min - R$,$Max - R$,∅,∅,∅), C_δ ⟩. La taille de cette forme normale est limitée par la taille de la forme normale du concept C, autrement dit, l'interprétation du connecteur δ n'augmente pas la taille de la forme normale.

La forme normale du concept C^ϵ est ⟨ (⟨ ∅, ∅, ∅ ⟩, $Univ$, $Min - R$, $Max - R$, ∅,∅, ∅, C_δ), ($c_{\delta.t}$, $c_{\delta.dom}$, $c_{\delta.max}$, $c_{\delta.min}$, $c_{\delta\pi}$, $c_{\delta r}$, $c_{\theta\epsilon} \cup c_{\delta\epsilon}$)⟩. La taille de la forme normale obtenue est strictement supérieure à la taille du concept C. Cet accroissement est dû à la saturation partielle que l'on fait pour représenter l'exception (ϵ). On doit vérifier que lorsqu'on itère n fois l'application de cette interprétation, la taille de la dénotation reste bornée par un polynôme. On calcule la taille de la forme normale d'un terme de taille n+1. Le pire des cas est celui où le terme est de la forme $(C^\epsilon)^n$.

La forme normale $fn((C^\epsilon)^n)$ obtenue par l'application de l'algorithme de normalisation est la suivante : $FN((C^\epsilon)^n)$= ⟨ (⟨ ∅, ∅, ∅ ⟩, $Univ$, $Min - R$, $Max - R$, ∅,∅, ∅, C_δ), ($c_{\delta.t}$, $c_{\delta.dom}$, $c_{\delta.max}$, $c_{\delta.min}$, $c_{\delta\pi}$, $c_{\delta r}$, $\{n, n-1, ..., 1\}$)⟩ avec les index suivants :

- 1 : ($c_{\delta.t}$, $c_{\delta.dom}$, $c_{\delta.max}$, $c_{\delta.min}$, $c_{\delta\pi}$, $c_{\delta r}$, ∅)
- 2 : ($c_{\delta.t}$, $c_{\delta.dom}$, $c_{\delta.max}$, $c_{\delta.min}$, $c_{\delta\pi}$, $c_{\delta r}$, $\{1\}$)
- 3 : ($c_{\delta.t}$, $c_{\delta.dom}$, $c_{\delta.max}$, $c_{\delta.min}$, $c_{\delta\pi}$, $c_{\delta r}$, $\{2, 1\}$)
- .
- .
- .
- n+1 : ($c_{\delta.t}$, $c_{\delta.dom}$, $c_{\delta.max}$, $c_{\delta.min}$, $c_{\delta\pi}$, $c_{\delta r}$, $\{n, n-1, ..., 1\}$)

La taille de la forme normale obtenue est de l'ordre de $n+1+taille(fn(C))/2$. La taille de la forme normale d'un terme de la forme $(C^\epsilon)^n$ est donc en $O(n)$. Donc, la taille d'une forme normale d'un concept polynomiale de $TL - JClassic^{+}_{\delta\epsilon}$ est polynomiale.

Lemme 2. *Le calcul d'une forme normale à partir d'un concept temporel de $TL - JClassic^{+}_{\delta\epsilon}$ de taille polynomiale se fait en un temps polynomial, si l'on adopte un fragment polynôme de l'algèbre d'Allen.*

Le calcul de la forme normale est réalisé par l'algorithme $Normalization(C)$. On

considère l'opération de mise à jour de la forme normale comme l'opération fondamentale dans cet algorithme. Il est clair que la complexité de l'algorithme $Normalization(C)$ dépend de la taille et du type des termes donnés en entrée. Pour calculer le coût en temps de l'algorithme $Normalization(C)$ appliqué sur une description temporel, on effectue les étapes suivantes :

— On met la description C sous forme d'un graphe étiqueté orienté complet.

— On calcule le coût en temps de l'algorithme appliqué sur le graphe obtenu.

Le graphe est défini par un couple $G = (V, E)$, où V est l'ensemble des nœuds représenté sous la forme (c_i, x_i, ren_i) et E est l'ensemble des arcs représenté par ($x_i\ rel_{(i,i')}\ x'_{i'}$). La complexité en temps de l'algorithme $Normalization(C)$ sur le graphe G est la somme de la complexité en temps de l'algorithme sur l'ensemble des nœuds et la complexité en temps de l'algorithme sur l'ensemble des arcs (Coût-Normalisation(G)= Coût-Normalisation-nœuds(V) + Coût-Normalisation-arc(E)).

1. **Le coût de l'algorithme $Normalization(C)$ sur les nœuds (c_i, x_i, ren_i) :**
   La mise à jour se fait sur l'élément t de la forme normale, on trouve deux cas :

   (a) *Si la description du concept qualifie (c_i) est non temporel* dans ce cas c_i sera directement représenté selon sa description sous forme d'un arbre binaire étiqueté et le cout de l'algorithme $Normalization(C)$ appliqué sur cet arbre a déjà été montré dans [69] qu'il est polynomiale en fonction de la taille de c_i.

   (b) *Si la description du concept qualifié (c_i) est temporel* dans ce cas c_i sera représenté d'abord sous forme d'un graphe, et ensuite la procédure $Expand$ effectuera un certain nombre de mises à jour à la forme normale précédente pour simplifier c_i. Donc la procédure $Expand$ doit parcourir tout les nœuds du graphe de c_i et faire les mises à jour nécessaires. Si la taille de c_i est polynomiale, la procédure $expand$ est donc de complexité temporelle polynomiale.

   Donc le coût de l'algorithme $Normalization(C)$ sur les nœuds (c_i, x_i, ren_i) reste polynomiale dans les deux cas.

2. **Le coût de l'algorithme $Normalization(C)$ sur les arcs ($x_i\ rel_{(i,i')}\ x'_{i'}$) :**
   La mise à jour de la forme normale se fait sur l'élément t_{cst} , cela revient à calculer la fermeture transitive des contrainte temporelle pour avoir un graphe complet et cohérent. Pour les contraintes temporels, la fermeture transitive dans l'algèbre

d'Allen est un problème NP-complet [85], si on considère qu'un fragment polynôme de l'algèbre d'Allen, dans ce de cas, la complexité de cet algorithme sera $O(n^3)$ [84].

Il est à noter que cet algorithme s'arrête s'il détecte une incohérence et il retourne comme résultat de Normalisation(c) la dénotation de b_0.

Lemme 3. *La comparaison de deux formes normales se fait en temps polynomial.*

Preuve

La comparaison de deux formes normales se fait avec la procédure $Compar$. Cette procédure fait une comparaison syntaxique de deux formes normales et s'arrête dés qu'elle rencontre une différence. La pire des cas est donc le cas ou il y a égalité des formes normales à comparer car on doit parcourir tous les éléments des formes normales. Dans ce cas, la complexité en temps de la procédure est dépend de la taille des formes normales, cette taille a déjà été preuvé être polynomiale en fonction de sa description, donc on conclue que la comparaison de deux formes normales s'effectue en temps polynomial.

Proposition 1 : *L'algorithme $TL - Sub_{\delta\epsilon}$ est de complexité polynomial si l'on adopte un fragment polynôme de l'algèbre d'Allen.*

Preuve : nous avons :

- La taille d'une forme normale correspondant à une description d'un concept polynomiale est polynomiale.
- Le calcul d'une forme normale d'un concept de $TL - JClassic_{\delta\epsilon}$ pour un fragment polynôme de l'algèbre d'Allen est fait en temps polynomial.
- La complexité de la fonction (Max-restricted path consistency) utilisé pour faire le prétraitement et le filtrage de l'ensemble de mappage de variables est $O(end^3)$, où e le nombre de contraintes, n le nombre de variable définie dans des domaines de taille au plus d [95].
- La comparaison de deux formes normales se fait en temps polynomiale

Nous en déduisons que l'algorithme pour calculer la subsomption $TL - Sub_{\delta\epsilon}$ est de complexité polynomiale s'il y a une restriction sur l'expressivité des contraintes qualitatives.

Nous prouvons maintenant que l'héritage est aussi de complexité polynomiale.

Proposition 2 : *La complexité de l'héritage est polynomiale.*

Preuve : l'algorithme de calcul d'héritage est principalement traverse la structure d'une forme normale (par les variables temporelles et les rôle) avec une simplification directe en cas d'exceptions en paire. Le calcul d'héritage est donc de complexité polynomiale.

4.6 La mise en œuvre de $TL - JClassic_{\delta\epsilon}$

Afin qu'on puisse profiter de notre logique dans les systèmes à bases de connaissances, nous avons mis en œuvre l'outil $TL-JClassic_{\delta\epsilon}$. Cet outil est développé en Java, il possède une interface utilisateur graphique (GUI) (voir la Figure 4.1) afin de permettre aux utilisateurs de manipuler facilement les connaissances temporelle non monotone, il permet de créer des bases de connaissances et d'ajouter (voir la Figure **??**), modifier ou supprimer des concepts, des rôles, et des individus d'un domaine donné. Il offre également des services inférences usuelles qu'on trouve dans les moteurs d'inférences pour raisonner et déduire de nouvelles connaissances (voir la Figure **??**).

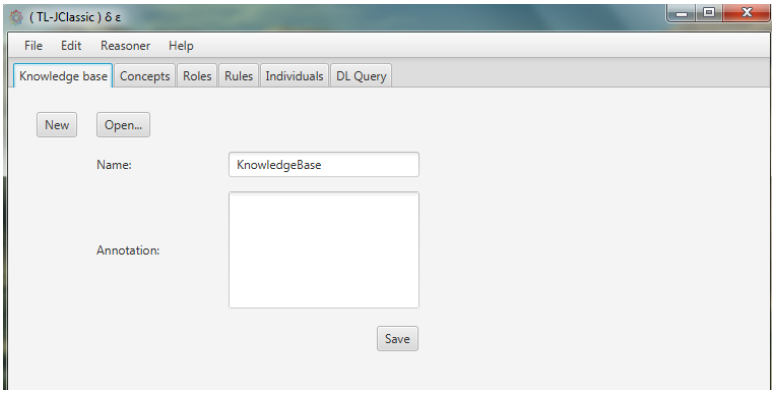

Figure 4.1 – Le raisoneur $TL - JClassic_{\delta\epsilon}$

4.7 Conclusion

Nous avons proposé et développé une logique de description temporelle qui permet de représenter de manière expressive les connaissances temporelles non monotones et de raisonner sur ces connaissances dans un délai raisonnable. Ce nouveau formalisme peut être utilisé dans n'importe quel domaine où les connaissances peuvent être modélisées en une hiérarchie des concepts. Dans le chapitre suivant, nous allons utiliser ce formalisme pour modéliser les règles de sécurité d'une manière expressive et de déduire de manière intelligente les droits d'accès dans un contexte donné.

CHAPITRE 5
UNE POLITIQUE DE SÉCURITÉ CONTEXTUELLE

5.1 Introduction

Les services cloud computing se caractérisent généralement par un grand nombre d'utilisateurs qui interagissent avec un volume important de données soumises à de fortes contraintes de confidentialité et de respect de la vie privée. Les règles d'accès et d'utilisation des données dans ces services doivent donc être clairement définies et respectées. Et vu que l'environnement cloud est dynamique, les règles doivent être également flexibles afin de s'adapter temporairement au changement de contexte. La logique de description temporelle non monotone $TL - JClassic_{\delta\epsilon}$ est capable de modéliser les règles de sécurité d'un modèle de contrôle d'accès contextuel sous forme prototypiques admettant des exception, elle permet aussi de gérer automatiquement les conflits entre les règles de politique afin de prendre une décision d'accès et cela par considérer que les règles les plus spécifiques annulent les règles générales et cela se fait grâce aux services de subsumption et d'héritage du $TL - JClassic_{\delta\epsilon}$.

Pour montrer l'intérêt de notre approche, nous proposons au cours de ce chapitre deux modèles de contrôles d'accès pour les données hébergées dans les SaaS et PaaS, à savoir, un modèle de contrôle d'accès dynamique basé sur le contexte temporel pour les données partagées dans les réseaux sociaux et une extension du modèle de contrôle d'accès basé sur l'organisation afin d'introduire l'objectif d'usage sous forme d'une séquence d'action qui peut inclure des actions facultatives qui peuvent être annulées dans certaines situations spécifiques. Un exemple d'illustration montre comment déduire de manière dynamique la séquence d'actions appropriée qui peuvent être effectuées par l'utilisateur qui demande l'accès aux données sensibles hébergées dans un service cloud sous différents contextes. Ce chapitre se terminera par quelques tests d'évaluation de performances de ce modèle.

5.2 Un nouveau modèle de contrôle d'accès dynamique $TBAC_{\delta\epsilon}$

Le partage des données personnelles est devenu une activité populaire sur les réseaux sociaux en ligne. Un réseau social est généralement un logiciel en tant que service (SaaS), utilisé et géré par différentes entités (le fournisseur, les membres et les applications tierces). Pour cela, l'utilisateur doit contrôler soigneusement qui peut accéder à leurs propres données partagées dans cette plateforme afin de préserver la confidentialité de ses données sensibles. La plupart des réseaux sociaux fournissent un mécanisme de contrôle d'accès basé sur le modèle DAC afin de permettre aux utilisateurs de configurer leurs paramètres de confidentialité. Par exemple dans Facebook, l'utilisateur définit ses préférences de confidentialité pour sa publication via un sélecteur d'audience qui ne prend en charge que cinq modes (public, amis d'amis, amis uniquement, amis spécifiques et moi seulement). Il est clair que ce mécanisme est statiques, pas flexible et pas suffisamment expressif, il ne permet pas à l'utilisateur de spécifier des règles d'autorisation sur la base du temps, de l'emplacement, etc. Il est donc inapproprié pour un système qui regroupe un grand nombre d'utilisateurs et une énorme quantité de données qui représentent principalement la vie réelle des utilisateurs. De plus, les données publiées par les utilisateurs restent néanmoins accessibles en ligne à moins qu'elles ne soient effacées par leurs propriétaires. Au fil du temps, ces données peuvent devenir inappropriées et embarrassantes [10]. Donc, une divulgation non autorisée ou accidentelle de ce type de données peut avoir de graves conséquences au niveau personnel ou professionnel des utilisateurs. Il est important de s'assurer que l'accès à une publication est accessible uniquement aux utilisateurs autorisés.

Le contrôle d'accès pour les données partagées dans les réseaux sociaux est encore un domaine de recherche relativement nouveau qui a attiré beaucoup d'attention récemment. Il existe différents travaux de recherche [96, 97, 98, 9, 99, 100, 101] qui explorent de nouvelles approches et de nouveaux mécanismes pour améliorer la confidentialité des données et répondre à certaines exigences comme par exemple qui peux m'identifier (tag) dans une photo, qui peux lire mes commentaires,... etc. La plupart de ces travaux reposent sur le principe du modèle ABAC, ils proposent une ontologie qui modélise la structure des réseaux sociaux avec le langage OWL et utilisent SWRL pour spécifier les règles d'autorisation qui permettent l'accès aux données partagées. Ces modèles n'offrent pas une flexibilité lors de la définition des règles d'autorisation et ils ne tiennent pas compte du contexte. Cependant, il est plus pratique pour les utilisateurs

de définir le contexte dans lequel leurs données doivent être consultées et plus particulièrement le contexte temporel, par exemple permettre aux utilisateurs de spécifier la période pendant laquelle l'accès est accordé ou refusé à leurs données partagées. Pour faire face à ces limites, nous proposons un modèle basé sur $TL-JClassic_{\delta\epsilon}$ plus dynamique et plus flexible, qui permet aux utilisateurs d'introduire des contraintes temporelles dans les règles d'autorisation associées à leurs données partagées.

5.3 La spécification du modèle $TBAC_{\delta\epsilon}$

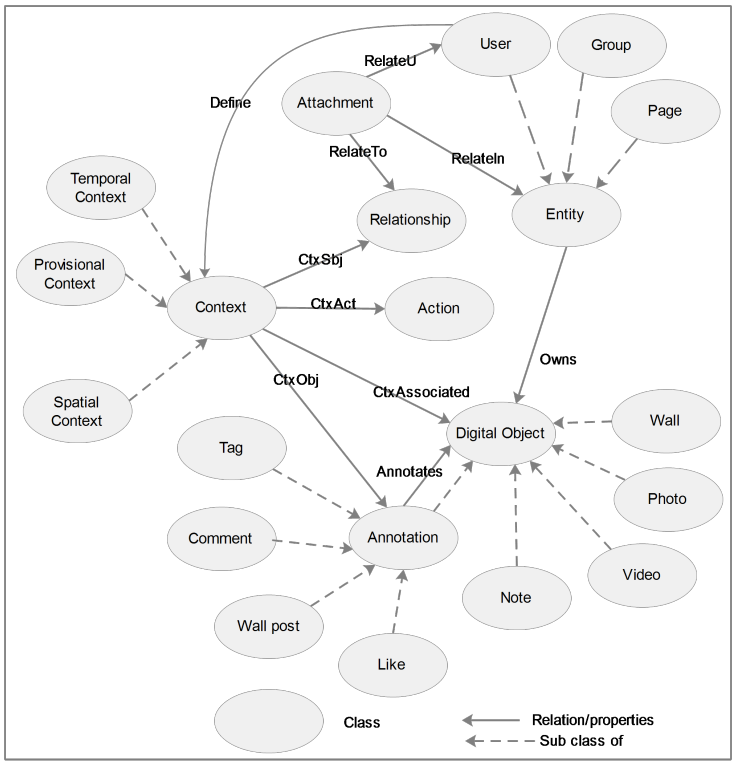

Figure 5.1 – Ontologie basée sur le contexte dans un réseau social

Lorsqu'un utilisateur rejoint un réseau social, il doit créer un profil de lui-même avec des données biographiques, il peut créer ensuite des liens sociaux avec d'autres membres en fonction de leurs choix et de leurs intérêts. Les réseaux sociaux offre souvent à leurs membres une variété de moyens d'interagir entre eux. Un utilisateur peut partager différents types d'informations (photo, vidéo, statut, etc.) avec d'autres utilisateurs, il peut faire des commentaires, des likes, des tags, etc. Il peut également rejoindre des groupes, être fan des pages, et organiser des événements. Toutes les activités réalisées par un utilisateur sont classées par ordre chronologique dans sa Chronologie à travers laquelle les autres utilisateurs ainsi que l'utilisateur lui-même peuvent consulter facilement ses activités passées.

La spécification d'un modèle de contrôle d'accès dans un réseau social est une tâche non triviale en raison du grand nombre d'utilisateurs et de leurs interconnexions. Nous proposons alors une ontologie simple pour modéliser la structure de notre modèle de contrôle d'accès. La Figure 5.1 représente l'ontologie proposée, qui à été inspirée par les travaux existants [102, 103]. Les concepts et leurs relations qui existent dans l'ontologie proposée sont définis comme suit :

Le concept $User$ représente un utilisateur qui a un profil dans le réseau social. Les concepts $Page$ et $Group$ sont introduits pour représenter les pages et groupes qui sont créés par des utilisateurs dans le réseau social. Un utilisateur peut créer des relations avec d'autres entités comme être ami avec un autre utilisateur, fan d'une page, administrateur d'un groupe, etc. pendant une période de temps. Le concept $Attachment$ défini la relation entre un utilisateur avec une autre entité, cela est exprimé par les relations suivantes $RelateU$, $RelateTo$, $relationIn$ avec les concepts $User$, $Relationship$ et $Entity$ respectivement. Le concept $Action$ représente les opérations de base (créer, lire, supprimer, etc.) qui peuvent être effectuée par les utilisateurs dans le systéme. Les concepts $User$, $Group$ et $Page$ sont des entités qui possèdent des objets numériques ($DigitalObject$). Le concept $DigitalObject$ représente n'importe quel objet de contenu numérique, il peut être $Photo$, $Video$, $Note$, $Wall$ et $Annotation$. Le concept $Wall$ représente la page d'accueil d'un profil d'un utilisateur, d'une page ou d'un groupe dans le réseau social. Le concept $Annotation$ est un objet numérique spécial qui représente un contenu numérique annoté avec un autre objet (par exemple, un commentaire annote une vidéo, une étiquette annote une personne sur une photo, etc.) et cela est représenté par la relation $Annotates$ qui relié le concept $Annotation$ par le concept $DigitalObject$. Le concept $Annotation$ peut être $Comment$, Tag, $Like$ et $Wallpost$ (publication). Le concept $Comment$ annote un objet numérique avec une note. Le concept Tag annote un objet numérique avec un profil d'utilisateur, et le concept $Wallpost$ annote un mur avec un objet numérique. Le concept $DigitalObject$ est associé à un contexte qui peut être défini par défaut par le système ou personnalisé par son propriétaire, en d'autres termes, le concept $Context$ représente les conditions qui permettent l'accès à l'objet cible, le concept $Context$ est lié aux concepts $Relation$, $Action$, $Annotation$ et $DigitalObject$ par les relations $CtxSbj$, $CtxAct$, $CtxObj$, $CtxAssociated$ respectivement. Le contexte peut être temporel, spatial ou provisoire [104]. Le concept $TemporalContext$ représente les conditions sur la période pendant laquelle l'utilisateur peut accéder à

la ressource, le concept $SpatialContext$ correspond aux conditions sur l'emplacement de l'utilisateur et le concept $ProvisionalContext$ indique les conditions sur les actions précédentes effectuées par l'utilisateur qui demande l'accès. Dans notre modèle, on s'intéresse au contexte temporelle, nous définissons dans la section suivante une base de connaissances $\mathcal{K}$ pour formaliser notre modèle de contrôle d'accès dynamique en utilsant $TL - JClassic^{+}_{\delta\epsilon}$

La TBox est définie comme suit :

$User \sqsubseteq \top$
$Group \sqsubseteq \top$
$Page \sqsubseteq \top$
$Entity \sqsubseteq \top$
$DigitalObject \sqsubseteq \top$
$Wall \sqsubseteq DigitalObject$
$Photo \sqsubseteq DigitalObject$
$Video \sqsubseteq DigitalObject$
$Note \sqsubseteq DigitalObject$
$Annotation \sqsubseteq DigitalObject$
$Tag \sqsubseteq Annotation$
$Comment \sqsubseteq Annotation$
$WallPost \sqsubseteq Annotation$
$Like \sqsubseteq Annotation$
$Context \sqsubseteq \top$
$TemporalContext \sqsubseteq Context$
$Action \sqsubseteq \top$
$Annotates \sqsubseteq AnnotatesA.Annotation \sqcap$
$AnnotatesD.DigitalObject$
$Owns \sqsubseteq OwnsE.Entity \sqcap$
$OwnsD.DigitalObject$
$Attachment \sqsubseteq RelateU.User \sqcap RelateTo.Relationship$
$\sqcap RelateIn.Entity$
$TemporalContext \sqsubseteq \Diamond Y(YR\sharp)(CtxSbj.Relationship \sqcap$
$CtxAct.Action \sqcap CtxObj.Annotation \sqcap$
$CtxAssociated.DigitalObject)@Y$

Tableau 5.1 – La base de connaissance de $TBAC_{\delta\epsilon}$

5.3.1 La définition des règles de sécurité temporelles

:

— $\delta Context@X \sqsubseteq Attachment \sqcap Action \sqcap Annotates \sqcap Owns \sqcap \delta TemporalContext@X$
Si un utilisateur U a une relation Rel avec l'entité E ($Attachment$), A est une action, Ann est une annotation qui annote l'objet numérique $Dobj$ ($Annotates$), l'objet numérique $Dobj$ appartient à l'entité E ($Owns$), s'il y a un intervalle de

temps X qui vérifie la relation R et que la relation Rel, l'action A, l'annotation Ann et l'objet numérique $Dobj$ sont valide pendant la période X, alors on peut dire que le contexte par défaut est valide à la période du temps X ($\delta Context@X$), et parce qu'une permission concrète peut être déduite d'un contexte par défaut ($is-permitted@X \sqsubseteq \delta Context@X$) , par conséquent, nous pouvons déduire que l'utilisateur U est autorisé à effectuer l'action A sur l'annotation Ann associe à l'objet numérique $Dobj$ pendant la période du temps X.

— $Context^{\epsilon}@X \sqsubseteq Attachment \sqcap Annotates \sqcap Action \sqcap Owns \sqcap TemporalContext^{\epsilon}@X$
S'il existe une exception sur le contexte temporel à l'intervalle de temps X ($TemporalContext^{\epsilon}@X$), et parce qu'une permission concrète ne peut pas être déduite d'un contexte exceptionnel ($is-permitted@X \notin Context^{\epsilon}@X$), nous pouvons déduire que l'utilisateur U est interdit d'effectuer l'action A sur l'annotation Ann associée à l'objet numérique $Dobj$ pendant l'intervalle X.

L'instanciation des concepts et les relations selon la situation actuelle dans le réseau social forme l'ABox de $\mathcal{K}$. Par exemple, Alice considère Bob comme un ami proche dans son profil. Ce fait peut être représenté comme :
Attachment (A1) ⊑ RelateU.User (Bob) ⊓ RelateTo.Relationship (CloseFriend) ⊓ RelateIn.Entity (AliceProfil).
Selon l'application, nous pourrions avoir une ou plusieurs ABox pour une TBox. Dans notre cas, chaque entité dans le réseau social a sa propre ABox.
Notre modèle permet à l'utilisateur d'incorporer des conditions temporelles lorsqu'il spécifie ses paramètres de confidentialité dans son profil. Les règles de sécurité définis permettent d'inférer de nouvelles connaissances, et en fonction de ces informations contextuelles déduites, la décision de contrôle d'accès sera prise.

5.4 La mise en place de $TBAC_{\delta\epsilon}$ dans un réseau social

L'architecture proposée pour la mise en place de notre modèle de contrôle d'accès dans un réseau social est illustrée dans la Figure 5.2, elle est basé sur la norme XACML de OASIS[105]. Cette norme présente un modèle architectural pour l'évaluation des politiques des contrôles d'accès très intéressant qui pourrait être appliquer à n'importe quel modèle de control d'accès. Les principaux composants de cette architecture que nous avons retenus sont : Le point d'application de politiques (Policy Enforcement Point PEP) est responsable de l'interception de toutes les demandes d'accès aux données

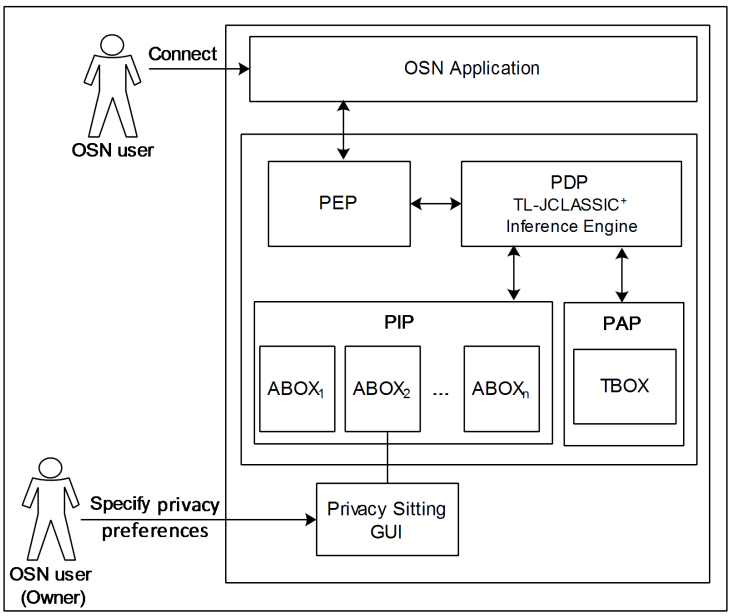

Figure 5.2 – Architecture d'autorisation proposée dans un réseau social

des utilisateurs partagées dans le réseau social, et envoie aux données ciblés si la requête est autorisée. Le point d'administration de politiques (Policy Administration Point PAP) agit comme un dépôt des politiques, il contient la TBOX qui inclut la spécification des règles d'autorisation. Le point d'information de politiques (Policy Information Point PIP) comprend l'ABOX de chaque entité dans le réseau social qui représente la source à partir de laquelle les informations peuvent être récupérées pour être utilisées lors de l'évaluation d'une demande d'accès. Le point de décision de politiques (Policy Decision Point PDP) est le composant qui prend la décision d' autoriser ou non l'accès aux données en utilisant les services d'inférence $TL - JClassic_{\delta\epsilon}$.

Un utilisateur peut définir pour chaque donnée partagée dans le réseau social, les conditions contextuelles sous lesquelles l'accès peut être autorisé, ces conditions seront insérées dans l'ABOX dédié a cet utilisateur. Lorsqu'une demande d'accès provient d'un autre utilisateur, certains faits concernant le contexte actuel de la demande sont temporairement ajoutés à l'ABOX pour être utilisés lors de la décision d'accès. Ensuite, le PDP évalue la demande et renvoie une réponse et en fonction de cette réponse, le PEP autorise ou non l'accès aux données.

5.5 La spécification et la mise en œuvre du modèle $P-OrBAC_{\delta\epsilon}$

Divers entreprises et organisations semblent se diriger vers le cloud computing et particulièrement vers le PaaS afin d'améliorer la qualité et l'accessibilité de leurs services tout en réduisant les coûts. Pour assurer la confidentialité de leurs données sensible dans ce nouveau système, il nous semble intéressant d'encourager les entreprises à définir des politiques de sécurité flexible basé sur l'objectif d'usage. Pour réaliser cet objectif, nous proposons un nouveau modèle basé sur $TL-JClassic_{\delta\epsilon}$ qui étend le modèle OrBAC en intégrant l'objectif d'usage sous forme d'une séquence d'actions.

5.6 La modélisation formelle du $P-OrBAC_{\delta\epsilon}$

L'idée du modèle OrBAC est de spécifier la politique de sécurité au niveau organisationnel. Pour cela, il introduit le rôle, la vue et l'activité en tant que des entités abstraites pour représenter le sujet, l'objet et l'action respectivement dans l'organisation qui est considérée comme l'entité centrale dans ce modèle. Dans cette section, nous définissons une base de connaissances $\mathcal{K}$ pour formaliser notre modèle de contrôle d'accès proposé $P-OrBAC_{\delta\epsilon}$. Notre $TBox$ alors comprend les entités de base qui existent dans le modèle OrBAC qui seront redéfinies avec $TL-JClassic^{+}_{\delta\epsilon}$, et les nouveaux concepts introduits pour représenter l'objectif d'usage, donc notre $TBox$ est présenté comme suit :

- $ORGANIZATION \sqsubseteq \top$
- $SUBJECT \sqsubseteq \top$
- $ROLE \sqsubseteq \top$
- $EMPOWER \sqsubseteq \forall EmpowerO : ORGANIZATION \sqcap \forall EmpowerS : SUBJECT \sqcap \forall EmpowerR : ROLE \sqcap EmpowerO\ AT-LEAST\ 1 \sqcap EmpowerS\ AT-LEAST\ 1 \sqcap EmpowerR\ AT-LEAST 1$

  Le concept $EMPOWER$ est utilisé pour indiquer que l'organisation emploie un sujet dans un rôle.

  $EmpowerO$, $EmpowerS$ et $EmpowerR$ sont les relations binaires définies pour structurer le lien entre les concepts ORGANIZATION, SUBJECT et ROLE comme suit :

  - $EmpowerO$: elle relie le concept EMPOWER au concept ORGANISATION.
  - $EmpowerS$: elle relie le concept EMPOWER au concept SUBJECT.
  - $EmpowerR$: elle relie le concept EMPOWER au concept ROLE.

— $ACTION \sqsubseteq \top$

— $ACTIVITY \sqsubseteq \top$

— $CONSIDER \sqsubseteq \forall considerO : ORGANIZATION \sqcap \forall considerA : ACTION \sqcap \forall considerAct : ACTIVITY \sqcap considerO\ AT-LEAST\ 1 \sqcap considerA\ AT-LEAST\ 1 \sqcap considerAct\ AT-LEAST\ 1$

Le concept $CONSIDER$ est utilisé pour indiquer que l'organisation considère une action faisant partie d'une activité.

$considerO$, $considerA$ et $considerAct$ sont les relations binaires introduites pour structurer le lien entre ORGANIZATION, OBJECT and VIEW, comme suit :

— $considerO$: elle relie le concept CONSIDER au concept ORGANIZATION.

— $considerA$: elle relie le concept CONSIDER au concept ACTION.

— $considerAct$: elle relie le concept CONSIDER au concept ACTIVITY.

— $OBJECT \sqsubseteq \top$

— $VIEW \sqsubseteq \top$

— $USE \sqsubseteq \forall useOr : ORGANIZATION \sqcap \forall useO : OBJECT \sqcap \forall useV : VIEW \sqcap useOr\ AT-LEAST\ 1 \sqcap useO\ AT-LEAST\ 1 \sqcap useV\ AT-LEAST\ 1$

Le concept USE est utilisé pour indiquer que l'organisation utilise un objet dans une vue.

$useOr$, $UseO$ et $UseV$ sont les relations binaires introduites pour structurer le lien entre les concepts ORGANIZATION, OBJECT and VIEW, comme suit :

— $UseOr$: elle relie le concept USE au concept ORGANIZATION.

— $UseO$: elle relie le concept USE au concept OBJECT.

— $UseV$: elle relie le concept USE au concept VIEW.

Les tâches sont les principaux blocs dans la définition des plans ou des processus qui permettent de spécifier comment utiliser les données privées dans un système. Pour cela, nous introduisons les concepts TASK et TASK-INSTANCE aux niveaux abstrait et concert (voir la Figure 5.3) dans le modèle $P-OrBAC_{\delta\epsilon}$ comme suit :

— Le concept TASK est un ensemble ordonné d'activités qui sont appliqués sur des vues afin de réaliser un objectif légitime.

— Le concept TASK-INSTANCE est une séquence d'actions concrètes qui sont appliquées sur des objets pour instancier une tâche.

Pour simplifier, nous représentons chaque activité appliquée sur une vue et chaque action appliquée sur un objet respectivement par les concepts $TASK_i$ et $TASK-$

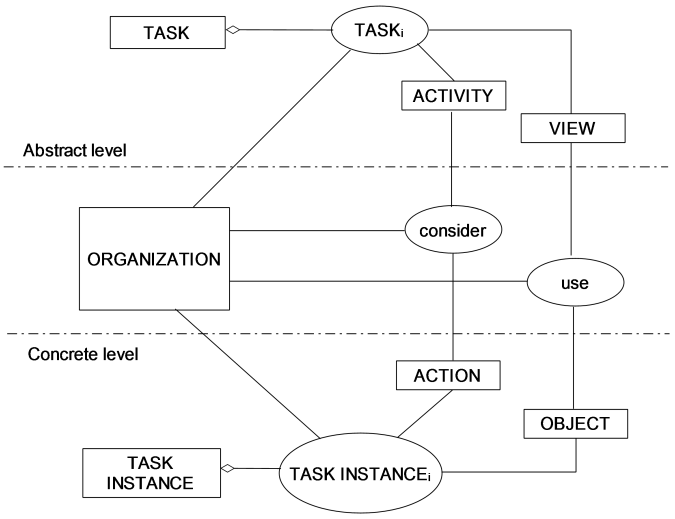

Figure 5.3 – L'intégration des concepts de $TASK$ et $TASK - INSTANCE$ dans le modèle $P - OrBAC_{\delta\epsilon}$

$INSTANCE_i$ où $i \in \mathbb{N}^*$.

— $TASK_i \sqsubseteq \forall taskO_i : ORGANIZATION \sqcap \forall taskA_i : ACTIVITY \sqcap \forall taskV_i : VIEW \sqcap$ $taskO_i\ AT - LEAST\ 1 \sqcap taskA_i\ AT - LEAST\ 1 \sqcap taskV_i\ AT - LEAST\ 1$

Le concept $TASK_i$ est un nouveau concept introduit dans notre modèle pour représenter que l'organisation applique une activité sur une vue.

$taskO_i$,$taskA_i$ and $taskV_i$ sont les relations binaires définis pour structurer le lien entre les concepts ORGANIZATION, ACTIVITY et VIEW, comme suit :

— $taskO_i$: elle relie le concept $TASK_i$ au concept ORGANIZATION.

— $taskA_i$: elle relie le concept $TASK_i$ au concept ACTIVITY.

— $taskV_i$: elle relie le concept $TASK_i$ au concept VIEW.

— $TASK{-}INSTANCE_i \sqsubseteq \forall task{-}instanceO_i : ORGANIZATION \sqcap \forall task{-}instanceA_i :$ $ACTION \sqcap \forall task - instanceO_i : OBJECT \sqcap task - instanceO_i\ AT - LEAST$ $1 \sqcap task - instanceA_i\ AT - LEAST\ 1 \sqcap task - instanceO_i\ AT - LEAST\ 1$

Le concept $TASK - INSTANCE_i$ est un nouveau concept introduit pour représenter que l'organisation exécute une action sur un objet.

$task{-}instanceO_i$, $task{-}instanceA_i$ et $task{-}instanceO_i$ sont les relations binaires définis pour structurer le lien entre les concepts ORGANIZATION, ACTION et OBJECT , comme suit :

— $task - instanceO_i$: elle relie le concept $TASK - INSTANCE_i$ au concept ORGANIZATION.

— $task - instanceA_i$: elle relie le concept $TASK - INSTANCE_i$ au concept

ACTION.

— $task - instanceO_i$: elle relie le concept $TASK - INSTANCE_i$ au concept OBJECT.

Maintenant, nous définissons les concepts temporels TASK et TASK-INSTANCE comme un ensemble ordonnées de $TASK_i$ et $TASK - INSTANCE_i$ respectivement en fixant un intervalle de temps qui limite la durée de chaque $TASK_i$ et $TASK - INSTANCE_i$ comme suit :

— $TASK \sqsubseteq \diamond(\overline{X_i})(\overline{Tc}).\ (\overline{TASK_i@X_i})$

où $\overline{X_i}$ est un ensemble de variables temporelles et $\overline{Tc}$ est un ensemble de contraintes temporelles entre les variables temporelles precedement definis.

— $TASK - INSTANCE \sqsubseteq \diamond(\overline{X_i})(\overline{Tc}).(\overline{TASK - INSTANCE_i@X_i})$

Les systèmes non monotones sont basés principalement sur l'hypothèse du monde fermé, qui stipule que si un fait positif n'a pas été spécifié ou ne peut pas être déduit de la base de connaissances, alors qu'il est faux et sa négation est vrai. Dans ce cas, il suffit de ne représenter que des faits positifs. En conséquence, nous devons adopter la politique fermée dans notre modèle de contrôle d'accès, nous définissons uniquement les PERMISSION et nous nous assurons donc que chaque demande d'accès a une réponse.

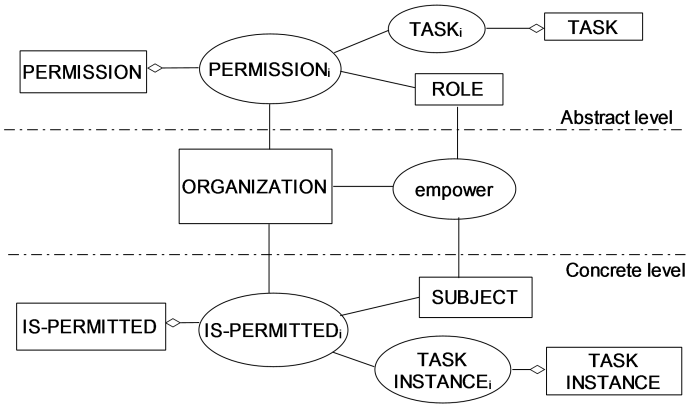

Figure 5.4 – La représentation de $PERMISSION$ et $IS - PERMITTED$ dans le modèle $P - OrBAC_{\delta\epsilon}$

Une permission dans notre nouveau modèle peut être vu comme une succession d'exécutions des taches élémentaires par un rôle (voir Figure 5.4). Donc, le concept de PERMISSION sera donc défini comme un ensemble d'autorisations élémentaires ordonnées ($PERMISSION_i$), en fixant un intervalle de temps qui limite la durée de validité pour chacune des autorisations élémentaires. Les concepts $PERMISSION_i$ et

$PERMISSION$ sont définis comme suit :

— $\delta PERMISSION_i \sqsubseteq \forall permissionO_i : ORGANIZATION \sqcap \forall permissionR_i : ROLE \sqcap \forall permissionT_i : TASK_i \sqcap permissionO_i\ AT-LEAST\ 1 \sqcap permissionR_i\ AT-LEAST\ 1 \sqcap permissionT_i\ AT-LEAST\ 1$

$\delta PERMISSION_i$ signifie que, l'organisation accorde par défaut l'autorisation au rôle d'effectuer une tâche élémentaire.

$permissionO_i$, $permissionR_i$ et $permissionT_i$ sont des relations binaires utilisées pour structurer le lien entre les concepts ORGANISATION, ROLE et TASK, comme suit :

— $permissionO_i$: elle relie le concept $PERMISSION_i$ au concept ORGANIZATION.

— $permissionR_i$: elle relie le concept $PERMISSION_i$ au concept ROLE.

— $permissionT_i$: elle relie le concept $PERMISSION_i$ au concept TASK$_i$.

Maintenant, nous définissons le concept PERMISSION comme une séquence de $\delta PERMISSION_i$ comme suit :

— $PERMISSION \sqsubseteq \diamond(\overline{X_i})(\overline{Tc}).\ (\overline{\delta PERMISSION_i @ X_i})$

où $\overline{X_i}$ est un ensemble de variables temporals et $\overline{Tc}$ est un ensemble de contraintes temporelles entre les variables temporelles.

Dans notre modèle proposé, la permission concrète ($IS-PERMITTED$) sera également représentée comme une séquence des permissions élémentaires concrètes ($IS-PERMITTED_i$). Les concepts $IS-PERMITTED_i$ et $IS-PERMITTED$ sont défini comme suit :

— $IS-PERMITTED_i \sqsubseteq \forall\ is-permittedS_i : SUBJECT \sqcap \forall\ is-permittedI_i : TASKINSTANCE_i \sqcap is-permittedS_i\ AT-LEAST\ 1 \sqcap is-permittedI_i\ AT-LEAST\ 1$

La permission concrète ($IS-PERMITTED_i$) signifie que le sujet est autorisé à effectuer une instance de la tâche$_i$.

$is-permittedO_i$, $is-permittedS_i$ et $is-permittedI_i$ sont des relations binaires utilisées pour structurer le lien entre les concepts ORGANIZATION, SUBJECT and TASKINSTANCE$_i$, comme suit :

— $is-permittedS_i$: elle relie le concept IS-PERMITTED$_i$ au concept SUBJECT.

— $is-permittedI_i$: elle relie le concept IS-PERMITTED$_i$ au concept TASKINSTANCE$_i$.

Maintenant, nous définissons le concept temporel $IS-PERMITTED$ comme une séquence de $IS-PERMITTED_i$ comme suit :

— $IS-PERMITTED \sqsubseteq \Diamond(\overline{X_i})(\overline{Tc}).(\overline{IS-PERMITTED_i@X_i})$

L'instanciation des différents concepts et relations binaires présentées ci-dessus formes l'ABox de notre base de connaissances $\mathcal{K}$ (voir la Figure 5.5), par exemple, l'hôpital (Hospital1) emploie Bob dans le rôle médecin. Cela peut être représenté comme : $EMPOWER(E1) \sqsubseteq EmpowerO.(Hospital1) \sqcap EmpowerS.(Bob) \sqcap EmpowerR.(Physician)$. Nous présentons dans la section suivante, la définition des règles de sécurité qui seront utilisées dans le modèle $P-OrBAC_{\delta\epsilon}$ pour déduire la séquence d'actions appropriée qui peut être effectuée par le sujet qui demande l'accès aux données privées dans un contexte donné.

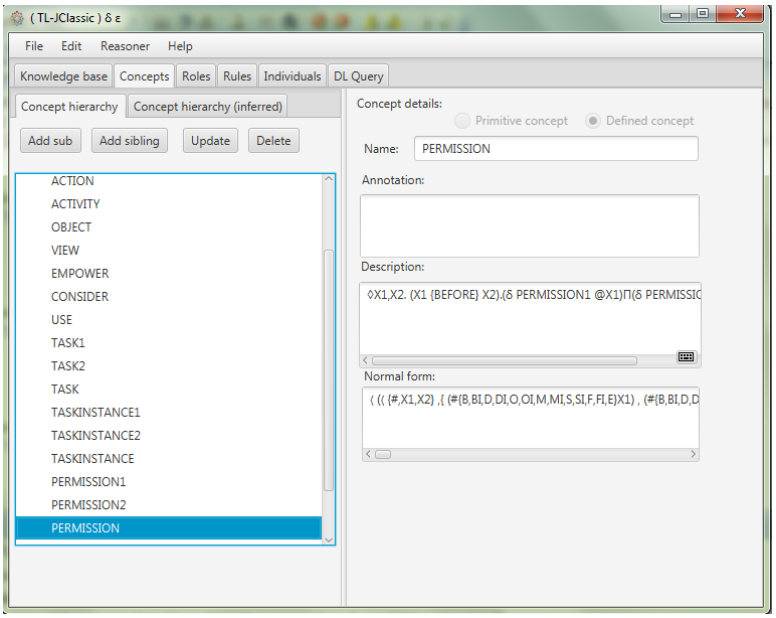

Figure 5.5 – La base de connaissences de $P-OrBAC_{\delta\epsilon}$

5.6.1 Définition des règles de sécurité

Après avoir présenté dans la section précédente la spécification de notre modèle, nous devons être en mesure d'exploiter les informations qu'il représente et d'en tirer des conclusions. Dans notre modèle, les permissions sont définies au niveau abstrait sous forme génériques sujettes à l'exception, donc pour répondre à la demande d'un utilisateur pour exécuter une séquence d'actions dans le système, nous définissons dans cette section les règles de sécurité qui permettent de déduire les permissions concrètes qui peuvent être dérivées des permissions abstraites :

— $\Diamond\ (\overline{X_i})\ (\overline{Tc}).\ \sqcap\ \delta\ (is-permittedS_i)\ FILLS\ \{S\} \sqcap \forall\ is-permittedI_i$:$(task-instanceA_i\ FILLS\ \{A_i\} \sqcap task-instanceO_i\ FILLS\ \{O_i\}))\ @X_i \sqsubseteq \Diamond\ (\overline{X_i})\ (\overline{Tc}).\ \sqcap$ $(empowerS\ FILLS\ \{S\} \sqcap empowerO\ FILLS\ \{Org\} \sqcap empowerR\ FILLS\ \{R\} \sqcap EMPOWER \sqcap considerA\ FILLS\ \{A_i\} \sqcap considerO\ FILLS\ \{Org\} \sqcap considerAct\ FILLS\ \{Act_i\} \sqcap CONSIDER \sqcap useO\ FILLS\ \{O_i\} \sqcap useOr\ FILLS\ \{Org\} \sqcap useV\ FILLS\ \{V_i\} \sqcap USE \sqcap \delta\ (permissionO_i\ FILLS\ \{Org\} \sqcap permissionR_i\ FILLS\ \{R\} \sqcap \forall\ permissionT_i$:$(taskO_i\ FILLS\ \{Org\} \sqcap taskA_i\ FILLS\ \{Act_i\} \sqcap taskV_i\ FILLS\ \{V_i\} \sqcap PERMISSION_i\ )))\ @X_i$

Si l'organisation Org habilite le sujet S dans le role R ($EMPOWER$), s'il existe une relation entre Org, l'action A_i et l' activité Act_i (Consider), s'il existe une relation entre Org, l'objet O_i et la vue V_i (Use), si org définis la sous tâche T_i par l'activité Act_i qui s'appliquait dans la vue V_i, si Org par défaut accorde le rôle R la permission d'effectuer la tâche T_i ($\delta PERMISSION_i$) durant l'intervalle X_i, dans ce cas, S est permis d'exécuter l'instance de tache $task-instance\ I$ qui est l'action A_i qui s'appliquent sur l'objet O_i à l'intervalle X_i ($\sqcap\ \delta IS-PERMITTED_i@X_i$), et parce qu'une permission concrète peut être déduite d'une permission par défaut ($IS-PERMITTED \sqsubseteq \delta IS-PERMITTED$) (à partir des propriétés des concepts de $TL-JClassic_{\delta\epsilon}$), on peut déduire que le sujet S est autorisé à exécuter l'action A_i défini sur l'objet O_i

— $\Diamond\ (\overline{X_i})\ (\overline{Tc}).\ \sqcap\ (is-permittedS_i\ FILLS\ \{S\} \sqcap \forall\ is-permittedI_i$:$(task-instanceA_i\ FILLS\ \{A_i\} \sqcap task-instanceO_i\ FILLS\ \{O_i\}))^{\epsilon}\ @X_i \sqsubseteq \Diamond\ (\overline{X_i})\ (\overline{Tc}).\ \sqcap\ (empowerS\ FILLS\ \{S\} \sqcap empowerO\ FILLS\ \{Org\} \sqcap empowerR\ FILLS\ \{R\} \sqcap EMPOWER \sqcap considerA\ FILLS\ \{A_i\} \sqcap considerO\ FILLS\ \{Org\} \sqcap considerAct\ FILLS\ \{Act_i\} \sqcap CONSIDER \sqcap useO\ FILLS\ \{O_i\} \sqcap useOr\ FILLS\ \{Org\} \sqcap useV\ FILLS\ \{V_i\} \sqcap USE \sqcap (permissionO_i\ FILLS\ \{Org\} \sqcap permissionR_i\ FILLS\ \{R\} \sqcap \forall\ permissionT_i$:$(taskO_i\ FILLS\ \{Org\} \sqcap taskA_i\ FILLS\ \{Act_i\} \sqcap taskV_i\ FILLS\ \{V_i\} \sqcap PERMISSION_i\ ))^{\epsilon})\ @X_i$

Si nous avons une exception pour le rôle R d'exécuter la tâche T_i ($PERMISSION_i^{\epsilon}@X_i$), on déduit que nous devons ignorer la $task-instance_i\ I_i$ et une permission concrète ne peut être déduite d'une tâche élémentaire ignorée ($IS-PERMITTED \notin IS-PERMITTED^{\epsilon}$), on peut en déduire que le sujet S est interdit d'effectuer cette $task-instance_i\ I_i$ à l'intervalle X_i.

En utilisant les services d'inférences de $TL-JClassic_{\delta\epsilon}$, nous calculons la séquence des permissions concrètes appropriées à être effectuée dans le système selon le contexte

dans lequel la demande de l'accès a été effectuée. Dans la section suivante, nous illustrons formellement comment la décision d'accès basée sur notre modèle peut être déduite dans des contextes différents.

5.7 Étude de cas

Durant ces dernière années, de nombreux pays se sont engagés dans des programmes ambitieux de dossiers médicaux électroniques (EMR - Electronic Medical Record) sous forme PaaS (à l'échelle d'un pays) avec pour objectif d'accroître la qualité des soins tout en réduisant les coûts. Un système de gestion de dossiers médicaux électroniques constitue donc une collection de dossiers de patients informatisés, contenant chacun l'historique médical complet d'un individu. L'accès ubiquitaire à ces données vise à rendre les informations de santé des patients disponibles en toute sécurité n'importe où et n'importe quand. La mise en place des politiques de contrôle d'accès pour les données médicales qui sont extrêmement sensibles et confidentielles est souvent régie par des textes législatifs. Le dossier d'un patient n'apparaît à un professionnel de santé que sous l'angle des besoins de sa tâche au sein de l'organisation.

Supposons que, Alice possède un dossier médical au système national de gestion de dossiers médicaux électroniques. Alice est infectée par le VIH/sida. « VIH » est l'abréviation de « virus de l'immunodéficience humaine ». Le terme « immunodéficience » désigne l'état d'un système immunitaire affaibli. On dit des personnes infectées par le VIH qu'elles sont séropositives. Notez que cette information est extrêmement sensible et en même temps c'est une information essentielle dans le processus de soins. Un accès inapproprié à cette donnée peut entraîner une stigmatisation ou une discrimination envers le patient. Pour cela, la réglementation nationale précise que l'information sur le VIH/sida d'un patient ne sera pas divulguée sans le consentement du patient, sauf dans des cas spécifiques tels qu'une urgence. Un jour, Alice a eu un accident de voiture et elle est admise au service des urgences à Hospital1 à 23 :00. Le médecin Bob qui traite Alice dans l'hôpital veut consulter son historique médical, pour voir s'il y a des facteurs à considérer. Cela est instancié par les individus suivants :

$TASK-INSTANCE_1\ (I1) \sqsubseteq task-instanceO_1.(Hospital1) \sqcap task-instanceA_1.\ (read)$
$\sqcap\ task-instanceO_1.(Alice-HIV/AIDS)$

$TASK-INSTANCE_2\ (I2) \sqsubseteq task-instanceO_2.(Hospital1) \sqcap task-instanceA_2.\ (read) \sqcap$
$task-instanceO_2.(Alice-Allergies)$

$IS-PERMITTED_1\ (Isp1) \sqsubseteq is-permittedS_1.(Bob) \sqcap is-permittedI_1.(I1)$

$IS-PERMITTED_2\ (Isp2) \sqsubseteq is-permittedS_2.(Bob) \sqcap is-permittedI_2.(I2)$

$IS-PERMITTED\ (Isp) \sqsubseteq (Isp1)\ @[23:03-23:04] \sqcap (Isp2)\ @\ [23:05-23:06]$

Pour évaluer la demande de Bob, nous vérifions si nous pouvons déduire $(IS-PERMITTED\ (Isp))$.

Dans la politique de sécurité de l'hôpital, nous avons cette règle d'autorisation, en situation d'urgence, le médecin est autorisé à consulter l'historique médical d'un patient. La tâche (consulter l'historique médical) est définie comme séquence des sous tâches facultatives suivantes, consulter ses maladies chroniques et consulter ses allergies.

$TASK_1\ (T1) \sqsubseteq taskO_1.(Hospital1) \sqcap taskA_1.(consult) \sqcap taskV_1.(chronicillness)$

$TASK_2\ (T2) \sqsubseteq taskO_2.(Hospital1) \sqcap taskA_2.(consult) \sqcap taskV_2.(allergies)$

$PERMISSION_1\ (P11) \sqsubseteq permissionO_1.(Hospital1) \sqcap$

$permissionR_1.(Doctor) \sqcap permissionT_1.(T1)$

$PERMISSION_2\ (P12) \sqsubseteq permissionO_2.(Hospital1) \sqcap$

$permissionR_2.(Doctor) \sqcap permissionT_2.(T2)$

$PERMISSION\ (P1) \sqsubseteq \diamond\ (X_1, X_2)\ (X_1\{p\}X_2)\ \delta\ (P11)\ @X_1 \sqcap \delta\ (P12)@\ X_2$

L' $ABox$ contient aussi :

$ORGANIZATION(Hospital1)$

$SUBJECT(Bob)$

$ROLE(Doctor)$

$ACTION(read)$

$ACTIVITY(Consultation)$

$VIEW(Patientallergies)$

$VIEW(Patientchronicillness)$

$OBJECT(Alice-HIV/AIDS)$

$OBJECT(Alice-Allergies)$

$EMPOWER(E1) \sqsubseteq EmpowerO.(Hospital1) \sqcap EmpowerS.(Bob) \sqcap EmpowerR.(Doctor)$

$CONSIDER(C) \sqsubseteq considerO.(Hospital1) \sqcap considerA.(Read) \sqcap considerAct.(Consult)$

$USE(U1) \sqsubseteq useOr.(Hospital1) \sqcap useO.(Alice-HIV/AIDS) \sqcap useV.(chronicillness)$

$USE(U2) \sqsubseteq useOr.(Hospital1) \sqcap useO.(Alice-Allergies) \sqcap useV.(allergies)$

En utilisant les règles de sécurité, nous en déduisons :

$\delta\ IS{-}PERMITTED_1\ (Isp1)\ @\ [23:03{-}23:04] \sqsubseteq EMPOWER(E1) \sqcap CONSIDER(C)$

$\sqcap USE(U1) \sqcap \delta\ PERMISSION_1\ @\ X_1$

$\delta\ IS{-}PERMITTED_2\ (Isp2)\ @\ [23:05{-}23:06] \sqsubseteq EMPOWER(E1) \sqcap CONSIDER(C) \sqcap USE(U2) \sqcap \delta\ PERMISSION_2\ @X_2$

alors, on peut en déduire $IS-PERMITTED_1\ (Isp1)\ @\ [23:03-23:04] \sqcap IS-PERMITTED_2\ (Isp2)\ @\ [23:05-23:06]$ (voir la Figure 5.6)

Donc, nous dérivons $IS-PERMITTED(Isp)$. Cela signifie que, Bob est autorisé à effectuer la séquence d'actions suivante, lire (Alice-VIH/SIDA) puis lire (Alice-allergies).

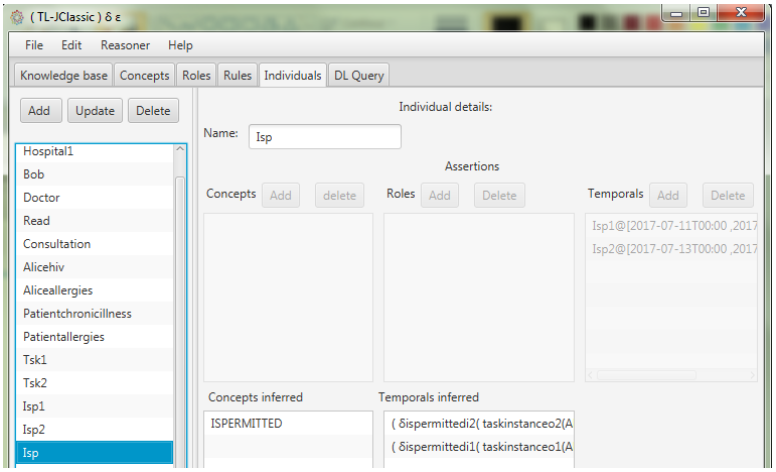

Figure 5.6 – Une décision d'accès dans le contexte "urgent"

Après un examen médical, le résultat montre qu'Alice a subi une fracture au bras et son état n'est pas grave, elle est transférée à un spécialiste pour placer un plâtre. Anna est le médecin qui placera un plâtre à Alice mais avant de faire ça, elle veut aussi consulter l'historique médical d'Alice, une demande d'accès est donc envoyé au système.

La permission de consulter l'historique médical du patient devrait être temporairement adapté à ce nouveau contexte, donc la permission sera définie comme suit :

$PERMISSION\ (P2) \sqsubseteq \Diamond\ (X_1, X_2)\ (X_1\{p\}X_2)\ \delta\ (P21)@\ X_1 \sqcap \delta\ (P22)@\ X_2$

où $\delta\ (P21)$ =$\delta\ (P11)^{\epsilon}$ et $\delta\ (P22)$= $\delta\ (P12)$

Donc $PERMISSION\ (P2) \sqsubseteq \Diamond\ (X_1, X_2)\ (X_1\{p\}X_2)\ \delta\ (P11)^{\epsilon}\ @\ X_1 \sqcap \delta\ (P12)\ @\ X_2$

Les nouveaux instanciations ajoutés à la $ABox$ sont :

$EMPOWER(E2) \sqsubseteq EmpowerO.(Hospital1) \sqcap EmpowerS.(Anna) \sqcap EmpowerR.\ (Doctor)$

En utilisant les règles de sécurité, on obtient :

$\delta\ IS-PERMITTED_1^{\epsilon}\ (Isp1)\ @\ [23:33-23:34] \sqsubseteq EMPOWER(E2) \sqcap CONSIDER\ (C) \sqcap USE(U1) \sqcap PERMISSION_1{}^{\epsilon}\ @\ X_1$

$\delta\ IS-PERMITTED_2\ (Isp2)\ @\ [23:35-23:36] \sqsubseteq EMPOWER(E2) \sqcap CONSIDER\ (C) \sqcap USE(U2) \sqcap \delta\ PERMISSION_2\ @\ X_2$

Alors, nous en déduisons que $IS-PERMITTED_2$ $(Isp2)$ @ $[23:35-23:36]$ (voir la Figure 5.7). Dans ce cas, le système refusera temporairement la demande de Anna pour

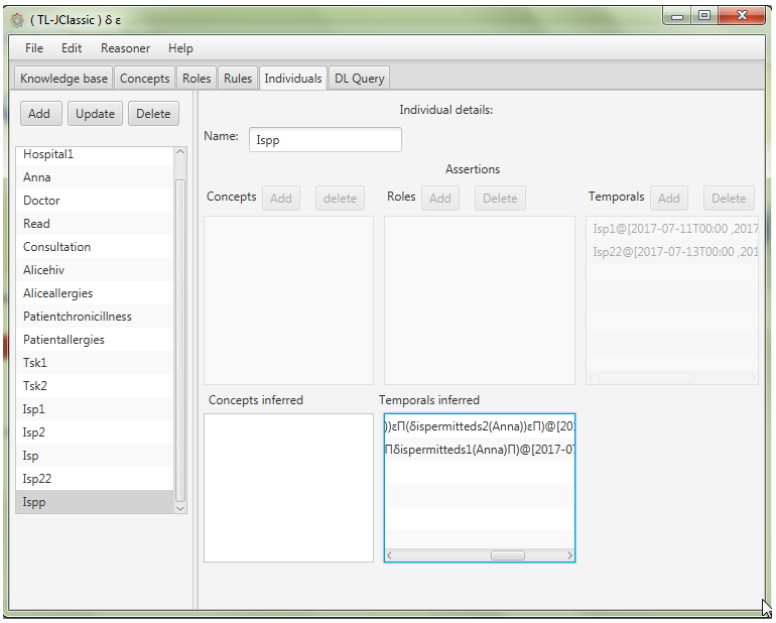

Figure 5.7 – Une décision d'accès dans le contexte "normal"

lire les maladies chroniques d'Alice. Elle est autorisée à lire seulement Alice-allergies pendant [23 : 35-23 : 36].

Cet exemple illustre que, la prise en considération de contexte actuel quand on prend une décision d'effectuer ou non des tâches au sein de système, nous permet de garantir que l'accès aux données des patients se fera de façon contrôlée. Par conséquent, les politiques de sécurités qui limitent l'accès aux données privées à base de l'objectif d'usage qui est représenté sous forme d'un plan, elles doivent être définies de manière flexible afin de capturer la séquence d'actions appropriée lorsque le contexte change.

5.8 Tests et évaluation

Cette section décrit les tests qui ont été effectués afin d'évaluer notre modèle proposé ($P-OrBAC_{\delta\epsilon}$). La procédure et les mesures utilisées pour tester les performances sont décrites dans la section suivante.

Pour évaluer notre modèle proposé, nous avons choisi de le comparer avec le modèle Task-Role Based Access control(TRBAC) présenté dans [106], TRBAC est un modèle de contrôle d'accès qui inclut des contraintes sur les tâches et les rôles correspondants aux ceux qui vont tenter d'accéder à un système.

Nous avons implémenté des prototypes pour les deux modèles de contrôle d'accès et

mené quelques expérimentations afin de comparer leurs performances au niveau du système EMR qui comprend environ 800 dossiers médicaux des patients. Nous avons développé un module qui permet aux professionnels de la santé de rechercher les dossiers des patients à des fins de recherche sous différents contextes. Pour cela, nous avons défini une procédure qui permet de rechercher et de recueillir toutes les sous tâches qui sont autorisées à cet utilisateur selon un contexte précis. Dans cette étude, nous nous concentrons sur les sous tâches qui permettent l'accès aux données sensibles dans le système. Étant données certaines caractéristiques contextuelles, quelles sont pour un utilisateur jouant un certain rôle, les actions autorisé à effectuer dans le système ?

Les performances du système de contrôle d'accès sont mesurée en termes des mesures de rappel, précision et F-mesure [107]. Dans notre cas, les mesures rappel et précision représenteront la capacité d'un mécanisme de contrôle d'accès pour sélectionner les sous tâches autorisées qui peuvent être exécutées selon les deux modèles de contrôle d'accès lorsqu'un utilisateur envoi une demande d'accès dans un contexte spécifique, leurs valeurs sont calculées en utilisant les formules suivantes :

$$Rappel = \frac{\text{Nombre total de tâches pertinentes autorisées dans le contexte } C}{\text{Nombre total de tâches pertinentes dans le contexte } C}$$

$$Prcision = \frac{\text{Nombre total de tâches pertinentes autorisées dans le contexte } C}{\text{Nombre total de tâches autorisées}}$$

$$F - measure = 2 * \frac{\text{Rappel * Précision}}{\text{Rappel + Précision}}$$

Les résultats sont présentés et décrits dans la section suivante.

5.8.1 Résultats

	$P-OrBAC_{\delta\epsilon}$			$TRBAC$		
requête	Rappel	Précision	F-measure	Rappel	Précision	F-measure
1	0,07	1	0,13	0,04	1	0,08
2	0,13	1	0,24	0,08	1	0,15
3	0,20	1	0,33	0,12	1	0,21
4	0,27	1	0,42	0,16	1	0,28
5	0,33	1	0,50	0,20	1	0,33

Tableau 5.2 – Les résultats des tests

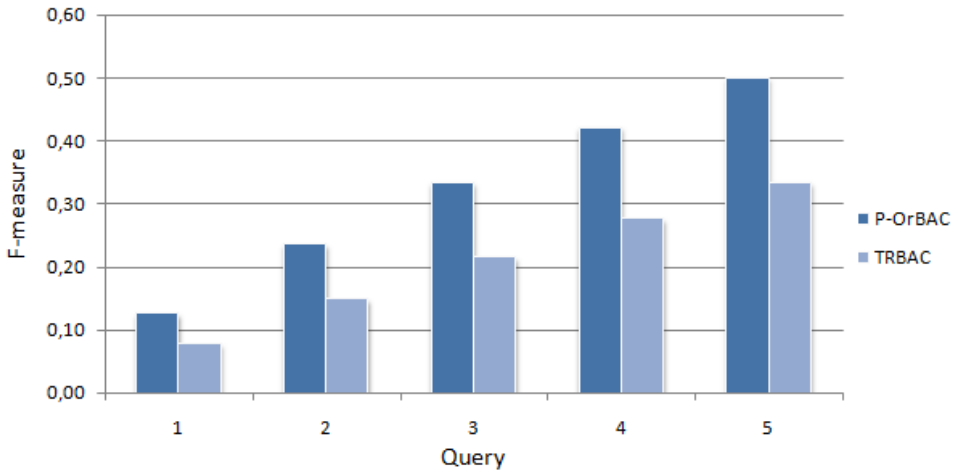

Figure 5.8 – Histogramme représente F-mesure vs requête dans les modèles $P - OrBAC_{\delta\epsilon}$ et $TRBAC$

5.9 Discussion

Les résultats présentés dans le tableau 5.2 et dans la Figure 5.8 illustrent les valeurs de mesure calculées à partir des décision d'accès déduites aux différentes requêtes effectuées dans le système à l'aide des mécanismes $P - OrBAC_{\delta\epsilon}$ et TRBAC. Lorsqu'on compare les résultats des deux mécanismes, nous constatons que notre mécanisme donne de meilleures performances. Bien que la performance de précision est similaire pour les deux mécanismes, cela est dû au fait que le système EMR exige une grande précision ce qui signifie que le mécanisme de contrôle d'accès doit accorder l'accès seulement à l'utilisateur autorisé mais pour les mesures de rappel, notre modèle proposé montre des valeurs plus élevées sur toutes les requêtes par rapport à TRBAC et cela permet à notre modèle d'avoir des valeurs F-measure plus élevé. Cela signifie que notre modèle ne permet pas au système de sélectionner toutes les tâches à évaluer, parce que les tâches sont déjà catégorisées en contexte et cela ne peut pas se faire dans TRBAC parce que le contexte n'est pas présenté explicitement dans leurs règles. Par conséquent, notre mécanisme permet de sélectionner les tâches à effectuer plus correctement dans le système.

CONCLUSION GÉNÉRALE

Le cloud computing a réussi à réaliser l'idée de l'informatique dématérialisé, aujourd'hui les entreprises et les particuliers n'auront plus à se préoccuper des problématiques liées à la conception, l'installation, la maintenance et l'administration de leurs système informatique, les fournisseurs du cloud computing se chargent et leurs fournissent des services de qualités mais le cloud est comme chaque nouvelle vague de technologie apporte aussi avec lui de nouveaux risques de sécurité. L'hébergement des données sensibles dans les services cloud peut augmenter considérablement la possibilité d'une utilisation abusive de ces données. Notre objectif dans cette thèse est de proposer de nouvelle technique permettant de garantir la confidentialité de données sauvegardées dans le cloud.

Une politique d'autorisation a pour objectif de contrôler l'accès aux données conformément à un ensemble de règles d'autorisation. Après l'étude de l'environnement cloud computing et l'analyse des principaux modèles des contrôle d'accès proposé dans la littérature, nous avons constaté qu'un système de contrôle d'accès dans le cloud a besoin que sa politique change et s'adapte temporairement aux changements de contexte afin de protéger ses données et de mieux garantir les services qui conviendraient à ses utilisateurs. Pour cela, il est nécessaire d'incorporer des contraintes temporelles dans les règles d'autorisations et il est intéressant aussi de combiner des permissions et des interdictions dans une même politique de sécurité, néanmoins cela conduit facilement à des situations de conflits qui nécessitent d'être résolut afin d'avoir une seul réponse a une requête d'accès. Pour répondre à ces exigences, nous avons opté pour l'utilisation d'un formalisme logique non monotone qui permet de fournir un cadre permettant de spécifier formellement des regèles d'autorisation contextuelle sous forme prototypique admettant des exceptions et qui peut servir à résoudre les conflits éventuels dans la politique de sécurité.

Le formalisme pour lequel nous avons opté pour la représentation des règles d'autorisations est celui de la logique de description non monotone $JClassic_{\delta\epsilon}$. Cette logique a pour avantage de disposer d'un langage de représentation expressif et d'avoir des algorithmes de raisonnement décidables. De plus, ce formalisme est suffisamment souple pour introduire de nouveaux constructeurs afin de répondre à des besoins particuliers,

il suffit juste de tenir compte de la complexité de la subsomption qui en résulte. Nous avons donc proposé d'étendre la logique $JClassic_{\delta\epsilon}$ avec une dimension temporelle afin de permettre d'exprimer des contraintes temporelles qualitatives dans la définition des concepts temporels et permet également de représenter des connaissances temporelle sujette à l'exception. Ce nouveau formalisme nous servira par ensuite à classifier les connaissances temporelles et leurs héritages sous forme hiérarchique tout en gardant une complexité de calcul raisonnable.

Nous avons choisi initialement de travailler sur les réseaux sociaux qui sont des exemples de service cloud, en utilisant notre logique. Nous avons proposé un modèle de contrôle d'accès dynamique basé sur le contexte temporel, qui permet aux utilisateurs d'introduire des contraintes temporelle afin de limiter l'accès à leurs données partagées dans les réseaux sociaux, ainsi que de leur permettre de faire des exceptions pour certains règles dans un contexte particulier et par conséquent, la décision d'accès sera déduit de façon intelligente en fonction du contexte actuel du système.

Nous sommes intéressés par la suite au modèle de contrôle d'accès basé sur l'objectif d'usage qui permet de garantir que les données privées ne peuvent être utilisées que pour l'objectif auquel elles sont destinées. Nous avons donc proposé une nouvelle approche flexible pour définir et mettre en œuvre ce modèle aux données hébergés en cloud. Dans notre nouveau modèle, une règle d'autorisation basée sur l'objectif d'utilisation est définie comme un ensemble ordonné de permission élémentaire pouvant inclure des actions optionnelles et qui peuvent être ignorés ou annulés dans certains cas afin de s'adapter de manière flexible aux changements de contexte dans l'environnement cloud. La requête d'accès au système dans ce modèle sera différente, elle représente une demande d'exécuter une séquence d'action. De ce fait, le processus de prise de décision sera réduit à un problème de classification afin de déduire l'objectif de l'accès de l'utilisateur et pour déterminer la séquence d'action appropriée à exécuter en utilisant l'héritage défavorable défini dans notre logique.

Notre approche logique de mise en œuvre de politique de sécurité est plus flexible et plus approprié aux exigences de l'environnement cloud que les autres modèles existant. Néanmoins notre travail peut être amélioré de différentes manières :

Tout d'abord, il serait intéressant d'expérimenter notre modèle formel dans un cadre réel. La poursuite du développement et l'évaluation supplémentaire doit se produire

pour évaluer la performance de notre modèle proposé dans les services de cloud computing existants.

- Il est intéressant d'étendre notre formalisme logique proposé pour d'autres types de contraintes temporelles telles que les contraintes temporelles métriques qui peuvent être utilisés comme des contraintes pour les actions définis dans un concept temporel.

-Il est important de considérer des règles de politiques plus sophistiquées, en particulier les règles de délégation et de transfert de droits qui est couramment utilisée dans les organisations. Une délégation est un transfert temporaire des droits d'accès d'un sujet chargé de certaines tâches à un autre pour les accomplir. Cette forme d'autorisation est basée aussi sur la notion de temps, il est nécessaire de proposer des mécanismes plus concrets qui permettent de satisfaire ce genre d'exigences.

RÉFÉRENCES

1. Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.
2. Barrie Sosinsky. *Cloud computing bible*, volume 762. John Wiley & Sons, 2010.
3. Pierangela Samarati and Sabrina Capitani de Vimercati. Access control : Policies, models, and mechanisms. In *International School on Foundations of Security Analysis and Design*, pages 137–196. Springer, 2000.
4. Narhimene Boustia and Aicha Mokhtari. A dynamic access control model. *Applied Intelligence*, 36(1) :190–207, 2012.
5. Alessandro Artale and Enrico Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research*, 9 :463–506, 1998.
6. James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11) :832–843, 1983.
7. Khalida Guesmia and Narhimene Boustia. Orbac from access control model to access usage model. *Applied Intelligence*, pages 1–21, 2017.
8. Racha Ajami, Noha Ramadan, Nader Mohamed, and Jameela Al-Jaroodi. Security challenges and approaches in online social networks : A survey. *IJCSNS*, 11(8) :1, 2011.
9. Michelle Madejski, Maritza Johnson, and Steven M Bellovin. The failure of online social network privacy settings. *Department of Computer Science, Columbia University, Tech. Rep. CUCS-010-11*, 2011.
10. Oshrat Ayalon and Eran Toch. Retrospective privacy : Managing longitudinal privacy in online social networks. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, page 4. ACM, 2013.
11. Ji-Won Byun, Elisa Bertino, and Ninghui Li. Purpose based access control of complex data for privacy protection. In *Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 102–110. ACM, 2005.

12. Qun Ni, Elisa Bertino, Jorge Lobo, Carolyn Brodie, Clare-Marie Karat, John Karat, and Alberto Trombeta. Privacy-aware role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 13(3) :24, 2010.

13. Faranak Farzad, SK Eric, and Patrick CK Hung. Role-based access control requirements model with purpose extension. In *WER*, pages 207–216, 2007.

14. Michael Carl Tschantz, Amitava Datta, and Jeannette M Wing. Formalizing and enforcing purpose restrictions in privacy policies. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 176–190. IEEE, 2012.

15. Guillaume Plouin. *Cloud Computing-2e éd. : Une rupture décisive pour l'informatique d'entreprise*. Dunod, 2011.

16. Franck Leon. *La construction des Business Models des fournisseurs de services d'infrastructure Cloud Computing (IaaS)*. PhD thesis, Université Nice Sophia Antipolis, 2015.

17. Imad M Abbadi and Andrew Martin. Trust in the cloud. *information security technical report*, 16(3) :108–114, 2011.

18. Fawaz Paraiso. *soCloud : une plateforme multi-nuages distribuée pour la conception, le déploiement et l'exécution d'applications distribuées à large échelle*. PhD thesis, Université des Sciences et Technologie de Lille-Lille I, 2014.

19. Jeremy Geelan et al. Twenty one experts define cloud computing. *Cloud Computing Journal*, 4 :1–5, 2009.

20. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms : Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6) :599–616, 2009.

21. Lizhe Wang, Gregor Von Laszewski, Andrew Younge, Xi He, Marcel Kunze, Jie Tao, and Cheng Fu. Cloud computing : a perspective study. *New Generation Computing*, 28(2) :137–146, 2010.

22. Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing : Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 5–13. Ieee, 2008.

23. Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds : towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1) :50–55, 2008.

24. Wayne Jansen and Timothy Grance. Sp 800-144. guidelines on security and privacy in public cloud computing. 2011.

25. James E Smith and Ravi Nair. The architecture of virtual machines. *Computer*, 38(5) :32–38, 2005.

26. Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing : state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1) :7–18, 2010.

27. Houssem Medhioub. *Architectures et mécanismes de fédération dans les environnements Cloud Computing et Cloud Networking*. PhD thesis, Evry, Institut national des télécommunications, 2015.

28. Bhaskar Prasad Rimal, Admela Jukan, Dimitrios Katsaros, and Yves Goeleven. Architectural requirements for cloud computing systems : an enterprise cloud approach. *Journal of Grid Computing*, 9(1) :3–26, 2011.

29. Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing : issues and challenges. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 27–33. Ieee, 2010.

30. Cameron Coles and John Yeoh. Cloud adoption practices & priorities survey report. *Cloud Security Alliance*, 2015.

31. Christian Delettre. *Plateforme ouverte, évolutive, sécurisée et orientée utilisateur pour l'e-commerce*. PhD thesis, Université Nice Sophia Antipolis, 2014.

32. Romuald Thion. *STRUCTURATION RELATIONNELLE DES POLITIQUES DE CONTRÔLE D'ACCÈS REPRÉSENTATION, RAISONNEMENT ET VÉRIFICATION LOGIQUES*. PhD thesis, Université Paul Sabatier Toulouse, 2008.

33. Lili Qiu, Yin Zhang, Feng Wang, Mi Kyung, and Han Ratul Mahajan. Trusted computer system evaluation criteria. In *National Computer Security Center*. Citeseer, 1985.

34. Butler W Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1) :18–24, 1974.

35. D Elliott Bell and Leonard J La Padula. Secure computer system : Unified exposition and multics interpretation. Technical report, MITRE CORP BEDFORD MA, 1976.

36. Kenneth J Biba. Integrity considerations for secure computer systems. Technical report, MITRE CORP BEDFORD MA, 1977.

37. Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2) :38–47, 1996.

38. David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3) :224–274, 2001.

39. Anas Abou El Kalam. *Modèles et politiques de sécurité pour les domaines de la santé et des affaires sociales*. PhD thesis, Institut National Polytechnique de Toulouse-INPT, 2003.

40. Alban Gabillon. An authorization model for xml databases. In *Proceedings of the 2004 workshop on Secure web service*, pages 16–28. ACM, 2004.

41. Fredéric Cuppens and Nora Cuppens-Boulahia. Les modèles de sécurité. *Sécurité des systèmes d'information,(Traité IC2, série Réseaux et télécoms)*, pages 13–48, 2006.

42. Roshan K Thomas and Ravi S Sandhu. Task-based authorization controls (tbac) : A family of models for active and enterprise-oriented authorization management. In *Database Security XI*, pages 166–181. Springer, 1998.

43. James BD Joshi, Elisa Bertino, and Arif Ghafoor. An analysis of expressiveness and design issues for the generalized temporal role-based access control model. *IEEE Transactions on Dependable and Secure Computing*, 2(2) :157–175, 2005.

44. K Roshan and R Thomas. Tmac : A primitive for applying rbac in collaborative environment. In *2nd ACM Workshop on RBAC, Fairfax, Virginia, USA*, pages 6–7, 1997.

45. Anas Abou El Kalam, RE Baida, Philippe Balbiani, Salem Benferhat, Frédéric Cuppens, Yves Deswarte, Alexandre Miege, Claire Saurel, and Gilles Trouessin. Organization based access control. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 120–131. IEEE, 2003.

46. Frédéric Cuppens and Nora Cuppens-Boulahia. Modeling contextual security policies. *International Journal of Information Security*, 7(4) :285–305, 2008.

47. Salem Benferhat, Rania El Baida, and Frédéric Cuppens. A stratification-based approach for handling conflicts in access control. In *Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 189–195. ACM, 2003.

48. Eric Yuan and Jin Tong. Attributed based access control (abac) for web services. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE, 2005.

49. Jaehong Park and Ravi Sandhu. Towards usage control models : beyond traditional access control. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 57–64. ACM, 2002.

50. Jaehong Park and Ravi Sandhu. The ucon abc usage control model. *ACM Transactions on Information and System Security (TISSEC)*, 7(1) :128–174, 2004.

51. Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The nist model for role-based access control : towards a unified standard. In *ACM workshop on Role-based access control*, volume 2000, pages 1–11, 2000.

52. Zuraini Zainol and Keiichi Nakata. Generic context ontology modelling : A review and framework. In *Computer Technology and Development (ICCTD), 2010 2nd International Conference on*, pages 126–130. IEEE, 2010.

53. Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1) :4–7, 2001.

54. Gail-Joon Ahn and Ravi Sandhu. Role-based authorization constraints specification. *ACM Transactions on Information and System Security (TISSEC)*, 3(4) :207–226, 2000.

55. Indrakshi Ray, Na Li, Robert France, and Dae-Kyoo Kim. Using uml to visualize role-based access control constraints. In *Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 115–124. ACM, 2004.

56. Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari. Trbac : A temporal role-based access control model. *ACM Transactions on Information and System Security (TISSEC)*, 4(3) :191–233, 2001.

57. James BD Joshi, Elisa Bertino, and Arif Ghafoor. Temporal hierarchies and inheritance semantics for gtrbac. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 74–83. ACM, 2002.

58. Till Mossakowski, Michael Drouineaud, and Karsten Sohr. A temporal-logic extension of role-based access control covering dynamic separation of duties. In *Temporal Representation and Reasoning, 2003 and Fourth International Conference on Temporal Logic. Proceedings. 10th International Symposium on*, pages 83–90. IEEE, 2003.

59. Maria Luisa Damiani, Elisa Bertino, Barbara Catania, and Paolo Perlasca. Geo-rbac : a spatially aware rbac. *ACM Transactions on Information and System Security (TISSEC)*, 10(1) :2, 2007.

60. Fujun Feng, Chuang Lin, Dongsheng Peng, and Junshan Li. A trust and context based access control model for distributed systems. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 629–634. IEEE, 2008.

61. Ali Ahmed and Ning Zhang. A context-risk-aware access control model for ubiquitous environments. In *Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference on*, pages 775–782. IEEE, 2008.

62. Mary J Culnan. Protecting privacy online : Is self-regulation working ? *Journal of Public Policy & Marketing*, 19(1) :20–26, 2000.

63. Sarah Spiekermann and Lorrie Faith Cranor. Engineering privacy. *IEEE Transactions on software engineering*, 35(1) :67–82, 2009.

64. Fabio Massacci and Nicola Zannone. Privacy is linking permission to purpose. In *International Workshop on Security Protocols*, pages 179–191. Springer, 2004.

65. Naikuo Yang, Howard Barringer, and Ning Zhang. A purpose-based access control model. In *Information Assurance and Security, 2007. IAS 2007. Third International Symposium on*, pages 143–148. IEEE, 2007.

66. Amirreza Masoumzadeh and James BD Joshi. Purbac : Purpose-aware role-based access control. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1104–1121. Springer, 2008.

67. Mohammad Jafari, Reihaneh Safavi-Naini, and Nicholas Paul Sheppard. Enforcing

purpose of use via workflows. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pages 113–116. ACM, 2009.

68. Michael Bratman. Intention, plans, and practical reason. 1987.
69. Narhimene Boustia. *Un formalisme de sécurisation des bases de données multi-niveaux*. PhD thesis, USTHB, 2011.
70. Khair Eddin Sabri and Nadim Obeid. A temporal defeasible logic for handling access control policies. *Applied Intelligence*, 44(1) :30–42, 2016.
71. Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In *Handbook on ontologies*, pages 21–43. Springer, 2009.
72. Franz Baader. *The description logic handbook : Theory, implementation and applications*. Cambridge university press, 2003.
73. Ronald J Brachman and Hector J Levesque. The tractability of subsumption in frame-based description languages. In *AAAI*, volume 84, pages 34–37, 1984.
74. Hector J Levesque and Ronald J Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational intelligence*, 3(1) :78–93, 1987.
75. Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial intelligence*, 48(1) :1–26, 1991.
76. Franz Baader and Werner Nutt. Basic description logics. In *Description logic handbook*, pages 43–95, 2003.
77. Asuncion Gomez-Perez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science & Business Media, 2006.
78. Chan Le Duc. *Transformation d'ontologies basées sur la logique de description : Application dans le commerce électronique*. PhD thesis, Université Nice Sophia Antipolis, 2004.
79. Amedeo Napoli. *Une introduction aux logiques de descriptions*. PhD thesis, INRIA, 1997.
80. Nicolas Chleq. *Contribution à l'étude du raisonnement temporel. Résolution avec contraintes et application à l'abduction en raisonnement temporel*. PhD thesis, Ecole Nationale des Ponts et Chaussées, 1995.

81. Malek Mouhoub. *Contribution à l'étude des techniques de propagation de contraintes symboliques et numériques pour le raisonnement temporel*. PhD thesis, 1996.

82. Brian A Haugh. Non-standard semantics for the method of temporal arguments. In *IJCAI*, pages 449–455, 1987.

83. Joseph Y Halpern and Yoav Shoham. A propositional modal logic of time intervals. *Journal of the ACM (JACM)*, 38(4) :935–962, 1991.

84. Andrei Krokhin, Peter Jeavons, and Peter Jonsson. Reasoning about temporal relations : The tractable subalgebras of allen's interval algebra. *Journal of the ACM (JACM)*, 50(5) :591–640, 2003.

85. Marc B Vilain and Henry A Kautz. Constraint propagation algorithms for temporal reasoning. In *Aaai*, volume 86, pages 377–382, 1986.

86. Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations : a maximal tractable subclass of allen's interval algebra. *Journal of the ACM (JACM)*, 42(1) :43–66, 1995.

87. Alessandro Artale and Enrico Franconi. A survey of temporal extensions of description logics. *Annals of Mathematics and Artificial Intelligence*, 30(1-4) :171–210, 2000.

88. Alessandro Artale and Enrico Franconi. Temporal description logics. *Handbook of Time and Temporal Reasoning in Artificial Intelligence*, 1, 2005.

89. Albrecht Schmiedel et al. *A temporal terminological logic*. Techn. Univ., Projektgruppe KIT, 1990.

90. Robert Weida and Diane Litman. Subsumption and recognition of heterogeneous constraint networks. In *Artificial Intelligence for Applications, 1994., Proceedings of the Tenth Conference on*, pages 381–388. IEEE, 1994.

91. Ouarda Bettaz, Narhimene Boustia, and Aryan Mokhtari. Extending nonmonotonic description logic with temporal aspects. In *Innovations in Intelligent Systems and Applications (INISTA), 2013 IEEE International Symposium on*, pages 1–5. IEEE, 2013.

92. Robert A Weida and Diane J Litman. Terminological reasoning with constraint networks and an application to plan recognition. *KR*, 92 :282–293, 1992.

93. Alan K Mackworth and Eugene C Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial intelligence*, 25(1) :65–74, 1985.

94. Alan K Mackworth. Consistency in networks of relations. *Artificial intelligence*, 8(1) :99–118, 1977.

95. Romuald Debruyne and Christian Bessiere. From restricted path consistency to max-restricted path consistency. In *Principles and Practice of Constraint Programming-CP97*, pages 312–326. Springer, 1997.

96. Ashutosh Saxena, Ina Jain, and M Choudary Gorantla. An integrated framework for enhancing privacy in online social networks. In *Proceedings of the 6th ACM India Computing Convention*, page 5. ACM, 2013.

97. Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thuraisingham. Semantic web-based social network access control. *computers & security*, 30(2) :108–115, 2011.

98. Talel Abdessalem and Imen Ben Dhia. A reachability-based access control model for online social networks. In *Databases and Social Networks*, pages 31–36. ACM, 2011.

99. Saumya Omanakuttan and Madhumita Chatterjee. Trust based access control for social networks (stbac). *International Journal of Innovations in Engineering and Technology (IJIET)*, pages 325–331, 2013.

100. Anna C Squicciarini, Federica Paci, and Smitha Sundareswaran. Prima : a comprehensive approach to privacy protection in social network sites. *annals of telecommunications-annales des télécommunications*, 69(1-2) :21–36, 2014.

101. Mahnoosh Alizadeh, Seyyed Ahmad Javadi, Milad Amini, and Rasool Jalili. Policy specification and enforcement in online social networks using mknf+. In *Information Security and Cryptology (ISCISC), 2012 9th International ISC Conference on*, pages 48–53. IEEE, 2012.

102. Amirreza Masoumzadeh and James Joshi. Osnac : An ontology-based access control model for social networking systems. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 751–759. IEEE, 2010.

103. Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thuraisingham. A semantic web based framework for social network

access control. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 177–186. ACM, 2009.

104. Khalida Guesmia and Narhimene Boustia. A contextual access control model for online social network. In *COMPUTATION TOOLS 2014, The Fifth International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking*, pages 26–31. IARIA, 2014.

105. Claudio Agostino Ardagna, Sabrina De Capitani di Vimercati, Stefano Paraboschi, Eros Pedrini, and Pierangela Samarati. An xacml-based privacy-centered access control system. In *Proceedings of the first ACM workshop on Information security governance*, pages 49–58. ACM, 2009.

106. Rose Ann Zuniga and Susan Festin. A design for task-role based access control for personal health record systems. *Philippine Engineering Journal*, 38(1), 2017.

107. Yasmina Ghebghoub, Saliha Oukid, and Omar Boussaid. An mda approach to secure access to data on cloud using implicit security. *International Journal of Information and Computer Security*, 8(2) :107–120, 2016.