HAL
open science

Agents utilisateurs pour la protection des données personnelles : modélisation logique et outils informatiques

Guillaume Piolle

To cite this version:

Guillaume Piolle. Agents utilisateurs pour la protection des données personnelles : modélisation logique et outils informatiques. Intelligence artificielle [cs.AI]. Université Joseph-Fourier - Grenoble I, 2009. Français. NNT : . tel-00401295

HAL Id: tel-00401295

https://theses.hal.science/tel-00401295v1

Submitted on 2 Jul 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

UNIVERSITÉ JOSEPH FOURIER - GRENOBLE I

THÈSE

pour l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER

Spécialité : Informatique

préparée au Laboratoire d'Informatique de Grenoble – UMR 5217

dans le cadre de l'École Doctorale MSTII
(Mathématiques, Sciences et Technologies de l'Information, Informatique)
présentée et soutenue publiquement

par

Guillaume Piolle

le 2 juin 2009

Agents utilisateurs pour la protection des données personnelles : modélisation logique et outils informatiques

Directeur de thèse : Yves Demazeau
Co-directeur de thèse : Jean Caelen

JURY

M. Olivier Boissier	Examinateur
M. Jean Caelen	Co-directeur de thèse
M. Yves Demazeau	Directeur de thèse
Mme Amal El Fallah Seghrouchni	Rapporteur
M. Andreas Herzig	Examinateur
M. Yves Ledru	Président
M. Gwendal Legrand	Examinateur
M. Carles Sierra	Rapporteur

Remerciements

Définition 0.1 (Remerciements)**.** Les **Remerciements** sont un ensemble (non ordonné, donc) de couples (`nom`, `raison`) énumérant la liste des personnes que l'auteur souhaite remercier en particulier, pour avoir contribué d'une manière ou d'une autre à la réalisation de ce travail. Ce n'est que grâce à ces personnes (et à leur aide, leur soutien, leur collaboration, leur disponibilité, leurs promesses, leurs menaces, leurs coups de pied au derrière...) qu'il a pu mener à bien ses travaux de recherche et néanmoins survivre jusqu'à sa soutenance.

Les nominés sont :

- Mes encadrants, Yves Demazeau et Jean Caelen, pour m'avoir laissé toute la liberté nécessaire (et même bien davantage) à mon épanouissement scientifique ;
- Les rapporteurs de ma thèse, Carles Sierra et Amal El Fallah Seghrouchni, pour leur travail d'évaluation et leurs commentaires constructifs ;
- Les examinateurs de mon jury, Yves Ledru, Gwendal Legrand, Olivier Boissier et Andreas Herzig, pour l'intérêt porté à mon travail ;
- François Meyer, pour la conversation qui m'a poussé à faire de mon sujet de thèse ce qu'il est devenu ;
- Andreas Herzig, pour son extraordinaire disponibilité, sa gentillesse, son expertise, sa vivacité d'esprit, pour l'aide inestimable qu'il m'a prodiguée tout au long de ma thèse et pour m'avoir conseillé sur les débuts de ma vie de chercheur ;
- Antônio Carlos da Rocha Costa, pour avoir accepté de relire intégralement ma thèse et pour m'avoir fourni ses retours dans des délais absolument imbattables ;
- Olivier Boissier, pour ses retours fréquents sur mon travail et pour les références bibliographiques qu'il m'a fournies à plusieurs occasions ;
- Robert Demolombe, Philippe Balbiani, Laurence Cholvy, Frédéric Cuppens et Jean Sallantin, pour leur regard critique ou plus que critique sur mon travail et pour les pistes de travail qu'ils m'ont suggérées ;
- Jan Wielemaker et Paul Singleton, pour leur travail sur SWI-Prolog et JPL, respectivement, pour leur présence sur la liste de diffusion de SWI et pour m'avoir de nombreuses fois tiré de la substance sombre et malodorante dans laquelle mon style de programmation catastrophique m'avait préalablement plongé ;
- Markus Triska, pour avoir porté son solveur de contraintes sur domaines finis sur SWI-Prolog juste à temps pour que je puisse l'utiliser ;
- Amir Pnueli, Donald Nute et Marek Sergot, pour leur disponibilité, le temps qu'ils m'ont accordé et les conseils qu'ils m'ont prodigués ;
- Stéphanie Riché, Dorian Gaertner et Carole Adam, pour leur capacité à faire partager leur expérience et leurs travaux ;
- Pablo Seban, pour son attentif travail de relecture et ses lumières en logique modale ;
- Valérie Camps et Gaëlle Calvary, pour leur bonne humeur et leur soutien,
- Bernard Genevès, Michel Burlet, Denis Bouhineau, Pascal Sicard, Marie-Christine Fauvet, Claudia Roncancio, Christophe Bobineau ainsi que tous les enseignants et étudiants de l'université Joseph Fourier et de l'Ensimag avec lesquels j'ai travaillé, parce que les activités d'enseignement m'ont souvent aidé à ne pas lâcher prise et à me souvenir pourquoi je suis rentré dans la recherche ;
- Dominique Moreira, pour sa patience parfois mise à rude épreuve ainsi que pour sa grande compréhension de la détresse psychologique et des névroses quotidiennes du doctorant moyen ;
- Joris Deguet, pour avoir partagé avec moi un bureau trop grand et un bureau trop petit,

pour avoir encouragé ma curiosité scientifique parfois vacillante, pour m'avoir fait profiter de sa culture musicale insondable, pour avoir partagé sa voiture, son appartement, son canapé, ses amis, ses pizzas, son frigo, son temps, ses bonnes adresses, pour revenir en France de temps en temps quand même ;
- Xavier Clerc, pour ses cocktails improbables, ses compilateurs exotiques et sa DVD-thèque inaccessible ;
- Les membres passés et présents de l'équipe Magma, notamment Dimitri, Alex, Shadi Abras, Hussein Joumaa, Ludivine Crépin, Cyrille Martin, Humbert Fiorino et Julie Dugdale, pour les pauses interminables et biquotidiennes qui font la qualité de la recherche (à ce qu'il paraît), pour les discussions scientifiques, politiques, économiques, théologiques, gastronomiques, cinématographiques et musicologiques ;
- Marie-Caroline Croset, pour ne pas avoir soutenu avant moi et ne pas m'en tenir rigueur pour autant, et pour quelques années de soutien psychologique mutuel ;
- Morton Stevens, pour avoir fourni les ingrédients nécessaires à une ambiance de travail studieuse et appliquée ;
- Les Nantais, Rennais et assimilés, pour se souvenir encore de moi même après quelques années à Grenoble ;
- Mes parents évidemment, pour leurs efforts à essayer de comprendre pourquoi je ne fais pas plutôt un vrai métier ;
- Florie enfin, pour m'avoir supporté pendant presque toute ma thèse, pour avoir accepté énormément de choses, pour avoir sacrifié un certain nombre de week-ends, pour avoir fait l'effort de me relire et d'essayer de me comprendre, pour avoir copieusement enrichi la SNCF, pour avoir accepté de jouer à la roulette avec sa mutation, pour avoir essayé d'ouvrir mon esprit à ces formes de culture qui ne requièrent pas la présence d'un ordinateur, et surtout pour être encore là aujourd'hui !

Théorème 0.1 (Correction)**.** On peut aisément montrer que chaque couple (`nom`, `raison`) a effectivement toute sa place dans les remerciements. La démonstration, triviale, en est laissée en exercice au lecteur.

Conjecture 0.1 (Incomplétude)**.** On conjecture l'incomplétude des remerciements par rapport à la sémantique dite « du monde réel » (fort décriée en informatique mais néanmoins souvent utilisée comme référence). Il semble en effet que les ressources cognitives limitées de l'auteur ainsi que la quantité difficilement évaluable des éléments lui ayant permis d'aller au bout de ce qu'il avait commencé aient engendré un flou certain dans la définition des remerciements. Il est également conjecturé avec audace que les personnes omises par erreur de la définition n'en tiendront pas rigueur outre mesure à l'auteur.

Résumé

Les usages dans le domaine des systèmes multi-agents ont évolué de manière à intégrer davantage les utilisateurs humains dans les applications. La manipulation d'informations privées par des agents autonomes appelle alors à une protection adaptée des données personnelles. Les présents travaux examinent d'abord le contexte légal de la protection de la vie privée, ainsi que les divers moyens informatiques destinés à la protection des données personnelles. Il en ressort un besoin de solutions fondées sur les méthodes d'IA, autorisant à la fois un raisonnement sur les réglementations et l'adaptation du comportement d'un agent à ces réglementations. Dans cette perspective, nous proposons le modèle d'agent *PAw* (*Privacy-Aware*) et la logique DLP (*Deontic Logic for Privacy*), conçue pour traiter des réglementations provenant d'autorités multiples. Le composant de raisonnement normatif de l'agent analyse son contexte hétérogène et fournit une politique cohérente pour le traitement des données personnelles. L'agent PAw contrôle alors automatiquement sa propre utilisation des données en regard de cette politique. Afin d'appliquer cette politique de manière distante, nous étudions les différentes architectures d'applications distribuées orientées vers la protection de la vie privée, notamment celles fondées sur les principes du *Trusted Computing*. Nous en proposons une complémentaire, illustrant la possibilité d'utiliser différemment cette technologie. L'implémentation de l'agent PAw permet la démonstration de ses principes sur trois scénarios, montrant ainsi l'adaptabilité de l'agent à son contexte normatif et l'influence des réglementations sur le comportement de l'application.

Mots-clés

Systèmes multi-agents, protection des données personnelles, logique déontique, sécurité informatique, systèmes normatifs.

User agents for personal data protection: logical modelling and computing tools

Abstract

Usage in the domain of multi-agent systems has evolved so as to integrate human users more closely in the applications. Manipulation of private information by autonomous agents has then called for an adapted protection of personal data. This work first examines the legal context of privacy protection and the various computing methods aiming at personal data protection. Surveys show a significant need for AI-based solutions, allowing both reasoning on the regulations themselves and automatically adapting an agent's behaviour to these regulations. The Privacy-Aware (PAw) agent model and the Deontic Logic for Privacy, designed to deal with regulations coming from multiple authorities, are proposed here in this perspective. The agent's normative reasoning component analyses its heterogeneous context and provides a consistent policy for dealing with personal information. PAw agent then automatically controls its own usage of the data with regard to the resulting policy. In order to enforce policies in a remote manner, we study the different distributed application architectures oriented towards privacy protection, several of them based on the principles of Trusted Computing. We propose a complementary one, illustrating a different possible usage of this technology. Implementation of the PAw agent allows demonstration of its principles over three scenarios, thus showing the adaptability of the agent to its normative context and the influence of the regulations over the behaviour of the application.

Keywords

Multi-agent systems, personal data protection, deontic logic, computer security, normative systems.

Table des matières

Introduction **1**
Contexte . . . 1
Problématique . . . 1
État de l'art . . . 2
Objectif . . . 2
Thèse défendue . . . 3
Approche . . . 3
Organisation du manuscrit . . . 3

Synopsis **5**

I Éléments d'état de l'art **17**

1 Vie privée **19**
1.1 La sphère privée . . . 19
1.1.1 Nomenclature et définitions . . . 19
1.1.2 Aspects légaux du droit à la sphère privée . . . 21
1.1.3 Protection de la sphère privée au quotidien . . . 25
1.2 Les dimensions techniques de protection de la vie privée sur Internet . . . 27
1.2.1 Gestion des identités . . . 27
1.2.2 Communications IP anonymes . . . 28
1.2.3 Accès anonymes aux services . . . 28
1.2.4 Autorisation préservant la vie privée . . . 29
1.2.5 Gestion des données personnelles . . . 29
1.3 La protection des données personnelles . . . 30
1.3.1 Les six axes de la protection des données personnelles . . . 30
1.3.2 Protection locale et protection étendue . . . 32
1.3.3 Problématiques spécifiques aux domaines d'intérêt . . . 34
1.4 Technologies de protection des données personnelles . . . 36
1.4.1 *Platform for Privacy Preferences* . . . 36
1.4.2 *Sticky policies* . . . 37
1.4.3 Gestion déportée des données sensibles . . . 38
1.4.4 Agents utilisateurs . . . 38
1.5 Discussion . . . 39
1.5.1 Représentation et raisonnement . . . 39
1.5.2 Le problème de la confiance . . . 39
1.5.3 Agents humains et agents artificiels . . . 40

2 Agents cognitifs et normes **41**
2.1 Modèles d'agents . . . 41
2.1.1 Le modèle BDI . . . 41
2.1.2 Le modèle AgentSpeak . . . 42
2.1.3 Les modèles PRS, dMars et JACK . . . 43
2.2 Systèmes normatifs . . . 44
2.2.1 Caractériser les comportements appropriés . . . 44
2.2.2 Les différents types de normes . . . 44
2.2.3 Systèmes normatifs et systèmes multi-agents . . . 45
2.2.4 Systèmes d'agents normatifs et enrégimentation . . . 45
2.2.5 Subjectivité et violations . . . 46
2.2.6 Centralisation et distribution . . . 47
2.2.7 Nécessité d'un formalisme . . . 48
2.3 Éléments de logique modale . . . 48
2.3.1 Modalités . . . 48
2.3.2 Axiomatiques . . . 49
2.3.3 Sémantique de Kripke . . . 50
2.3.4 Logiques temporelles linéaires . . . 52
2.4 Logique déontique et politiques de sécurité . . . 57
2.4.1 Logique déontique standard (SDL) . . . 57
2.4.2 Politiques de sécurité . . . 58
2.4.3 Logiques déontiques et logiques temporelles . . . 59
2.4.4 Fusion de normes et gestion de l'inconsistance . . . 64
2.4.5 Formalismes alternatifs . . . 65
2.5 Synthèse et discussion . . . 66

II Outils théoriques pour la protection de la vie privée **69**

3 L'agent *PAw* **71**
3.1 Systèmes multi-agents centrés utilisateur . . . 71
3.2 Gestion des profils d'utilisateurs . . . 72
3.2.1 Modèles existants . . . 72
3.2.2 S'abstraire de la couche de gestion des profils . . . 75
3.3 Modèle d'un *Privacy-Aware Agent* . . . 75
3.3.1 Les composantes de la gestion PAw . . . 76
3.3.2 Architecture de l'agent PAw . . . 77
3.3.3 Discussion sur le modèle . . . 80

4 Appropriation des normes par l'agent PAw **81**
4.1 Le contexte normatif . . . 81
4.1.1 Autorités normatives . . . 81
4.1.2 Hétérogénéité du contexte . . . 83
4.1.3 Cohérence du contexte . . . 83
4.1.4 Dynamique des normes . . . 85
4.1.5 Synthèse des besoins . . . 86
4.2 Définition de la logique DLP . . . 86
4.2.1 Syntaxe du langage de base $\mathcal{L}_{DLP}$ . . . 87

4.2.2 Syntaxe de la logique DLP . . . 94
4.2.3 Axiomatique . . . 97
4.2.4 Structures sémantiques . . . 102
4.2.5 Sémantique de la logique DLP . . . 103
4.3 Utilisation de la logique DLP . . . 107
4.3.1 P3P et DLP . . . 107
4.3.2 Représentation des « normes datées » . . . 108
4.3.3 Exemples de normes . . . 121
4.3.4 Synthèse . . . 127
4.4 Conflits d'obligations . . . 128
4.4.1 Aperçu d'approches existantes . . . 128
4.4.2 Insuffisance de la notion existante de conflit normatif . . . 128
4.4.3 Définitions . . . 129
4.4.4 Détection des conflits d'obligations . . . 130
4.4.5 Arbitrage des conflits d'obligations . . . 131
4.4.6 Synthèse . . . 136
4.5 Impact sur les croyances . . . 137
4.5.1 Croyances sur des données personnelles . . . 137
4.5.2 Gestion normative des croyances sensibles . . . 138
4.6 Positionnement . . . 140

5 Assurer la protection étendue 141
5.1 Exigences de la protection étendue . . . 141
5.1.1 La protection étendue rapportée aux six axes réglementaires . . . 141
5.1.2 Les différents niveaux de la protection étendue . . . 143
5.2 Les technologies fondées sur le *Trusted Computing* . . . 148
5.2.1 Principes du *Trusted Computing* . . . 149
5.2.2 Une architecture d'auto-profilage (*self-profiling*) . . . 155
5.2.3 Une architecture utilisant les *sticky policies* . . . 158
5.2.4 Caractérisation des architectures possibles . . . 160
5.3 Proposition d'architecture applicative complémentaire . . . 163
5.3.1 Architecture de base . . . 163
5.3.2 Améliorations optionnelles . . . 166
5.3.3 Lien avec les institutions électroniques . . . 168
5.3.4 Critique de l'architecture . . . 169
5.4 Discussion . . . 170
5.4.1 Critiques générales de l'informatique de confiance . . . 170
5.4.2 Pertinence de l'utilisation de l'informatique de confiance pour la protection étendue des données personnelles . . . 171

III Mise en œuvre 173

6 Implémentation 175
6.1 Utilisation de la plate-forme JACK . . . 175
6.1.1 Exploitation des concepts de JACK . . . 175
6.1.2 Interface avec Prolog . . . 176
6.2 Couche de raisonnement normatif . . . 176

6.2.1 Le formalisme RDLP . . . 178
6.2.2 Représentation textuelle des formules RDLP . . . 181
6.2.3 Détection et arbitrage des conflits . . . 182
6.3 Mise en œuvre de la protection des données . . . 187
6.3.1 Activation des normes . . . 188
6.3.2 Protection locale : procédures de manipulation des croyances . . . 188
6.3.3 Éléments de protection étendue . . . 189
6.4 Discussion . . . 191

7 Évaluation **193**
7.1 Validation du modèle . . . 193
7.1.1 Validation du modèle logique . . . 193
7.1.2 Validation du modèle de gestion des croyances . . . 195
7.1.3 Validation de la gestion protocolaire . . . 196
7.2 Scénarios applicatifs . . . 196
7.2.1 Scénario de gestion médicale . . . 196
7.2.2 Scénario d'intelligence ambiante . . . 198
7.2.3 Scénario de commerce électronique . . . 200
7.3 Discussion . . . 202
7.3.1 Fusion des scénarios . . . 203

Conclusion et perspectives **205**
Synthèse des travaux . . . 205
État de l'art . . . 205
Outils théoriques . . . 205
Mise en œuvre . . . 205
Bilan . . . 206
Perspectives sur les aspects de modélisation . . . 207
Prise en compte de la paraconsistance au sein d'une autorité normative . . . 207
Raisonnement sur les croyances déontiques d'autres agents . . . 208
Perspectives sur la protection des données personnelles . . . 208
Raisonnement sur l'inférence de données personnelles . . . 208
Spécificités des croyances à caractère personnel . . . 208
Perspectives sur la gestion de la sphère privée . . . 209
Protection des données non explicites de la sphère privée . . . 209
Méthodes formelles pour la protection de la vie privée . . . 209
Ouverture : agents personnels et libertés individuelles . . . 210

IV Annexes et bibliographie **211**

A Paradoxes de la logique déontique standard **213**

B Extraction de formules DLP à partir de politiques P3P **215**
B.1 Éléments utilisés pour l'extraction . . . 215
B.1.1 `<POLICY>` . . . 215
B.1.2 `<ENTITY>` . . . 216
B.1.3 `<STATEMENT>` . . . 216
B.1.4 `<PURPOSE>` et `<PPURPOSE>` . . . 217

B.1.5 <DATA-GROUP> 217
B.1.6 <RETENTION> 217
B.1.7 <RECIPIENT> 219
B.2 Exemples d'éléments P3P ignorés à l'extraction 221
B.2.1 <DISPUTES> 221
B.2.2 <NON-IDENTIFIABLE> 222
B.3 Synthèse des prédicats de $\mathcal{L}_{DLP}$ pouvant être produites lors de l'analyse d'une politique P3P 222

C Positionnement dans le paradigme VOYELLES **223**

Bibliographie **225**

Table des figures

2.1 Boucle d'exécution du modèle AgentSpeak 43
2.2 Exemple de modèle de Kripke 51
2.3 La modalité neXt en LTL 53
2.4 Raisonnement sur le futur en LTL : les modalités G et F 54
2.5 La modalité $\mathcal{U}$ (*Until*) 55
2.6 Exemple de modèle SDL 58

3.1 Architecture de l'agent PAw 78

4.1 Grammaire du langage de base $\mathcal{L}_{DLP}$ 87
4.2 Prédicats du langage $\mathcal{L}_{DLP}$ 88
4.3 Spécificité de la modalité $\mathcal{U}^{-}$ 96
4.4 Exemple de domaine d'interprétation DLP 103
4.5 Exemple de modèle DLP (avec une seule modalité d'obligation) 106

5.1 Premier niveau de confiance de la protection étendue 146
5.2 Deuxième niveau de confiance de la protection étendue 147
5.3 Troisième niveau de confiance de la protection étendue 148
5.4 Quatrième niveau de confiance de la protection étendue 149
5.5 Schéma des éléments constitutifs d'un *Trusted Platform Module* 150
5.6 Principe de base de l'architecture d'auto-profilage 156
5.7 Amélioration optionnelle de l'architecture d'auto-profilage 157
5.8 Architecture de Casassa Mont *et al.* 158
5.9 Architecture par agents mobiles 164

6.1 Automate de fonctionnement de l'agent PAw lors de l'ajout d'une nouvelle autorité normative 177

B.1 Exemple de réécriture d'un élément `<POLICY>` 215
B.2 Exemple de réécriture d'un élément `<ENTITY>` 216
B.3 Exemple de réécriture d'éléments `<PURPOSE>` et `<PPURPOSE>` 217
B.4 Exemple de réécriture d'un élément `<DATA-GROUP>` 218
B.5 Exemples de réécriture d'un élément `<RETENTION>` 219
B.6 Exemples simples de réécriture d'un élément `<RECIPIENT>` 221
B.7 Exemple plus complexe de réécriture d'un élément `<RECIPIENT>` 221

Liste des tableaux

2.1 Correspondances entre axiomes d'une logique et propriétés de la relation d'accessibilité $\mathcal{R}$ 52

4.1 Relations entre axes réglementaires et prédicats de $\mathcal{L}_{DLP}$ 89
4.2 Lecture de l'opérateur de modélisation de la logique DLP 105
4.3 Comparaison d'opérateurs existants pour l'obligation avec échéances 116

B.1 Correspondances entre éléments P3P et prédicats de $\mathcal{L}_{DLP}$ 222

Liste des algorithmes

1 arbitrer(Δ) 134
2 valider(Δ, E) 183
3 initialiser(ListeAutorités, Module) 185
4 expand(Module) 186
5 doexpand(Module) 187

Introduction

Contexte

La notion de vie privée d'un individu (ainsi que la question de sa protection) est apparue et a évolué dans les sociétés occidentales parallèlement à l'émergence de l'ensemble des libertés individuelles. La *Magna Carta*, la Glorieuse Révolution britannique, les textes des Lumières, la Déclaration d'Indépendance des États-Unis d'Amérique ou la Déclaration des Droits de l'Homme et du Citoyen sont quelques-unes des étapes historiques témoignant de l'importance croissante donnée à l'individu dans la société. On y voit poindre les délicats équilibres et points de compromis qui existent entre les libertés individuelles et le bien collectif. La gestion de ces équilibres occupe toujours une place prépondérante dans l'organisation de la chose publique des nations policées.

Parmi ces libertés individuelles, le droit à la vie privée et à sa protection apparaît historiquement assez tôt, mais uniquement dans son principe. L'apparition de la photographie, notamment, sera à la source d'études fondatrices comme l'article "*the right of privacy*" de Samuel D. Warren et Louis D. Brandeis, posant en 1890 les fondement du droit des individus à protéger leur image et, de manière plus générale, leur vie privée [WB90].

La délimitation du droit à la vie privée (et à sa protection) n'est pas un problème trivial, car cette notion dépend de la culture, de l'histoire locale et des sensibilités individuelles. La conscience collective d'un besoin de protection de la vie privée peut notamment être fortement influencée par la mise au grand jour de failles significatives dans cette dernière. En France par exemple, le scandale du projet de fichage SAFARI, dans les années 1970, constitue une des motivations principales de la rédaction de la loi « Informatique et Libertés » [Ré78]. Une fois définie, la protection de la vie privée n'en est pas plus simple à assurer. En effet, les usagers maîtrisent rarement la formulation spécifique des textes réglementaires. Leur conception de ce qui est protégé ou pas, et dans quelle mesure, peut rester très floue.

Problématique

L'avénement de l'ère numérique, dans le dernier quart du vingtième siècle, a déclenché de nouvelles réflexions sur le sujet, les nouveaux outils induisant de nouveaux risques. De la même manière que la photographie a permis la propagation des images, Internet et les applications distribuées d'une manière générale permettent le partage, la duplication, le traitement automatisé de nombreux types d'informations. Le public prend assez rapidement conscience des possibilités offertes par les nouveaux développements techniques, mais également des risques parallèlement encourus par leurs données personnelles. Cette inquiétude de l'utilisateur va de pair avec une prise de conscience du législateur, comme en témoignent les nombreux textes réglementaires régulant la protection des données. Néanmoins, il reste un fossé entre le droit formellement

exprimé, son application plus ou moins stricte par les acteurs du monde informatique et sa compréhension par les usagers, souvent rebutés par la forme des textes.

Les exemples du commerce électronique et des réseaux sociaux, applications emblématiques associées à deux phases d'expansion du réseau internet, permettent d'illustrer ces difficultés. Dans le premier cas, l'utilisateur est forcé, pour acquérir un produit ou un service, de transmettre des informations personnelles et potentiellement sensibles, comme son adresse ou son numéro de carte de crédit. Son sentiment de sécurité est alors directement lié au niveau de confiance qu'il a dans son immatériel interlocuteur. Dans les réseaux sociaux, les utilisateurs sont incités, afin de profiter au mieux de l'environnement personnalisé, à dévoiler énormément d'informations sur eux-mêmes, sans toujours mesurer le risque associé. D'une manière générale, les utilisateurs de systèmes informatiques semblent manquer d'un moyen de mettre en relation directe les textes préservant leur vie privée et leurs données personnelles avec l'usage qu'ils en font.

État de l'art

Les nouveaux domaines d'exploration dans le domaine informatique sont à la fois des sources de risques pour la vie privée des utilisateurs et des sources d'inspiration pour de nouvelles méthodes de protection des données personnelles.

Les outils de sécurité informatique (et les méthodes cryptographiques en particulier) permettent d'assurer l'intégrité ou la confidentialité de données, mais les possibilités de ces outils dépendent énormément de la manière dont ils sont utilisés. Lorsque les besoins en matière de sécurité des données doivent être pris en compte en même temps que le besoin pour des tiers, dans un certain contexte, d'accéder à ces données, le recours aux simples méthodes de chiffrement s'avère insuffisant.

Les systèmes multi-agents, d'autre part, prennent de plus en plus en compte l'utilisateur dans le système, allant jusqu'à proposer d'associer un agent personnel à chaque utilisateur humain d'un système, dans le but d'interfacer ses interactions et de faciliter son expérience applicative. La caractéristique fondamentale d'autonomie, qui permet de différencier les agents des autres types de composants logiciels, leur permet de prendre certaines décisions techniques de manière adaptée à un contexte riche (prenant en compte les aspects techniques, les objectifs et préférences exprimés par l'utilisateur, les informations disponibles sur les autres agents du système), permettant ainsi de fournir une assistance adaptée à l'utilisateur humain.

Objectif

Dans ce contexte, il nous paraît pertinent et prometteur d'examiner comment un tel **agent utilisateur** pourrait également assurer la protection des données de son propriétaire, et notamment par le biais de méthodes issues du domaine de la sécurité informatique. Un tel agent lui éviterait ainsi le nécessaire examen des textes de loi, réglements et contrats définissant et encadrant cette protection, en effectuant une analyse automatique de ces contraintes et en adaptant son comportement de manière à lesrespecter au mieux. Les deux problèmes particuliers que cette approche doit réduire sont d'une part l'analyse des contraintes réglementaires et légales délimitant la protection de la vie privée de l'utilisateur, et d'autre part la qualité de la protection proprement dite de ces données, qui doit être assurée à la fois de manière locale et distribuée, en relation avec les dites contraintes. Pour aucun de ces deux problèmes il n'existe actuellement de solution pleinement satisfaisante, alors qu'il nous semble que c'est une étape absolument nécessaire pour faire de l'informatique un outil respectueux des libertés individuelles.

Thèse défendue

Nous proposons, dans le cadre de cette thèse, de montrer qu'il est effectivement possible de concevoir un agent utilisateur chargé de protéger les données personnelles qui lui sont confiées. L'agent que nous proposons, pour être efficace dans ce rôle, présente les caractéristiques suivantes :

- Il est capable de se représenter les diverses sources de contraintes réglementaires qui définissent la protection des données personnelles de l'utilisateur : lois, réglements, contrats, préférences...
- Il est capable de modifier son comportement en fonction de ces contraintes, de manière à pouvoir prendre en compte des variations des contraintes existantes ou l'ajout de nouvelles ;
- Il est capable de limiter son propre comportement aux actions autorisées par ces contraintes, de telle manière que les activités de l'agent utilisateur soient les plus irréprochables possible en regard des réglementations ;
- Lorsque des données qui sont sous sa garde doivent être transmises à d'autres agents dans le cadre d'une transaction concertée, il s'assure que les contraintes qu'il estime peser sur le traitement de ces données soient respectées par les autres agents, de telle manière que l'utilisateur humain puisse être convaincu que ses données seront protégées correctement au cours de ses activités.

Approche

Au vu du caractère très répandu des applications dans lesquelles ces fonctionnalités sont vitales, nous pensions pouvoir appréhender cette problématique par un assemblage judicieux de technologies existantes. Cependant, il s'est avéré qu'il demeurait des problèmes fondamentaux non résolus, ce qui a orienté notre thèse vers des aspects beaucoup plus théoriques. Notamment, il nous a fallu concevoir des outils logiques pour la représentation et le raisonnement sur les réglementations qui soient spécifiquement adaptés à la protection des données personnelles, avant de pouvoir nous appuyer dessus pour établir les deux premières caractéristiques fonctionnelles de l'agent. La limitation du comportement de l'agent (troisième caractéristique), dès lors qu'elle peut s'adosser sur ces outils logiques, peut bénéficier de techniques de contrôle d'accès dynamique préexistantes. La quatrième fonctionnalité, plus délicate car mettant en jeu des contraintes sur des agents distants et indépendants, peut être approchée en s'inspirant de certaines architectures de sécurité informatique orientées vers le contrôle de contenu.

Organisation du manuscrit

Le synopsis du manuscrit, qui suit immédiatement cette introduction, résume le contenu des différentes sections et fournit un guide de lecture du document, qui s'articule en trois parties et sept chapitres.

Dans la partie I, nous examinons les éléments d'état de l'art qui nous semblent nécessaires pour atteindre nos objectifs. Le chapitre 1 traite de la vie privée en général. Nous y définissons les termes qui nous seront utiles, nous présentons le contexte légal de la problématique et examinons comment les propositions existantes dans le domaine de l'informatique traitent généralement du problème de la protection de la vie privée des utilisateurs. Le chapitre 2 nous permet d'introduire des notions et des outils spécifiques à la recherche sur les agents autonomes ainsi que sur la notion de norme, qui nous servira à décrire les contraintes sur la protection des données personnelles.

Dans la partie II, nous décrivons notre proposition proprement dite, à savoir l'agent *PAw* (pour *Privacy-Aware*), qui respecte les spécifications informelles que nous venons d'énoncer. Le chapitre 3 présente l'agent *PAw* dans ses principes et son architecture, détaillant ses différents composants constitutifs. Dans le chapitre 4 nous mettons en place les modèles logiques qui nous permettent de représenter les contraintes en matière de vie privée et de raisonner sur ces dernières, par le biais de la logique DLP (*Deontic Logic for Privacy*). Le chapitre 5 examine comment les développements existants dans le domaine des architectures d'applications distribuées peuvent permettre de s'assurer du respect de ces contraintes par des agents tiers et propose une architecture complémentaire présentant de nouvelles caractéristiques techniques.

Dans la partie III nous expliquons comment nous avons mis en œuvre les propositions que nous avons énoncées. Le chapitre 6 décrit l'implémentation du composant de raisonnement logique ainsi que l'intégration à l'agent *PAw* des connaissances disponibles sur les architectures d'applications. Le chapitre 7 propose des éléments de validation et d'évaluation de nos propositions et de leur mise en œuvre dans le contexte de l'agent *PAw*, notamment par le biais de trois scénarios applicatifs.

Dans la conclusion, nous présentons une synthèse, un bilan fondé sur nos objectifs initiaux ainsi qu'un ensemble de perspectives de travail ouvertes par nos travaux, qui posent de nouvelles questions dans le domaine des modèles formels, de la protection des données personnelles et de la gestion de la sphère privée.

Synopsis

Partie I - Éléments d'état de l'art

1 - Vie privée et protection des données personnelles

1.1 - La sphère privée

Le terme anglais *privacy* a, suivant le contexte, de nombreuses acceptions. La traduction que l'on en fait diffère suivant le domaine d'application et le point de vue du locuteur. Nous introduisons, d'une manière générale et dans le domaine de l'informatique en particulier, les termes de *sphère privée*, *droit à la vie privée*, *protection de la sphère privée* et *protection des données personnelles*. Ces notions ont été intégrées dans la plupart des cadres légaux nationaux au cours des trente dernières années. Les textes concernés présentent une vision de la sphère privée axée sur des impératifs économiques très contextualisés, et remis à jour au fil de l'évolution des technologies et des menaces. Malgré cette législation, les individus mettent potentiellement en danger leur vie privée lorsqu'ils interagissent avec une société pour en obtenir un produit ou un service, lorsqu'ils utilisent des équipements électroniques communicants ou lorsque des informations particulièrement sensibles, comme les données médicales d'un patient, sont confiées à des traitements automatisés. L'émergence d'internet est en grande partie responsable de l'évolution des inquiétudes du grand public quant à la protection de sa vie privée en général, et de ses données sensibles en particulier.

1.2 - Les dimensions de la protection de la vie privée

Deswarte propose en 2006 une classification en cinq domaines des techniques de protection de la vie privée sur internet. Nous reprenons ici cette classification. La **gestion des identités** recouvre les technologies qui permettent à l'utilisateur de masquer sa véritable identité. Cet objectif peut notamment être atteint par l'utilisation de différentes identités virtuelles, correspondant à différentes applications, différents profils comportementaux ou différents niveaux de sécurité, et entre lesquels il est difficile d'établir des corrélations. D'autre part, de nombreuses technologies ont été développées pour permettre l'établissement de **communications IP anonymes**, que ce soit pour des applications pair-à-pair fortement distribuées et à accès essentiellement anonyme, ou bien pour des applications client-serveur avec une notion de session très forte. Des techniques d'obfuscation, de brouillage statistique ou de redirections peuvent permettre de garantir à l'utilisateur que ses accès applicatifs ne pourront pas être associés à son adresse IP par un observateur du réseau. Les **accès anonymes aux services**, quant à eux, concernent l'anonymisation des messages de l'utilisateur au niveau du protocole applicatif. En effet, dans le cas général, la communication entre utilisateur et service comporte beaucoup d'informations identifiantes, et il est parfois possible de limiter la portée de ces informations sans détériorer la qualité du service. Il existe également des techniques d'**autorisations préservant la vie**

privée, permettant de séparer la phase d'authentification de la phase d'autorisation d'accès, afin que l'utilisateur puisse accéder de manière anonyme au service. L'identité, ou d'autres caractéristiques requises, peut être vérifiée par un tiers indépendant du service, et l'utilisateur peut accéder au service en présentant des certificats anonymes émis par ce tiers de confiance. La **gestion des données personnelles** couvre de son côté les dispositifs de protection des données qui sont susceptibles d'identifier un agent utilisateur humain, artificiel ou institutionnel. Elle concerne notamment la protection des informations de profil, de personnalisation, les préférences utilisateur transmises à une application, les identifiants et les requêtes faites au nom de l'utilisateur. C'est sur ce dernier domaine de la protection de la vie privée que portent nos travaux.

1.3 - La protection des données personnelles

L'analyse des textes légaux et réglementaires, ainsi que des recommandations courantes, permet de classer en six axes les éléments constitutifs des politiques de protection des données personnelles. Ces axes sont l'**information** de l'utilisateur, son **consentement**, son droit à **modifier** ou supprimer les données prélevées, la **justification** de la collecte de données personnelles, la durée de **conservation** et les conditions de **transmission** de ces données à des tiers. Il existe d'autres classifications, suivant des notions transverses à ces six dimensions. Nombre d'applications qui se réclament de la protection des données personnelles se concentrent sur le problème du contrôle d'accès aux données. Outre le fait que le contrôle d'accès n'est pas le seul problème à traiter, il est généralement considéré avec un point de vue strictement local. Or, le problème de la protection des données personnelles est essentiellement distribué, il faut donc définir une **protection étendue**, dont la portée suivrait la diffusion des données. Il est possible de retrouver toutes ces problématiques dans quelques domaines spécifiques. Dans le domaine de la santé, la mise en œuvre des dossiers médicaux personnalisés impose des contraintes fortes et spécifiques sur la manipulation de données particulièrement sensibles pour leurs propriétaires. En informatique ambiante, les objets de l'environnement interagissent constamment entre eux, échangent des informations et produisent des traces qui peuvent être agrégées en profils comportementaux. Dans le cadre du commerce électronique, les transferts d'informations personnelles sont une nécessité absolue, et l'utilisation qui en est faite est rarement maîtrisée. Ces trois domaines sont susceptibles d'inspirer des scénarios d'usage pour les solutions de protection des données personnelles.

1.4 - Technologies de protection des données personnelles

Le standard P3P du *World Wide Web Consortium* est un outil désormais incontournable de la communication des sites web sur leur politique de protection des données personnelles. Les schémas XML et les outils correspondants peuvent être intégrés dans des applications pour gérer l'information et le consentement automatique de l'utilisateur. Une autre approche courante (et commune à de nombreuses propositions) consiste à attacher aux informations sensibles les méta-données de description de la politique de sécurité associée, les applications s'engageant à respecter cette « politique collante ». Si le principe en est intuitif, pratique et adapté à la distribution des applications et des données, il ne donne cependant aucune garantie à l'utilisateur quant au respect de la politique par une application distante. Des propositions ont également été faites pour déporter la gestion des données utilisateur sur un serveur spécialisé, mettant en œuvre des mécanismes de contrôle d'accès sophistiqués, visant à s'assurer du bien-fondé des différentes demandes d'accès au profil qui lui sont faites. Ces approches ont apporté des idées intéressantes,

notamment dans le cadre de la gestion des identités virtuelles, mais souffrent de limitations discriminantes. Dans certaines propositions, la gestion des données de l'utilisateur humain est confiée à un agent artificiel (qui prend alors le nom d'agent assistant ou agent personnel) chargé d'interfacer ses interactions avec différents services et applications. Dans cette perspective, c'est l'agent assistant qui met en œuvre les différents mécanismes (principalement de contrôle d'accès) assurant la sécurité des données personnelles de l'utilisateur.

1.5 - Discussion

Quelle que soit la solution technique choisie pour assurer la protection des données, il faut, pour respecter au mieux le cadre réglementaire et légal, que cette dernière permette une représentation et un raisonnement sur les réglementations en vigueur, afin de pouvoir s'adapter dynamiquement à celles-ci. Dans toutes les propositions présentées, fondées sur un contrôle local de l'accès aux données, la conviction de l'utilisateur que la politique de protection des données sera respectée par un agent distant repose sur sa seule confiance dans cet agent. Ce dernier étant précisément l'entité à qui profiterait (le plus souvent) une violation de cette politique, la confiance par défaut de l'utilisateur dans un environnement ouvert, tel qu'internet, doit rester très faible. Il y a donc ici un besoin de reporter la confiance de l'utilisateur sur un agent qui n'aurait pas d'intérêt à la violation de la politique. D'autre part, les problématiques que nous venons de présenter concernent essentiellement les utilisateurs humains du système, et les solutions proposées sont orientées par cette considération. De ce fait, les propositions que nous avons décrites ne comprennent aucune automatisation de nature à mettre en relation les exigences de la problématique avec les techniques utilisées. L'association entre ces agents humains et des agents artificiels servant d'interface avec les applications et les services permettrait d'automatiser nombre de procédures et de décisions. Il convient donc de s'intéresser à ce que le paradigme des agents artificiels (et notamment cognitifs) peut apporter pour la protection des données personnelles.

2 - Agents cognitifs et normes

2.1 - Modèles d'agents

Les travaux de recherche dans le domaine des systèmes multi-agents comportent de nombreuses propositions de modèles pour les agents autonomes, et notamment cognitifs. Le modèle *Belief-Desire-Intention (BDI)* est sans doute le plus connu et le plus utilisé. Il trouve des déclinaisons dans la formalisation logique du modèle mental des agents, ainsi que dans des modèles procéduraux de développement d'agents logiciels comme PRS (*Procedural Reasoning System*).

2.2 - Systèmes normatifs

Les systèmes normatifs sont des systèmes dans lesquels le comportement idéal ou autorisé est décrit par des règles formelles : les normes. Dans ces systèmes, souvent appliqués à des systèmes multi-agents, on caractérise la notion de violation des normes par les agents, et les conséquences de ces violations. Il existe différents types de systèmes normatifs (centralisés ou non, omniscients ou non), qui sont plus ou moins adaptés pour modéliser les problèmes de protection des données personnelles.

2.3 - Éléments de logique modale

La bonne compréhension des propositions que nous allons faire nécessite l'introduction des notions de base de la logique modale, des outils et des formalismes que nous utilisons. Les modalités sont des opérateurs exprimant un concept plus ou moins complexe à propos de la formule sur laquelle ils sont appliqués. Elles vont en général par paires, une modalité de nature universelle étant associée à sa modalité duale, de nature existentielle. Les logiques modales dites normales s'appuient sur des règles d'inférences et des axiomes communs, et peuvent s'enrichir d'axiomes supplémentaires en fonction des notions à représenter. La sémantique de Kripke (une sémantique de mondes possibles) est la plus courante pour les logiques modales normales. Les mondes peuvent y représenter des états ou des exécutions possibles du système, et sont reliés entre eux par une relation binaire dite d'accessibilité, dont les propriétés sont étroitement liées à l'axiomatique de la modalité correspondante. Un exemple de logique modale normale est la famille des logiques temporelles linéaires, dans lesquelles les modalités représentent des notions liées au passé, au futur ou à la précédence temporelle d'événements. Dans ces logiques, les mondes représentent des instants, et sont reliés par une relation d'accessibilité en une chaîne linéaire qui va du passé vers le futur. Ces logiques servent à décrire de manière formelle des propriétés sur les successions temporelles d'événements.

2.4 - Logique déontique et politiques de sécurité

La logique déontique standard (SDL) est également une logique modale normale, où la modalité universelle représente la notion d'obligation ou de devoir. Cette logique permet également d'exprimer le caractère interdit, permis, facultatif ou gratuit d'une formule. SDL est généralement acceptée comme base pour les travaux sur la logique déontique, et notamment pour la représentation de politiques de sécurité. L'intérêt de la notion de politique de sécurité est de représenter de manière formelle les règles de fonctionnement d'un système, permettant ainsi de raisonner dessus ou d'y adapter automatiquement des processus. Les travaux dans le domaine portent notamment sur les vérifications de cohérence des politiques. Le besoin d'intégrer la notion de temps aux politiques à conduit à plusieurs propositions associant aux notions de la logique déontique les possibilités offertes par les logiques temporelles, notamment linéaires. Une telle association permet de mieux caractériser la notion de violation d'une obligation, ou encore d'exprimer des obligations comportant des durées de validité, ou des dates limites. Ces notions sont en effet courantes dans les régulations de la vie courante, et notamment dans les politiques de protection des données personnelles. De nombreux travaux existent également concernant les mécanismes de fusion de normes ou d'ensembles de normes. En effet, un agent doit souvent considérer plusieurs ensembles de normes différents (par exemple parce qu'il joue plusieurs rôles simultanément, ou qu'il appartient à plusieurs systèmes normatifs distincts), et ces ensembles de normes peuvent être incompatibles les uns avec les autres. Les inconsistances logiques ainsi générées peuvent par exemple être gérées par des techniques de fusion fondées sur des relations de préférence. D'autres formalismes logiques permettent de raisonner sur les normes, le temps et les inconsistances. Nous examinons par exemple le cas des logiques STIT, des logiques temporelles fondées sur la notion de fluent, des logiques paraconsistantes et des logiques défaisables.

2.5 - Synthèse et discussion

Les travaux existants formulent des propositions pour des problèmes qui se posent forcément lorsqu'un agent artificiel doit s'adapter aux diverses contraintes existantes en matière de protection des données personnelles. Cependant, aucun n'a un point de vue portant spécifiquement

cette orientation, il reste donc des questions en suspens qui doivent être traitées dans ce contexte particulier. En effet, suivant le rôle exact que l'on donne aux normes dans le système, les problèmes peuvent se poser de manières différentes.

Partie II - Outils théoriques pour la protection de la vie privée

3 - L'agent *PAw*

L'état de l'art en matière de protection des données personnelles met en exergue un certain nombre de principes et de bonnes pratiques, mais comporte encore principalement deux lacunes. La première est que les démarches d'automatisation de la gestion sont très rares et encore embryonnaires, alors que les travaux en matière de personnalisation et d'organisation des applications distribuées autour de l'utilisateur en montrent le besoin. Le second manque concerne la protection étendue des données, quasiment absente des solutions existantes. En réponse à ce constat, notre contribution consiste en la conception d'un type d'agent artificiel assistant (l'agent PAw, ou *Privacy-Aware Agent*), lié à son agent humain utilisateur et chargé d'interfacer ses interactions avec des applications de service tout en prenant en charge la protection de ses données personnelles.

3.1 - Systèmes multi-agents centrés utilisateur

La recherche sur les systèmes multi-agents a évolué au cours des dernières années pour mettre un accent particulier sur le rôle de l'utilisateur dans la compréhension et la conception du système. La notion, essentiellement applicative, de systèmes multi-agents centrés utilisateur désigne notamment les architectures dans lesquelles les utilisateurs humains développent une interaction privilégiée avec certains agents artificiels du système, qui leur servent d'interface.

3.2 - Gestion des profils utilisateur

De nombreuses approches proposent des modèles pour la représentation et le traitement des profils d'utilisateurs. Les informations peuvent être structurées sous diverses formes, centralisées ou distribuées, et peuvent être réduits à quelques types de données bien précis : des tuples attributs-valeurs, des règles logiques ou des journaux d'historique bruts. La plupart du temps, les problématiques liées à la sécurité des informations du profil ne sont pas abordées, ou alors d'une manière trop limitée. La nature des six axes de la protection des données personnelles peut cependant permettre de s'abstraire de la structure générale d'un modèle de profil, pour s'intéresser à une représentation intermédiaire des informations, utilisée pour la transmission de données à un service. Il est possible de construire un modèle abstrait permettant d'interfacer le moteur de gestion du profil utilisateur, quel qu'il soit, afin de ne traiter que des informations dans un format homogène.

3.3 - Modèle d'un *Privacy-Aware Agent*

La contribution proposée est axée autour de l'architecture d'agent *PAw* (*Privacy-Aware*), un agent cognitif conscient de son contexte normatif en matière de protection des données personnelles. Ce modèle intègre des fonctionnalités de raisonnement logique sur les réglementations à respecter et un modèle appliqué des outils techniques à mettre en œuvre pour cela. Tout d'abord, un agent PAw doit assurer un certain nombre de fonctionnalités essentielles. Il doit notamment être capable de représenter explicitement des données personnelles (appartenant

à son propriétaire ou à un autre utilisateur), de prendre connaissance des réglementations qui s'appliquent à son exécution courante, d'en déduire un ensemble de normes cohérent à respecter, d'asservir la gestion des données personnelles à ces normes et de choisir de manière dynamique les protocoles applicatifs à mettre en œuvre pour assurer la protection étendue de ces données. Ces fonctionnalités sont intégrées à l'agent PAw au sein d'une organisation en couches, suivant une architecture implémentant un modèle BDI. La couche logique assure les fonctionnalités de raisonnement sur les réglementations et politiques de sécurité, et fournit à la couche inférieure un ensemble de normes. La couche de gestion des croyances prend en charge les croyances concernant les normes en question, les croyances identifiées comme personnelles, les croyances sur la caractérisation des protocoles applicatifs et les croyances métier de l'agent. La couche des buts contient les buts normatifs de l'agent, qui caractérisent son objectif de respecter les normes, ainsi que ses buts métier. La couche des plans, qui est la couche la plus basse en termes de niveau d'abstraction, contient les procédures effectives mises en œuvre par l'agent, qui sont les plans de gestion des croyances personnelles respectant les normes, les plans métier liés aux activités applicatives de l'agent et les plans de mise en œuvre des protocoles applicatifs. La structure générale de l'agent PAw est fondée sur les travaux relatifs aux modèles BDI, et profite principalement des principes du modèle PRS. Les spécifications du modèle mental de l'agent peuvent être construites sous forme logique (et ainsi profiter des méthodes afférentes aux systèmes normatifs), mais les traitements nécessités par la gestion des données personnelles et la mise en place de protocoles de sécurité potentiellement complexes appellent des outils procéduraux, plus aisément configurables qu'un moteur purement logique.

4 - Appropriation des normes par l'agent PAw

4.1 - Le contexte normatif

Nous définissons la notion d'**autorité normative** comme une entité qui émet des normes dont elle est responsable. Cette notion peut être comparée (mais pas assimilée) à celles d'institution ou de dépôt de normes. Dans le cadre qui nous intéresse, l'agent PAw reconnaît un certain nombre d'autorités normatives, qui émettent des normes et qui exigent qu'elles soient respectées par l'agent. Ces normes constituent le **contexte normatif** qui s'applique lors d'une transaction ou d'une collaboration avec un autre agent. Au sein de ce contexte normatif, l'agent PAw doit respecter les normes émanant de plusieurs autorités normatives indépendantes. Suivant le ressort législatif dont dépendent son propriétaire, la machine sur laquelle il s'exécute et l'agent avec lequel il communique, l'agent PAw est soumis à plusieurs systèmes de lois nationales ou internationales. Il peut également dépendre de divers réglements, contrats ou encore des exigences de son utilisateur humain. Les différentes autorités normatives auxquelles est soumis l'agent PAw sont a priori faillibles et indépendantes les unes des autres. Il peut donc arriver que plusieurs autorités normatives émettent des normes qui entrent en conflit les unes avec les autres. Dans ce cas, il faut mettre au point une technique de fusion de normes qui permette à l'agent de construire un nouvel ensemble de normes exempt de conflits. De plus, suivant le profil applicatif de l'agent PAw, celui-ci est susceptible de changer de contexte normatif très fréquemment, par exemple s'il est mobile au niveau international, s'il communique avec des agents sous le couvert de différents contrats ou bien s'il est en charge à la fois d'affaires professionnelles et personnelles pour son utilisateur humain. Le contenu des normes édictées par les autorités normatives peut également varier au cours du temps. Le module de gestion des normes de l'agent PAw doit donc pouvoir gérer un contexte normatif hétérogène, dynamique et possiblement incohérent.

4.2 - Définition de la logique DLP

Nous prenons en compte les nombreux développements dans le domaine de la logique déontique et de la représentation des politiques de sécurité pour proposer une logique qui soit à même de représenter les réglementations en matière de protection des données personnelles. La logique DLP permet la gestion du contexte normatif en vue de l'adaptation du comportement de l'agent PAw à celui-ci. La logique DLP utilise un langage propositionnel de base, $\mathcal{L}_{DLP}$, qui décrit les informations relatives aux données personnelles et à leur traitement. La grammaire de $\mathcal{L}_{DLP}$ décrit les six axes de la protection des données personnelles. On définit alors la logique DLP comme une logique déontique et temporelle normale fondée sur un produit entre la logique déontique standard (adaptée pour prendre en compte la multiplicité des autorités et des agents) et la logique temporelle linéaire, appliqué au langage $\mathcal{L}_{DLP}$. Les formules DLP sont donc des normes décrivant des politiques ou des réglementations sur la gestion de ces données. L'axiomatique que nous proposons est celle de la logique déontique, pour la partie normative, et d'une sélection d'axiomes classiques pour la partie temporelle. Nous discutons la pertinence d'éventuels axiomes de conversion entre les deux notions. La logique DLP est interprétée sur des structures sémantiques bidimensionnelles, les *domaines d'interprétation DLP*, dont un axe représente le temps et l'autre les histoires possibles du système. Nous définissons une sémantique de Kripke pour la logique DLP, fondée sur les domaines d'interprétation précédemment introduits. Nous introduisons les relations d'accessibilité (temporelles et déontiques) correspondant aux diverses modalités utilisées et nous détaillons les règles d'interprétation des formules DLP.

4.3 - Utilisation de la logique DLP

Nous avons vu que le standard P3P est un moyen efficace de communiquer sur les politiques de protection des données personnelles. Cependant, il manque à ce formalisme la notion de norme et les capacités de raisonnement, c'est pourquoi nous avons introduit la logique DLP. Cependant, il paraît essentiel, pour des raisons d'intéropérabilité, de prévoir un moyen d'extraire des formules DLP à partir de politiques P3P. D'autre part, les notions à la fois déontiques et temporelles que sont les obligations avec échéances et les interdictions maintenues sont nécessaires à l'expression des réglementations en matière de protection des données personnelles. Nous discutons des critères à respecter pour concevoir de tels opérateurs et proposons une formalisation adaptée à DLP. Sur la base de quelques réglementations, en langue naturelle, traitant des six axes de la protection des données personnelles, nous montrons comment le langage DLP peut être utilisé pour formaliser les normes.

4.4 - Conflits d'obligations

De nombreux travaux existent dans le domaine de la détection et de la résolution des conflits en logique déontique. Nous détaillons quelques approches existantes ainsi que leurs caractéristiques. Il s'avère que la notion habituellement utilisée pour caractériser les conflits est inappropriée dans notre cas, car en donnant une place trop importante à la notion de permission, elle restreindrait abusivement les agissements de l'agent PAw dans des cas où une solution satisfaisant l'ensemble des normes est possible. En effet, les normes que nous considérons ici servent à contraindre le comportement de l'agent. Dans le cadre de cette utilisation très précise, une permission ne restreint pas les possibilités d'action et doit donc avoir un statut à part. Partant de ce constat, nous proposons une définition du conflit d'obligation fondée uniquement sur les obligations et ignorant les permissions, afin de s'affranchir de certaines inconsistances logiques non pertinentes. Nous proposons une procédure théorique de détection des conflits d'obligations

fondée sur une caractérisation purement axiomatique de ces derniers. Pour produire un ensemble de normes cohérent à partir d'un ensemble comportant des conflits, l'agent PAw met en place une procédure de fusion en se basant sur une relation de prévalence entre les autorités normatives. Les autorités normatives que l'agent juge plus importantes ont une prévalence plus élevée que celles que l'agent juge secondaires. Au cours de cette procédure d'arbitrage des conflits d'obligations, l'agent désactive un nombre minimal de normes, édictées par des autorités normatives de prévalence minimale.

4.5 - Impact sur les croyances

Les modèles classiques de gestion des croyances par un agent cognitif ne prennent pas en compte les spécificités de la protection des données personnelles. Nous proposons des moyens pour modifier les modèles existants de manière à permettre le traitement différencié de ces croyances particulières. En considérant à la fois les croyances normatives de l'agent sur la protection des données personnelles et ses croyances portant sur des informations sensibles, il est possible de construire des procédures spécifiques mettant en œuvre des traitements sur les données personnelles tout en respectant le contexte normatif de l'agent.

5 - Assurer la protection étendue au sein d'une transaction entre agents

5.1 - Exigences de la protection étendue

La protection étendue des données, telle que nous l'avons définie, nécessite de pouvoir garantir qu'une politique donnée s'applique bien aux informations sensibles, qu'elles soient gérée par l'agent personnel de l'utilisateur ou bien par un agent distant. À chacun des six axes de la protection des données personnelles correspondent des exigences spécifiques. Si l'on veut les satisfaire pleinement, il est nécessaire de caractériser la confiance que l'on peut avoir dans l'exécution d'un programme distant, dans le système d'exploitation distant, et dans le matériel de la plate-forme distante.

5.2 - Les technologies fondées sur le *Trusted Computing*

Les solutions de nature logicielle présentées précédemment échouent à assurer cette protection étendue. Les architectures utilisant les *Trusted Computing Platforms*, au contraire, peuvent fournir des garanties fortes sur le matériel et le logiciel distant. Cette technologie est notamment fondée sur l'existence d'une composant cryptographique matériel sur la plate-forme. Nous proposons une classification des architectures et protocoles fondés sur le *Trusted Computing* en fonction de la localisation des différents composants, de l'existence et de l'utilisation des tiers de confiance, ainsi que d'autres paramètres. Cette classification permet de déterminer les exigences (en termes de protection étendue) qui peuvent être assurées ou pas par l'architecture et les risques qu'elle fait courir à la vie privée de l'utilisateur.

5.3 - Proposition d'architecture applicative complémentaire

Nous proposons une nouvelle architecture profitant des possibilités du *Trusted Computing*. Cette architecture est orientée au maximum vers la protection de la vie privée de l'utilisateur, quitte à poser des restrictions sur le type de traitement distant envisageable et sur les garanties offertes aux fournisseurs de service. Cette architecture applicative vient compléter l'offre déjà existante, en offrant un maximum de garanties à l'utilisateur dans les cas où elle peut être utilisée.

5.4 - Discussion

Si le principe des *Trusted Computing Platforms* permet d'augmenter sensiblement la confiance dans les caractéristiques d'un traitement distant (chose particulièrement difficile à assurer sans contrainte sur le matériel distant), ce n'est pas nécessairement la seule solution possible à ce problème. D'autre part, il convient de prendre acte des dangers réels que certains types d'implémentations et d'utilisations peuvent faire courir à la vie privée des utilisateurs individuels.

Partie III - Mise en œuvre

6 - Implémentation

6.1 - Utilisation de la plate-forme JACK

Nous proposons l'implémentation d'un démonstrateur du modèle d'agent PAw à partir d'une architecture d'agent JACK, à l'aide de la plate-forme de développement correspondante, constituée d'une surcouche orientée agent du langage Java inspirée par le modèle PRS. Cette plate-forme dispose notamment de fonctionnalités permettant une mise en œuvre efficace de procédures de sécurité pour contrôler l'accès aux croyances sensibles.

6.2 - Couche de raisonnement normatif

La couche logique de l'agent fait usage d'une machine virtuelle Prolog, interfacée avec le modèle d'agent JACK. Au sein de la logique DLP, nous définissons un schéma de formules particulier, nommé RDLP. Nous posons l'hypothèse simplificatrice, restrictive mais justifiée, que les autorités normatives émettent des normes qui sont des formules sous la forme RDLP. Cette hypothèse permettra par la suite des traitements plus efficaces pour la détection et la résolution des conflits d'obligations. Lorsqu'il reçoit un nouvel ensemble de normes en provenance d'une autorité, l'agent PAw commence par vérifier que cet ensemble est acceptable, avant de tenter de l'intégrer à sa base de normes existante. Un moteur d'inférence écrit en Prolog lui permet de dériver une contradiction dans l'ensemble total des normes (déterminant ainsi un conflit d'obligations), d'identifier les normes individuelles participant au conflit et d'isoler celle qui doit être désactivée pour résoudre le conflit.

6.3 - Mise en œuvre de la protection des données

Lors de la phase d'arbitrage, l'agent produit un ensemble de normes actives (sa politique interne) et un ensemble de normes désactivées. Les normes actives sont examinées à chaque fois que l'agent doit manipuler ou transmettre des données à caractères personnel. Elles servent également à planifier des actions prévues par le contexte normatif, comme les suppressions de données. En lien avec ces normes, nous avons proposé un modèle de croyance spécifique pour la représentation des données personnelles. Il nous est donc possible, dans le cadre d'une implémentation en JACK, de les stocker dans une base de croyances séparée et de mettre en œuvre des politiques de restriction d'usage de ces informations par le biais du mécanisme de fonctions de rappel de JACK sur la manipulation des bases de croyances. L'agent PAw dispose de plus de connaissances factuelles sur l'ensemble des architectures applicatives que nous avons analysées et sur leurs diverses options possibles. Il est ainsi en mesure de déterminer, pour chaque interaction avec un agent tiers, quelle est l'architecture la plus adaptée au respect des normes

actives. L'introduction d'éléments de négociation lui permet de faire en sorte que l'architecture finalement choisie soit la plus favorable possible.

6.4 - Discussion

L'implémentation que nous proposons est un démonstrateur, conçu pour mettre en évidence les possibilités apportées, en termes de fonctionnalités pratiques, par nos développements théoriques. Elle reste limitée notamment en ce qui concerne les capacités de déduction logique de l'agent et la mise en œuvre réelle des architectures applicatives.

7 - Évaluation

7.1 - Validation du modèle

Nous examinons les restrictions posées en termes d'expressivité par l'utilisation du formalisme RDLP et nous assurons qu'elles ne limitent pas les capacités de l'agent PAw à détecter et arbitrer les conflits d'obligations. Nous caractérisons également les types de normes impossibles à représenter en RDLP et proposons des moyens pour contourner ces limitations. D'autre part, le modèle de représentation que nous proposons pour les croyances portant sur les données personnelles, associé aux mécanismes JACK mis en œuvre, doit permettre d'assurer une protection locale de ces informations. L'implémentation du démonstrateur reposant sur des informations de nature déclarative sur les plans métier, la protection locale des données personnelles reste sensible à certaines attaques. En ce qui concerne les différentes architectures d'application, leur implémentation dépasse le cadre de nos travaux. La gestion protocolaire n'étant mise en œuvre dans le démonstrateur que par le biais du raisonnement sur les informations caractéristiques des différents protocoles, elle n'est donc pas sujette à validation directe. L'utilisation des différents protocoles sur des scénarios applicatifs permet d'illustrer le caractère adapté ou non de telle architecture à tel type d'application.

7.2 - Scénarios applicatifs

Nous avons précédemment identifié l'informatique médicale, l'intelligence ambiante et le commerce électronique comme des domaines d'intérêts pour la conception de scénarios applicatifs. Le premier scénario se situe dans le domaine de la gestion d'un dossier médical informatisé. L'agent PAw est ici responsable de données médicales appartenant à des patients et susceptibles d'être consultées par le personnel médical et administratif d'un établissement hospitalier. Il devra faire en sorte que la consultation et l'utilisation de ces données particulièrement sensibles se fasse conformément à divers jeux de règles. Le deuxième scénario que nous proposons est orienté vers l'intelligence ambiante. Ici, l'agent PAw est embarqué dans un artéfact intelligent interagissant avec son environnement. Il devra organiser sa collaboration avec les autres artéfacts du système, en limitant la diffusion d'informations sensibles. Le troisième scénario applicatif proposé place l'agent PAw dans la situation d'un agent personnel dans le cadre d'une transaction de commerce électronique. Il doit interagir avec des agents services et des agents tiers de confiance, en vue d'obtenir un service commercial dans un environnement ouvert de type internet. La personnalisation de ce service nécessitant des informations de profil de l'utilisateur humain, l'agent PAw devra gérer au mieux la protection de ces données.

7.3 - Discussion

Les résultats de l'évaluation de l'agent PAw doivent être analysés en considérant les limitations intrinsèques de son implémentation. Notamment, la robustesse et l'utilisabilité des architectures applicatives de type *Trusted Computing* ne sauraient être évaluées sans une réelle mise en œuvre matérielle.

Première partie

Éléments d'état de l'art

Chapitre 1

Vie privée et protection des données personnelles

Dans ce premier chapitre, nous nous intéressons aux notions de vie privée et de protection des données personnelles telles qu'elles sont considérées dans le domaine de l'informatique. Nous étudierons la nature de ces concepts, la manière dont ils sont traités dans la réglementation ainsi que les moyens techniques qui leur sont rattachés.

1.1 La sphère privée

Nous commençons par nous intéresser à la notion de vie privée en général, en-dehors du cadre informatique. Il nous faut clarifier les termes que nous allons utiliser, avant de nous pencher sur les problèmes qu'ils peuvent poser du point de vue du « législateur » et du point de vue individuel.

1.1.1 Nomenclature et définitions

Les notions liées à la vie privée peuvent difficilement être définies sans s'intéresser aux différentes acceptions du terme *privacy* en anglais. En effet, si l'on peut le faire correspondre en français à la notion assez générale de « caractère privé » d'une chose, ce mot semble être considéré comme recouvrant un certain nombre de concepts liés. Suivant leur culture et leur point de vue, les auteurs les plus consciencieux prennent soin de préciser ce que désignent pour eux le terme *privacy*, sans l'utiliser indifféremment pour *right of privacy* ou *privacy protection*, par exemple. Nous considérerons ici le terme *privacy* comme une traduction imparfaite de la locution « vie privée », nous autorisant par la suite à définir des termes dérivés.

La mention du concept de vie privée éveille chez tout un chacun un ensemble de problématiques liées à notre vie quotidienne ou à notre perception de procédés techniques ou liés à un certain contexte professionnel. Ainsi, la capacité à cacher un certain nombre de choses sur soi à un journaliste, au public en général, à des collègues, à des connaissances, relève indubitablement de notre droit à la vie privée. La notion de surveillance des activités d'un individu, l'enregistrement ou le traitement d'informations le concernant, le fait d'entrer en communication avec lui sur la base des résultats d'un tel traitement sont autant d'actions en lien étroit avec la notion de vie privée ou de sphère privée. Le concept semble donc composite et par conséquent difficile à cerner. Néanmoins, assez curieusement, certains auteurs ont proposé des définitions très lapidaires dont on peut se demander si elles correspondent vraiment à cette vision naïve et intuitive de la vie privée.

En 1967, Alan Westin définit ainsi le terme *privacy* [Wes67], adoptant un point de vue de type juridique, confondant en un seul et même mot la chose et le droit à la chose :

> "Right of individuals to determine for themselves when, how and to what extent informations about them is communicated to other."

La vie privée est donc essentiellement une question de gestion des flux d'informations se rapportant à une personne physique. Günter Müller adopte un point de vue assez similaire mais plus abstrait [Mü06] en définissant ainsi la *privacy* :

> "Possibility to control the distribution and use of personal data."

Ces deux points de vue semblent limiter la notion de vie privée à une diffusion maîtrisée d'informations. Cela peut paraître frustrant au premier abord au vu des procédés relativement complexes et détachés de toute considération technique que nous venons de mettre en relation avec la notion de vie privée.

Certains auteurs partent de ce constat pour poser des définitions à vocation plus générale, incluant certaines de ces notions. Ainsi, Sara Baase propose par exemple dans son livre *The gift of fire* [Baa03] une acception légèrement différente et sans doute plus « parlante », suivant laquelle le mot *privacy* peut signifier indifféremment :

- L'absence d'intrusion ;
- Le contrôle des informations nous concernant[1] ;
- L'absence de surveillance.

Nous incluons donc ici explicitement deux nouveaux processus dans le concept de vie privée, processus qui semblent se rapporter à notre tentative de description intuitive de la vie privée. Il en ressort donc légitimement une certaine satisfaction. Cependant, cette circonscription de la notion de vie privée peut également paraître peu naturelle à cause d'un certain téléscopage conceptuel introduit par ces nouvelles notions. En effet, l'intrusion comme la surveillance sont craintes en tant qu'elles sont des menaces pour le contrôle de nos informations. Cette tentative de définition, comme nombre d'autres, amène à penser que les concepts de la vie courante que nous associons instinctivement à la notion de vie privée se réduisent effectivement tous à un problème de contrôle de l'information. Il est aisé en effet de considérer un à un les éléments que nous avions inclus dans la notion de vie privée, pour les exprimer sous la forme d'une manipulation d'informations.

Ceci posé, nous pouvons formaliser quelques définitions liées à la vie privée à partir des éléments déjà recueillis, afin de faire référence par la suite à des concepts clairs et distincts. Plutôt que d'utiliser le terme de « vie privée » qui, nous l'avons vu, peut s'avérer assez vague à cause même de son écho intuitif, nous prendrons appui sur le concept équivalent de **sphère privée**, notamment formalisé par Ludivine Crépin *et al* [CVJ+08].

Définition 1.1 (Sphère privée)**.** La sphère privée d'un individu est l'ensemble des informations, se rapportant à lui-même, qu'il considère comme sensibles et donc dignes d'être protégées. Cette sphère est **personnelle** (l'individu est le propriétaire des informations qu'elle contient), **personnalisable** (l'individu décide des informations qu'elle contient), **dynamique** (les informations peuvent y être ajoutées ou en être retirées) et **dépendante du contexte** (les informations qu'elle contient peuvent, en nature et en nombre, dépendre du temps, des activités de l'individu ou d'autres paramètres).

[1] Dans ce contexte, lesdites informations peuvent être de deux types : ou bien des faits (des actions ayant été exécutées) ou bien des données personnelles (comme des identifiants, des informations d'état civil ou des caractéristiques physiques).

Nous disposons, avec le concept de sphère privée, d'un outil aisément manipulable pour traiter de vie privée. La sphère privée encapsule donc toutes les informations, explicitement représentées ou non, qui nous concernent et que nous souhaitons protéger pour quelque motif que ce soit.

Définition 1.2 (Droit à la vie privée)**.** Le droit à la vie privée d'un individu est sa prétention aux caractères personnel, personnalisable, dynamique et contextuel de sa sphère privée ainsi qu'au contrôle de la diffusion, de l'utilisation et de la conservation des informations contenues dans sa sphère privée, quelles que soient la représentation de ces informations et la localisation de cette représentation.

Nous incluons ainsi explicitement dans le droit à la vie privée la notion de propriété des données, fortement liée à celle du contrôle. Nous explicitons ce contrôle (de manière compatible avec la définition de Westin) en l'appliquant à des données distantes, gérées par des tiers ou bien à la représentation desquelles nous n'avons pas accès.

Définition 1.3 (Protection de la vie (ou de la sphère) privée)**.** La protection de la vie privée est l'ensemble des mesures techniques visant à assurer le respect du droit à la vie privée.

1.1.2 Aspects légaux du droit à la sphère privée

Le fondement du droit à la vie privée est assez flou et prend ses sources dans les droits liés à l'individu et à la propriété privée. Deux points de vue particuliers s'opposent sur la nature philosophique de ce droit.

En 1890, dans leur essai *The right of privacy*, Samuel D. Warren et Louis D. Brandeis identifient le droit à la vie privée comme un droit à part, nécessitant une nouvelle protection juridique appropriée [WB90]. Dans cette étude, le droit à la vie privée, s'appliquant aux personnes physiques, leur permet d'interdire la publication d'informations ou de photographies les concernant.

Judith J. Thomson au contraire, dans sa recherche des fondements du droit à la vie privée, considère que ce n'est pas un « nouveau » droit, mais un droit dérivé qui procède du droit à la propriété privée, des droits attachés au corps d'un individu et du droit des contrats [Tho75]. Dans ces deux points de vue distincts nous voyons poindre d'une part la notion de propriété implicitement appliquée à une information, idée effectivement fondamentale dans la notion de droit à la vie privée, et d'autre part le concept essentiel de consentement de la personne physique dont la vie privée est mise en question.

Dans un contexte plus pragmatique et contemporain, le droit de la vie privée a été intégré dans la plupart des cadres légaux nationaux à partir de la fin du XVIIIe siècle, et adapté à l'évolution de l'environnement technique en matière de gestion de l'information au cours des trente dernières années. Les textes concernés présentent une vision de la sphère privée axée sur des impératifs économiques très contextualisés, et remise à jour au fil de l'évolution des technologies et des menaces.

Dans le cadre de notre étude, nous nous intéresserons, à titre d'exemple, au cadre légal existant en France. Depuis 2004, il est globalement accepté que le droit français en matière de protection de la vie privée est compatible avec la législation de la plupart des pays de l'Union européenne (se situant parmi les plus protectrices, derrière l'Espagne notamment), ou avec celle du Canada, par exemple. La législation fédérale américaine, par contre, est très différente et la présente étude ne saurait en constituer une illustration représentative.

En France, depuis la création du code civil en 1803, le droit des individus à la vie privée est affirmé par son article 9 [Ré03], mais de manière générique et appelant une interprétation

appuyée au regard des moyens de traitements mis à disposition par l'essor de l'informatique. En effet, les moyens techniques actuels permettent la mise en œuvre de collectes et de traitements automatisés des données personnelles, contexte qui n'était pas envisageable en 1803. Cette adaptation débute en 1978 par une loi nationale [Ré78] et va se poursuivre dans le cadre de l'Union européenne. L'essentiel du cadre législatif en matière de protection de la vie privée réside dans deux lois nationales [Ré78, Ré04], implémentant deux directives européennes [The95, The02].

Il est à noter que ces textes contiennent tous des limitations au droit à la vie privée, concernant notamment l'instruction des enquêtes judiciaires. Nous ne nous intéresserons pas au cas particulier des restrictions pouvant être apportées par des juges à la protection de la vie privée, pour nous situer plutôt dans le contexte de traitements informatiques ne faisant pas directement suite à un contentieux ou à une mesure de protection de la sécurité nationale.

1.1.2.1 La loi française 78-17 « Informatique et Libertés »

La spécificité des risques induits par les traitements automatisés des informations (et notamment ceux mis en œuvre par les administrations publiques), a motivé la création d'une nouvelle loi, en 1978. La France est alors le premier pays européen à inclure dans son droit national des dispositions spécifiques concernant les traitements informatiques de données personnelles. La loi 78-17 du 6 janvier 1978 [Ré78], « relative à l'informatique, aux fichiers et aux libertés » (dite loi « Informatique et Libertés »)[2] parle de *traitements automatisés* sur des *données nominatives*. Ce texte pose des principes qui sont depuis rentrés dans la culture française, de par les mentions légales apparaissant systématiquement dans les formulaires de collecte de données.

Les données nominatives sont définies comme celles pouvant être rattachées à une personne physique (que nous appellerons ici « l'intéressé »), « de manière directe ou indirecte ». Cette précision permet d'englober les données pouvant être liées à une personne après recoupement avec d'autres informations. Le responsable d'un traitement automatisé ou d'une base de données contenant ce type d'informations ne les possède pas, mais est responsable de leur sécurité.

Le texte introduit l'obligation fondamentale d'**informer** l'intéressé sur divers points concernant le traitement et de recueillir son **consentement** (sauf dans certains cas particuliers prévus par la loi, comme par exemple l'accomplissement d'une mission de service public). L'intéressé jouit également d'un **droit d'accès et de rectification** des données collectées, exerçable auprès du responsable du traitement.

La loi impose une limitation sur la **durée de conservation** des données. Celles-ci devront en effet être détruites une fois qu'elles ne sont plus nécessaires au traitement. Cette disposition, naturellement générale, a appelé par la suite à diverses réglementations spécifiques à certains secteurs d'activité. Les opérateurs de télécommunications, notamment, sont soumis à des directives précises concernant la durée de conservation des informations sur les communications de leurs abonnés.

La **transmission à des tiers** des données collectées et/ou traitées est également réglementée. Elle est par défaut interdite, sauf si l'intéressé a au préalable donné son consentement pour le transfert à un tiers ou à un ensemble de tiers identifiés.

La loi 78-17 crée également la Commission Nationale de l'Informatique et des Libertés (CNIL), garante des droits exprimés dans le texte (notamment le droit d'accès). D'une manière générale, les traitements automatisés portant sur des informations nominatives doivent faire

[2] Dans le cadre de nos travaux, lorsque nous nous référerons à la loi 78-17 [Ré78], nous considérerons qu'il s'agit de la version originalement promulguée, et non pas de la version consolidée (considérablement modifiée par la loi 2004-801 [Ré04]).

l'objet d'une déclaration à cette autorité, qui accorde une autorisation préalable à la mise en œuvre de la collecte et du traitement.

1.1.2.2 La directive européenne 95/46/CE

La directive européenne 95/46/CE « relative à la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données » [The95] est le premier et le principal texte réglementaire de l'Union européenne en matière de protection de la vie privée. Dans ce texte, le vocabulaire change par rapport à la loi 78-17, qui a alors presque vingt ans : on ne parle plus *données nominatives* mais de *données personnelles*. Cette nomenclature prévaudra par la suite dans les textes réglementaires français.

Un des premiers points précisés par cette directive est que la collecte des données doit être *loyale*. Ainsi, il est généralement interdit de collecter des données à l'insu de l'utilisateur ou contre son gré. De plus, il est spécifié que les données collectées doivent être « adéquates, pertinentes et non excessives au regard » de « finalités déterminées, explicites et légitimes ». Ainsi, on introduit un lien fort entre les données collectées et le but du traitement, sa **justification** : il est interdit de collecter des données superflues, non nécessaires, non directement liées au but poursuivi. Cette notion fondamentale est un axe de travail fort pour la protection de la vie privée, elle introduit le concept de *collecte minimale* de données, qui sera par la suite interprété du côté des usagers comme un concept de *transmission minimale* de données, préservant le caractère privé de ses informations. Ces aspects de la protection de la vie privée sont également désignés dans la littérature sous les termes de *principe de proportionnalité* et de *principe de finalité*.

Le texte introduit des dispositions particulières pour interdire (dans la plupart des cas) les traitements portant sur des données très sensibles, comme les convictions religieuses, l'orientation sexuelle ou les informations relatives à la santé.

Au-delà de ces nouvelles notions, la directive entérine les principes de base déjà présents dans la loi française. Notamment, elle renforce la protection vis-à-vis de la diffusion des données, en imposant une nouvelle phase d'information de l'intéressé lorsque ses informations sont utilisées par un tiers. Les transferts de données à destination de pays tiers sont également réglementés et limités aux pays pouvant justifier d'un « niveau de protection adéquat ».

Concernant le contrôle de l'application des diverses dispositions, la directive impose à chaque pays membre la création d'une autorité de contrôle nationale (la CNIL en est le représentant français), et leur regroupement en un « groupe de protection des personnes à l'égard du traitement des données à caractères personnel ». Cette organisation européenne est communément appelée « groupe de l'article 29 » ou plus couramment encore G29. La directive prévoit que les traitements doivent faire l'objet d'une notification préalable à l'autorité de contrôle. Certains types de traitements, sensibles par leur finalité ou leur envergure, font par ailleurs l'objet d'un contrôle préalable systématique.

1.1.2.3 La directive européenne 02/58/CE

La directive européenne 02/58/CE « concernant le traitement des données à caractère personnel et la protection de la vie privée dans le secteur des communications électroniques » [The02] est un exemple d'application sectorielle des principes de la directive de 1995. Elle définit des dispositions techniques de protection de la vie privée spécifiques au secteur des télécommunications. Elle reprend chacun des principes de base, pour les transcrire dans ce cas d'application particulier.

Un des intérêts de cette directive est qu'elle traite des transmissions électroniques non

sollicitées (*spam* ou *junk mail* en anglais), en établissant clairement une priorité de l'*opt-in* dans l'union européenne : si l'intéressé n'a pas explicitement donné son consentement pour être contacté par une société particulière, alors cette société ne peut utiliser ses informations personnelles pour lui faire une offre commerciale. Néanmoins, si l'intéressé a déjà eu recours aux services d'une société, la directive considère qu'il est légitime que celle-ci le recontacte pour lui proposer des services semblables. Les communications non sollicitées sont un cas particulier de l'application des principes de la directive 95/46 (ils dérivent des principes de justification, de consentement et de transmission aux tiers des informations), mais constituent un problème social et technique suffisamment présent pour justifier une législation dédiée et contextualisée. Ce texte réaffirme également le principe de la collecte minimale d'informations introduit par la directive de 1995 (et dérivant du principe de justification).

Cette directive interdit la pratique du *spoofing*, qui consiste à se faire passer pour une autre entité en forgeant une adresse ou un numéro téléphonique, par exemple, dans un but de prospection. L'usurpation d'identité, que ce soit celle d'une personne physique ou morale, est un acte délictueux qui constitue une atteinte particulière aux droits moraux de la personne concernée. Un autre point intéressant est soulevé dans l'article 14-3, qui stipule que « des mesures peuvent être adoptées afin de garantir que les équipements terminaux seront construits de manière compatible avec le droit des utilisateurs de protéger et de contrôler l'utilisation de leurs données à caractère personnel ». La législation européenne garantit donc théoriquement les utilisateurs contre la construction de matériels électroniques ou informatiques trop intrusifs, ou de manière générale présentant un risque pour leur vie privée.

1.1.2.4 La loi française 2004-801

La loi française numéro 2004-801 « relative à la protection des personnes physiques à l'égard des traitements de données à caractère personnel » [Ré04] a pour principal objectif de modifier la loi 78-17 déjà existante, pour la mettre en conformité avec les directives européennes de 1995 et 2002. La France est alors le dernier pays européen à transposer la directive de 1995.

Cette loi entérine le terme d'*informations personnelles* en remplacement des *informations nominatives*. La notion de **justification** de la collecte et du traitement (principe de collecte minimale) est transposée dans le texte français en utilisant les termes de la directive de 1995. Le texte de loi consolidé modifie également les pouvoirs de la CNIL. En particulier, l'étendue de la déclaration préalable est largement diminuée, et la commission dispose d'un rôle de contrôle plus important, assorti d'un pouvoir de sanction.

Ce texte déclencha en 2004 une polémique entre parlementaires et défenseurs des libertés individuelles. Ces derniers considéraient que la nouvelle loi affaiblissait le texte original, en particulier par le statut particulier qu'elle accorde aux traitements mis en œuvre par l'administration publique.

La principale contribution de cette loi aux principes généraux de la protection des données à caractères personnel dans le cadre des traitements automatisés consiste en l'introduction dans la loi française de la notion de justification. Pour le reste, elle reformule la loi existante pour l'harmoniser avec la directive européenne et redéfinit les instances de contrôle.

1.1.2.5 Synthèse

Lois et directives déterminent donc ce qui peut ou ne peut pas être fait de données informatiques susceptibles d'être rattachées à un individu particulier. Les textes réglementaires que nous avons abordés réduisent la sphère privée d'un individu à sa partie observable par un système

électronique ou informatique, aux informations qui peuvent être extraites, représentées et reliées entre elles. Ces textes constituent donc une spécialisation, une projection du principe général exprimé par le Code Civil sur le domaine de l'informatique et des télécommunications.

Il peut être intéressant ici de définir un nouveau terme aux acceptions plus restreintes. Ce que nous nommons « protection des données personnelles » sera donc, dans le cadre de nos travaux, une limitation de la protection de la vie privée à certaines données numériques. Cette acception pourra inclure des informations fournies par l'utilisateur, par son usage de l'outil informatique, ou par ses transactions avec un tiers. Elle excluera toutes les informations implicites, non représentées, qui constituent la sphère privée de l'utilisateur. Ces mêmes informations (par exemple les relations de l'utilisateur avec d'autres personnes, ou ses goûts artistiques) pourront rentrer dans le cadre de la protection des données personnelles dès lors qu'elles seront représentées en tant que telles dans le cadre d'une application informatique.

Définition 1.4 (Protection des données personnelles)**.** La protection des données personnelles consiste en l'ensemble des mesures techniques visant à assurer le respect du droit à la vie privée, limitées aux données de la sphère privée d'un utilisateur explicitement représentées sous forme numérique et mises en jeu dans le cadre d'une application informatique.

Concernant cette protection des données personnelles, l'étude de ces textes fait ressortir six thèmes clés, suivant lesquels s'alignent les diverses réglementations. Ces thèmes sont l'**information** de l'utilisateur, son **consentement**, son **droit d'accès et de modification**, le principe de **justification** de la collecte et du traitement, la **conservation** des données et leur **transmission** à des tiers. Cette partition des réglementations que nous introduisons ici appelle à une étude plus approfondie, visant à spécifier chaque axe de manière claire et distincte. Nous reviendrons dessus plus loin dans ce chapitre.

Avant cela, il faut bien noter qu'en se restreignant ainsi au monde du numérique, on reconnaît que l'on ignore délibérément tous les éléments de la sphère privée qui ne sont pas directement représentables dans un fichier informatique. Cette frontière reste cependant floue et a tendance à se déplacer au fur et à mesure que la technologie permet d'inférer davantage de faits et de règles du comportement d'un individu ou d'un groupe d'individus. Il apparaît donc maintenant comme clair que les données personnelles ne sont qu'une partie de la sphère privée. La protection de ces données sous la forme d'un fichier informatique ne constituent qu'une partie de la protection du droit à la vie privée, même restreint au domaine numérique. Cette conclusion appelle donc à une classification précise des différents domaines qui pourraient constituer la protection de la vie privée et la protection des données personnelles.

1.1.3 Protection de la sphère privée au quotidien

La protection de la vie privée, on l'a vu, n'intervient donc pas exclusivement dans le cadre d'applications informatiques. La notion de sphère privée apparaît en filigrane dans toutes les activités humaines à partir du moment où elles ont une dimension sociale. Le public y est d'ailleurs de plus en plus sensibilisé, même s'il ne semble pas toujours donner sa juste valeur au caractère privé de ses informations personnelles. L'émergence d'Internet est en grande partie responsable de l'évolution des inquiétudes du grand public quant à la protection de sa vie privée en général, et de ses données sensibles en particulier. Les individus mettent potentiellement en danger leur vie privée lorsqu'ils interagissent avec une société pour en obtenir un produit ou un service, lorsqu'ils utilisent des équipements électroniques communicants ou lorsque des informations particulièrement sensibles, comme les données médicales d'un patient, sont confiées

à des traitements automatisés. L'évolution de la structure des applications et des usages individuels, allant vers davantage de collaboration, de délocalisation, de fluidité et de transparence des données, contribue également de manière significative à cette évolution de cette crise de conscience du grand public, cristallisée dans quelques domaines en particulier. Divers domaines de la vie courante peuvent donc servir à l'élaboration de scénarios et de cas d'utilisation dans lesquels se posent des problèmes de protection de la vie privée, ou des données personnelles plus précisément.

1.1.3.1 Domaine de la santé

Les données de nature médicale appartenant aux utilisateurs (aux patients) sont perçues par ces derniers comme particulièrement sensibles, car touchant directement à leur corps et à leur intégrité physique, au cœur de tout ce qui peut être considéré comme privé. Malgré cela, les utilisateurs sont de plus en plus amenés à confier ces données à l'outil informatique et à en perdre partiellement le contrôle. La mise en œuvre du dossier médical informatisé (et distribué) est le premier et le plus évident de ces contextes d'informatisation des données médicales. Suivant les pays, cette mise en œuvre prend des formes et des extensions diverses, mais les questions de fond qui se posent (et qui conditionnent le choix des procédés techniques) sont les mêmes : en dépit du fait que le patient est propriétaire de ses données, est-ce que le corps médical n'a pas son mot à dire sur la manière dont elles doivent être traitées ? Qui doit avoir accès à ces données, où, comment et combien de temps seront-elles stockées ? Les outils techniques pour le maintien à domicile des personnes agées, handicapées, ou partiellement autonomes d'une manière générale, soulèvent également des questions spécifiques. Ce type d'application mêle en effet les problématiques de protection des données personnelles liées aux informations médicales (qui sont par exemple susceptibles d'être transmises automatiquement en cas d'urgence) à des problèmes de protection de la vie privée que l'on va retrouver dans le domaine de l'intelligence ambiante.

1.1.3.2 Domaine de l'intelligence ambiante

L'intelligence ambiante[3] met en œuvre de multiples équipements électroniques et terminaux informatiques, typiquement mobiles et de petite taille, dans un environnement au sein duquel ils communiquent, collaborent et coopèrent de manière quasiment transparente pour l'utilisateur humain. Les équipements de l'environnement disposent de capacités d'adaptation, de raisonnement, de mémorisation limitées uniquement par leurs faibles ressources physiques. Les données échangées entre les diverses entités sont donc constamment soumises à des traitements essentiellement distribués. Les objectifs de ces nombreuses interactions peuvent être multiples et sont souvent organisés autour de la personnalisation et de l'augmentation de l'expérience de l'utilisateur, dont l'utilisation du système (bâtiment ou environnement de travail par exemple) doit être facilitée. Les objets de l'environnement interagissent constamment entre eux, échangent des informations et produisent des traces qui peuvent être agrégées en profils comportementaux. Ces données sont susceptibles d'être partagées de manière intensive, dans le but d'améliorer la

[3] Carole Adam *et al* caractérisent ainsi l'intelligence ambiante [AEG+06] :

> « L'Intelligence Ambiante désigne un ensemble d'interfaces intelligentes supportées par des technologies de réseau et de traitement des données enfouies dans les objets du quotidien [...]. Son but est d'être attentive aux caractéristiques spécifiques de chacun, de s'adapter aux besoins des utilisateurs, d'être capable de répondre intelligemment à des indications parlées ou gestuelles, et même d'engager un dialogue. Elle doit être non intrusive et le plus souvent invisible pour l'utilisateur au quotidien. »

fluidité de l'expérience de l'utilisateur au sein de l'environnement intelligent. L'utilisation des composants radiofréquence RFID, en particulier, a été particulièrement étudiée pour sa capacité à diffuser et à créer des informations potentiellement personnelles et sensibles [JL02, SB05]. Ici, le compromis entre aisance d'utilisation et protection de la vie privée joue à plein, et les informations de l'utilisateur se retrouvent dans une situation particulièrement risquée. Typiquement, les scénarios d'intelligence ambiante mettent en œuvre des traitements relevant de la protection de la vie privée, mais pas forcément de la protection des données personnelles. En effet, les informations échangées sont souvent implicites par nature pour l'utilisateur. N'ayant pas accès à leur représentation, il lui est plus délicat de s'assurer de leur protection efficace.

1.1.3.3 Domaine du commerce électronique

Les scénarios de commerce électronique peuvent être très variés, allant d'une simple transaction à une collaboration étroite et complexe entre agents commerciaux dépendant d'entités différentes, ou à des techniques de raisonnement élaboré sur les habitudes du client. Dans le cadre d'une transaction commerciale électronique (le plus souvent sur Internet via un site marchand), la communication d'informations personnelles est essentielle. Les informations apparaissant comme sensibles de la manière la plus immédiate et évidente sont les données bancaires de l'utilisateur (qui ici devient un client), nécessaires au paiement de la transaction. Les utilisateurs sont globalement sensibilisés aux risques afférents à la communication d'un numéro de carte bancaire. En France particulièrement, la culture de la carte à puce a amené le public à être encore plus méfiant que dans les pays anglo-saxons, où la communication d'un numéro complet par téléphone est une pratique courante. Pour autant, les informations de nature bancaire ne sont pas les seules à être divulguées lors d'une transaction commerciale. La nature des objets ou des services achetés, agrégés en historique, constitue en elle-même une information révélatrice du mode de vie de l'utilisateur et d'une valeur commerciale indiscutable pour une entreprise. Les informations nécessaires à la livraison devraient également être considérées avec attention, ainsi que les identifiants de la connexion et du terminal de l'utilisateur. Les transferts d'informations personnelles sont une nécessité absolue pour les applications commerciales, et l'utilisation qui en est faite est rarement maîtrisée. Les possibilités de négociation avec un service informatique étant quasiment nulles, l'utilisateur peut être contraint par la nécessité de confier plus d'informations qu'il ne le devrait ou le voudrait. Il est donc alors de fait dessaisi d'une partie de son contrôle sur ses données personnelles.

1.2 Les dimensions techniques de protection de la vie privée sur Internet

Yves Deswarte et Carlos Aguilar Melchor proposent en 2006 une classification en cinq domaines des techniques de protection de la vie privée sur Internet [DA06b]. Nous reprenons ici cette classification, orientée par des considérations technologiques. Les exemples sont principalement ceux fournis par Deswarte et Aguilar Melchor.

1.2.1 Gestion des identités

La gestion des identités recouvre les technologies qui permettent à l'utilisateur de masquer sa véritable identité lorsque la connaissance de celle-ci n'est pas nécessaire au traitement[4]. L'ob-

[4] Cette technique se distingue fondamentalement du *spoofing* en ceci que l'utilisateur n'usurpe pas l'identité d'un tiers, il se contente de ne pas révéler la sienne.

jectif ici recherché est d'empêcher la corrélation des différentes activités de l'utilisateur, qui ne souhaite pas forcément que plusieurs utilisations successives d'une même application puissent être comparées, ou encore que différents prestataires de services s'entendent pour partager des informations sur son profil d'utilisation des ressources.

La principale solution technique à ce problème est le concept d'**identité virtuelle**. L'utilisateur en possède plusieurs, correspondant à différentes applications, différents profils comportementaux ou différents niveaux de sécurité. Ces profils doivent être suffisamment anonymes pour qu'il soit difficile d'établir des corrélations entre eux. Typiquement, chacun d'eux est identifié par une chaîne de caractère arbitraire (le **pseudonyme**) qu'il n'est possible de relier à l'utilisateur que sous certaines conditions. La plupart du temps, seul l'utilisateur lui-même (ainsi éventuellement que des tiers de confiance) en est capable.

Le concept d'identité virtuelle est notamment repris dans le protocole IDsec, développé par l'IETF [Int], ou dans les travaux du projet européen PRIME [Eur08]. C'est également un concept clé dans les architectures fondées sur les *Trusted Computing Platforms* (TCP) [Tru03a]. Nous aurons l'occasion de revenir plus en détail sur ces applications.

1.2.2 Communications IP anonymes

Plusieurs technologies ont été développées pour permettre la non-traçabilité des communications sur un réseau IP, que ce soit pour des applications pair-à-pair fortement distribuées et à accès essentiellement anonymes, ou bien pour des applications client-serveur avec une notion de session très forte.

Les **routeurs MIX** (proposés par David Chaum), par exemple, permettent de cacher le lien existant entre les messages entrants et sortants, par l'utilisation de bourrage, de chiffrement aléatoire et de brouillage statistique (émission de messages fictifs). Ces techniques permettent d'éviter qu'un observateur extérieur puisse relier un message entrant à un message sortant en analysant son contenu ou en étudiant les séquences temporelles d'entrées et sorties. La mise en réseau de routeurs MIX (de manière à limiter l'impact de la prise de contrôle de l'un des routeurs par un attaquant et à rendre l'ensemble robuste aux collusions de routeurs) est le principe de base de certains réseaux « anonymes » destinés à des applications pair à pair (comme Tarzan [FM02]) ou de courrier électronique (comme Mixminion [DDM03]), par exemple.

Les **Réseaux DC** (pour *Dining Cryptographers*) sont pour leur part des réseaux dans lesquels tous les messages sont émis de manière anonyme (la protection de l'émetteur étant garanti par une émission de données, significatives ou non, en continu) et diffusés à tous les membres (ce qui garantit l'anonymat du récipiendaire, qui est cependant le seul à pouvoir déchiffrer le message si son contenu est protégé). Ce type de technique présente toutefois l'inconvénient de gaspiller les ressources réseau, ce qui constitue un frein à son passage à l'échelle.

Les **foules** et les **hordes** sont d'autres exemples de technologies visant à assurer l'anonymat d'utilisateurs navigant sur le web, par le biais d'un re-routage aléatoire des requêtes entre les membres de la communauté d'utilisateurs. L'anonymat des requêtes et la possibilité que l'utilisateur reçoive la réponse associée sont garantis par le partage d'une clé de chiffrement secrète par chaque paire d'utilisateurs.

1.2.3 Accès anonymes aux services

Ce domaine concerne l'anonymisation des messages de l'utilisateur au niveau du protocole applicatif. En effet, dans le cas général la communication entre utilisateur et service comporte beaucoup d'informations identifiantes, et il est parfois possible de limiter la portée de

ces informations sans détériorer la qualité du service. Une anonymisation efficace semble donc nécessairement passer par des relais mandataires (*proxies*) applicatifs chargés d'obfusquer les informations sensibles, qui sont souvent spécifiques au service ou à l'application.

1.2.4 Autorisation préservant la vie privée

Il existe des techniques permettant de séparer la phase d'authentification de la phase d'autorisation d'accès, afin que l'utilisateur puisse accéder de manière anonyme au service. Lors de la phase d'authentification, l'utilisateur prouve son identité auprès d'un tiers de confiance (via l'utilisation d'un login et d'un mot de passe, d'un certificat cryptographique ou d'un *challenge* quelconque). Le tiers de confiance peut également, en fonction du type de requête et des exigences des fournisseurs de service, vérifier que l'utilisateur respecte certaines caractéristiques. Le tiers émet ensuite une accréditation (*credential*) affirmant que l'identité de l'utilisateur, ainsi éventuellement que ses autres caractéristiques, ont été vérifiées. L'utilisateur se servira ensuite de cette accréditation (qui ne mentionne pas son identité, mais dont l'authenticité peut être vérifiée auprès du tiers de confiance) pour accéder à des services ou à des ressources.

Ce type de protocole, de la même inspiration que le système de jetons d'accès de Kerberos [Mas], permet ainsi d'une part à l'utilisateur d'accéder à un service sans dévoiler son identité, et d'autre part au gestionnaire du service de se convaincre que l'intermédiaire de confiance a bien vérifié les caractéristiques requises concernant tous les utilisateurs accrédités.

Le principe de l'autorisation préservant la vie privée (utilisant des accréditations anonymes) a été mis en œuvre de diverses manières. Deswarte et Aguilar Melchor mentionnent notamment les applications de *e-Cash* (fondées sur les travaux de David Chaum), permettant d'effectuer un paiement électronique sans dévoiler son identité, l'architecture à base de clés asymétriques SPKI/SDSI (développée par l'IETF) et l'application IDEMIX (*Identity Mixer*) développée par IBM.

On peut également mentionner OpenID [Ope], un protocole de plus en plus courant sur le web, issu de l'initiative *Identity 2.0* [Ide]. Le système OpenID (intégré à l'interface d'authentification de nombreux sites internet et mis en place côté client par le biais de plugiciels comme Sxipper [Sxi]) permet aux utilisateurs d'enregistrer leur identité sur un serveur OpenID de leur choix. Pour accéder à un site web nécessitant une autorisation, l'utilisateur fournit alors une URI, possédée par le serveur OpenID, qui permet au gestionnaire du site web d'obtenir une accréditation d'identité de la part du serveur OpenID, sans avoir à prendre connaissance ni à enregistrer un login ou un mot de passe. L'utilisateur peut en outre se servir d'un ou de plusieurs serveurs OpenID pour gérer plusieurs identités virtuelles, appelées *persona*.

1.2.5 Gestion des données personnelles

La gestion des données personnelles couvre les dispositifs de protection des données qui sont susceptibles d'identifier un agent utilisateur humain, artificiel ou institutionnel. Elle concerne notamment la protection des informations de profil, de personnalisation, les préférences utilisateur transmises à une application, les identifiants et les requêtes faites au nom de l'utilisateur. Deswarte et Aguilar Melchor mettent en valeur un point de droit qui apparaît dans les législations française et européenne : les informations personnelles se rapportant à un individu sont la propriété de cet individu, quel que soit le système sur lequel elles sont stockées. Le propriétaire dudit système n'hérite pas de la propriété des données, mais uniquement de la responsabilité de leur protection.

Les auteurs pointent deux sous-axes : la **minimisation des données personnelles**, qui correspond au principe introduit en 1995 dans la législation européenne, et l'**auto-détermination des données**, qui traduit un des problèmes les plus critiques qui se posent à nous, à savoir assurer la protection des données personnelles indépendamment des systèmes et des tiers auxquels elles sont confiées. Deswarte et Aguilar Melchor estiment que l'utilisation d'architectures à base de composants logiciels et matériels « dignes de confiance » (les *Trusted Computing Platforms* [Tru03a], fondées sur des composants particuliers appelés *Trusted Computing Modules* ou TPM [Tru03b]) peut être un moyen efficace pour assurer cette dernière propriété. Néanmoins, ils notent également que ces mêmes technologies peuvent aussi être utilisées pour poursuivre des objectifs allant à l'encontre des intérêts de l'utilisateur. Par la suite, nous nous pencherons plus avant sur ces technologies afin de caractériser les utilisations qui peuvent en être faites.

Parmi les cinq axes technologiques identifiés par Deswarte et Aguilar Melchor, ce dernier est manifestement celui qui correspond le mieux à la définition que nous avons donnée de la protection des données personnelles. Nous considérons que c'est principalement sur cette partie de la protection de la vie privée que portent nos travaux.

1.3 La protection des données personnelles

Maintenant que nous disposons du vocabulaire adéquat ainsi que d'une vue d'ensemble sur les familles de dispositifs techniques pour la protection de la vie privée, nous proposons de nous concentrer sur un domaine en particulier, à savoir la protection des données personnelles [PDC06]. Nous essaierons de caractériser comment cet aspect particulier est traité dans les documents réglementaires, nous analyserons quelques-unes des nombreuses propositions techniques s'y rapportant et le mettrons en lien avec les trois axes applicatifs déjà identifiés pour la protection de la sphère privée en général.

1.3.1 Les six axes de la protection des données personnelles

L'analyse des textes légaux et réglementaires (ainsi que des recommandations que l'on peut couramment observer dans les chartes d'utilisation de systèmes informatiques) nous a permis de classer en six axes les éléments constitutifs des réglementations en matière de protection des données personnelles. Cette classification vient compléter, de manière transversale, les « critères communs » fixés par la norme ISO/IEC 15408-2, publiée en 1999 [ISO99], décomposant la protection des données personnelles en quatre critères techniques évaluables que doit respecter un système :

- La possibilité pour l'utilisateur d'agir de manière anonyme, de manière qu'aucun autre utilisateur ne puisse l'identifier ;
- La possibilité d'agir sous un pseudonyme, interdisant l'identification directe par les autres utilisateurs mais permettant tout de même de relier l'utilisateur à ses actions ;
- L'impossibilité pour les autres utilisateurs d'établir des corrélations entre les différentes activités de l'utilisateur ;
- La non-observabilité, interdisant aux autres utilisateurs de pouvoir décider si une action est en cours.

Ces « critères » expriment les conditions que doit respecter un système pour garantir la protection de la vie privée de ses utilisateurs, sans toutefois caractériser cette protection elle-même.

La classification que nous proposons [PDC06] est d'une autre nature. Elle identifie six axes, six dimensions suivant lesquelles une autorité (comme un système législatif) peut émettre des

exigences sur la protection des données personnelles. À la différence de l'ISO/IEC, qui a par essence un rôle normatif, nous ne préjugeons pas ici de l'étendue de ces exigences, mais uniquement de leur nature, notre but étant de pouvoir décrire, caractériser et comparer des réglementations. Néanmoins, à titre d'illustration il nous paraît commode de nous référer aux exigences légales françaises et européennes, que nous avons déjà mentionnées.

1.3.1.1 Information

Le premier axe réglementaire concerne l'**information** de l'utilisateur. Dans tous les textes étudiés [Ré78, art. 27] [The95, sect. IV], [The02, préambule, alinéa 23] [Ré04, art. 32], on impose au responsable d'un traitement informatique portant sur des données personnelles d'informer les propriétaires de ces données d'un certain nombre de caractéristiques de ce traitement. Typiquement, le type d'information fourni est défini par les cinq autres axes réglementaires.

Une réglementation peut par exemple imposer que lorsqu'un traitement mettant en jeu des données personnelles a lieu, les propriétaires de ces données soient informés de la nature du traitement.

1.3.1.2 Consentement

Le deuxième axe réglementaire concerne l'accord, exprimé par le propriétaire des données, à la collecte et au traitement de ses informations personnelles. Dans les textes étudiés, ce **consentement** est qualifié suivant les cas d'« explicite » ou « indubitable ». Dans la législation européenne, le consentement *opt-in* est de rigueur, c'est-à-dire que par défaut l'on considère que l'utilisateur n'autorise pas le traitement.

Bien évidemment, les textes légaux prévoient (pour le consentement comme pour certains autres axes) des exceptions dans les cas mettant en jeu la santé publique, la sécurité nationale, l'instruction des affaires judiciaires...

1.3.1.3 Modification

Le troisième axe, que nous intitulons « **modification** », regroupe plusieurs concepts liés. C'est à cet axe que nous rattachons le droit d'accès et de modification initialement introduit par la loi 78-17 [Ré78, chap. 5] ainsi que toute réglementation visant à donner à l'utilisateur les moyens de demander une mise à jour ou une suppression des informations personnelles collectées. Les réglementations en la matière sont généralement de deux ordres : la nécessité pour l'utilisateur de disposer d'un moyen d'effectuer de telles requêtes auprès du responsable du traitement, et l'obligation (généralement conditionnelle) pour ce dernier d'y accéder.

1.3.1.4 Justification

Ce quatrième axe est de loin le plus complexe des six. Il a pour vocation de regrouper les réglementations traitant de la question de fond : « est-il *justifié* de collecter telle donnée et de l'utiliser dans le cadre de tel traitement ? ». Cet axe se réfère donc aux notions de finalité, de proportionnalité et de minimisation des données, principalement abordés par la directive européenne de 1995 [The95]. Les réglementations s'y rapportant auront donc notamment pour objet la restriction du type d'information collectée en fonction du traitement déclaré et l'encadrement de la réutilisation des informations collectées pour d'autre traitements.

Les règles édictées en la matière sont souvent très dépendantes du domaine d'application, puisqu'elles ont pour but de signifier quel type d'information peut être utilisé pour quel type de traitement.

1.3.1.5 Conservation

Le cinquième axe traite de la **conservation** des données personnelles après leur collecte. Les textes posent des limites (en général supérieures, parfois inférieures) aux durées de conservation en fonction du contexte : statut des acteurs, nature des données et des traitements. Dans le cas général, la législation européenne prévoit que les données personnelles ne doivent pas être conservées « plus longtemps que nécessaire » (sic), les modalités de cette règle générique étant précisées dans des cas particuliers ou laissées aux soins d'autres autorités de réglementation (contrats, réglementations sectorielles, textes de loi plus spécifiques...).

1.3.1.6 Transmission

Le sixième axe concerne la **transmission** à des tiers de données déjà collectées. Les réglementations en la matière autorisent, interdisent ou limitent cette transmission (qui peut éventuellement prendre la forme d'une transaction commerciale). La règle générale dans l'Union européenne veut que de telles transmissions soient interdites, à moins que l'utilisateur n'y ait expressément consenti. Encore une fois, des exceptions sont ménagées pour permettre notamment la transmission de données personnelles aux pouvoirs exécutif et judiciaire lorsque cela est nécessaire[5].

1.3.2 Protection locale et protection étendue

Les six axes réglementaires que nous avons identifiés font référence à un certain nombre de mesures de protection des données personnelles. Suivant les axes, ces mesures de protection doivent être soit mises en place en un point bien particulier de l'application (typiquement, le terminal de l'utilisateur dont les données personnelles sont en jeu), soit déployées sur l'ensemble de l'application répartie. Pour distinguer ces deux concepts au sein de la protection des données personnelles, nous distinguerons la **protection locale** de la **protection étendue**.

1.3.2.1 Protection locale des données personnelles

Définition 1.5 (Protection locale des données personnelles)**.** La protection locale des données personnelles est le sous-ensemble des mesures de protection des données personnelles ne nécessitant que la vérification de propriétés sur le terminal de l'utilisateur propriétaire des données.

[5] On pourrait considérer que l'axe « transmission » est en fait une partie intégrante de l'axe « justification », en ceci que la transmission des données constitue en général une utilisation de celles-ci pour une autre finalité que celle initialement prévue, déclarée et consentie. Cependant, c'est une action très particulière, à distinguer des autres traitements, car elle met en quelque sorte le responsable du traitement dans une position de mandataire de l'utilisateur : il doit prendre la décision de divulguer ou pas l'information en question à un tiers, comme l'utilisateur a pris cette décision envers lui. La transmission de données à un tiers peut constituer une brèche significative dans la protection des données, en fonction de la confiance accordée à ce dernier. Elle peut donner lieu, une fois les données divulguées, à des actions violant des réglementations relatives à tous les autres axes. C'est pour cette raison que nous considérons qu'il faut traiter cette dimension dans un axe réglementaire séparé. Cette position est d'ailleurs confirmée par le soin apporté par les législateurs français et européens pour distinguer les réglementations traitant de la transmission des données.

La protection locale telle que nous l'entendons contient donc l'ensemble des vérifications qui peuvent être faites directement par l'utilisateur. Cela concerne donc en particulier les deux premiers axes de notre classification. En effet, l'utilisateur peut aisément déterminer par lui-même, localement, de quoi il a été **informé** et des **consentements** qu'il a exprimés ou non. Le troisième axe (**modification**) est également partiellement concerné, en ceci que l'utilisateur sait s'il dispose ou non d'un moyen de contacter le responsable d'un traitement en vue d'exercer un éventuel droit d'accès, de modification ou de suppression.

Le caractère purement local de ces trois axes est toutefois discutable (la répartition entre axes locaux et étendus ne constituant qu'une orientation générale). En effet, le responsable d'un traitement peut être le seul à savoir qu'il n'a pas fourni telle ou telle information à propos du traitement ; ce traitement peut avoir lieu à l'insu du propriétaire des données, qui ignore alors qu'une éventuelle réglementation réclamant son consentement a été violée ; enfin, lorsqu'une requête de modification est formulée par un utilisateur, le respect de cette requête ne peut être a priori contrôlé depuis la plate-forme de ce dernier.

De par sa nature, la protection locale des données personnelles est relativement facile à assurer de manière simple par des méthodes de contrôle d'accès classique, assortis de conditions d'application.

1.3.2.2 Protection étendue des données personnelles

Définition 1.6 (Protection étendue des données personnelles)**.** La protection étendue des données personnelles est le sous-ensemble des mesures de protection des données personnelles nécessitant la vérification de propriétés sur des plates-formes non contrôlées par l'utilisateur propriétaire des données.

La protection étendue traite donc des propriétés qui ne sont plus vérifiables localement. Les trois derniers axes (**justification**, **conservation** et **transmission**) de notre classification relèvent typiquement de la protection étendue : l'utilisation réelle qui sera faite des données fournies, la durée pendant laquelle elles vont être conservées et leur éventuelle transmission à des tiers ne sont a priori pas vérifiables directement depuis la plate-forme de l'utilisateur. La protection étendue concerne les risques qui surviennent une fois que les données ont été confiées à quelqu'un d'autre que leur propriétaire.

Le troisième axe (**modification**), s'il comporte, comme nous l'avons vu, une composante locale, relève également la protection étendue. Localement, l'utilisateur peut décider seul si oui ou non il dispose des moyens de demander une mise à jour ou une suppression des données, mais la nature distante de ces traitements nécessite des moyens de contrôle et de vérification particuliers.

Un ensemble de méthodes efficace de protection étendue permettrait à un utilisateur d'obtenir des garanties fortes sur les propriétés distantes de ses données : comment elles sont utilisées, combien de temps elles sont conservées et avec qui elles sont partagées. Néanmoins, ceci constitue le principal verrou du domaine de la protection des données personnelles, la vérification des propriétés distantes semblant indécidable dans le cas général[6].

[6] Pour illustrer cette idée, il suffit de considérer qu'Alice donne son numéro de téléphone à Bob, puis qu'elle lui demande de l'oublier. Comment Bob pourrait-il la convaincre qu'il a accédé à sa demande ? Si les méthodes formelles d'authentification [BAN89], par exemple, permettent de prouver que l'on connaît une information, il paraît intuitivement impossible de prouver que l'on ignore une information. Bob peut en effet toujours prétendre et simuler avoir détruit ou oublié le numéro de téléphone : tant qu'il ne commet pas une erreur (provoquant une fuite d'information), il reste crédible. Cette explication fondée sur le sens commun ne remplace évidemment pas une démonstration formelle, qui à notre connaissance reste à établir.

1.3.2.3 Nécessité d'une protection étendue

Un grand nombre d'applications et de propositions techniques qui se réclament de la protection de la vie privée, ou plus particulièrement de la protection des données personnelles, se concentrent sur le problème du contrôle d'accès aux données. Outre le fait que le contrôle d'accès n'est pas le seul problème à traiter, il est généralement considéré avec un point de vue strictement local. Il existe en effet de nombreux systèmes fondés sur des règles de contrôle d'accès complexes, utilisant par exemple la notion de rôle (du type de RBAC [FK92]) ou d'organisation (comme OrBAC [AEB+03]). Les caractéristiques d'un demandeur y sont examinées, d'une manière ou d'une autre, avant de décider si les données peuvent lui être confiées. Cependant, une fois que le demandeur accède aux données, le rôle du contrôle d'accès, essentiellement local, est terminé[7]. À partir de ce point, le niveau de protection des données personnelles est donc limité par la confiance que l'utilisateur peut avoir dans le demandeur, confiance qui peut être impunément trahie puisqu'en l'absence de protection étendue, les propriétés distantes (relatives principalement aux trois derniers axes réglementaires) ne sont pas vérifiables par l'utilisateur. C'est ainsi que nombre de solutions techniques n'abordent pas le problème majeur (mais délicat) de la protection des données personnelles, à savoir la proposition de solutions techniques pour assurer la protection étendue.

Il nous faudra donc, dans notre analyse de l'état de l'art des solutions techniques de protection des données personnelles, prendre en compte la gestion des aspects de protection étendue aussi bien que des aspects purement locaux.

1.3.3 Problématiques spécifiques aux domaines d'intérêt

Pour ilustrer cette classification, nous reprenons ici les trois domaines que nous avons introduits plus haut, à savoir le domaine de la santé, l'informatique ambiante et le commerce électronique. Nous proposons ici un aperçu du type de problèmes pouvant être posés par chacun de ces domaines d'application.

1.3.3.1 Problématiques dans le domaine de la santé

En informatique médicale, on se repose souvent sur un **consentement** a priori ou par défaut du patient, en particulier lorsque celui-ci est dans l'incapacité de s'exprimer. Un accès rapide et complet aux données peut être une nécessité vitale pour le patient, mais la divulgation (**transmission**) de ces mêmes données à des personnes dont l'accès n'est pas **justifié** par des raisons médicales peut aller à l'encontre des intérêts du patient. C'est par exemple le cas lorsqu'un établissement bancaire ou d'assurances s'enquiert du dossier médical d'un de ses clients. D'autre part, tout membre du personnel médical n'est pas forcément autorisé à accéder à tout type d'information médicale à n'importe quel moment. Les politiques d'accès dépendent a priori de l'état du patient, des actes médicaux prévus, du statut des personnes requérant les données et de leur implication dans les actes médicaux en question. La mise à jour des données (**modification**) et leur **conservation** peuvent être vitales pour le patient, mais elles peuvent

[7] Dans des systèmes comme RBAC ou OrBAC, il peut exister une autorité régulant les activités des entités du système, à la manière d'un administrateur pour un système informatique, par exemple. L'introduction d'une telle autorité réduit donc artificiellement les propriétés distantes (pour les entités du système, comme l'utilisateur propriétaire des données), relevant de la protection étendue, à des propriétés locales (pour l'autorité de contrôle). Néanmoins, dans un « cas réel » comme par exemple dans le cadre des domaines d'applications que nous avions identifiés comme sensibles pour la protection de la vie privée, l'existence d'une telle autorité omnisciente est généralement impossible. Nous serons amenés à discuter de cette question plus en détail par la suite.

dépendre de procédures dont il n'a qu'une connaissance partielle. L'utilisateur, devenu un patient, est ici plongé dans un univers complexe dont il ne maîtrise en général pas les subtilités, notamment lorsqu'il n'est pas suffisamment **informé** (ou qu'il n'est pas capable de comprendre l'information). Pour cette raison et pour sa propre sécurité, les responsables du personnel médical auront donc tendance à éviter à l'utilisateur de prendre, sur la manipulation de ses données, des décisions qui pourraient s'avérer dangereuses pour lui. L'utilisateur perd ici en contrôle et s'en remet à des tiers pour gérer ses données personnelles.

1.3.3.2 Problématiques dans le domaine de l'intelligence ambiante

En informatique ambiante et plus particulièrement dans le contexte de l'intelligence ambiante, l'utilisateur est en interaction constante avec les objets de son environnement. Dans ce contexte, l'utilisateur est amené à dévoiler des informations de diverses natures : des données explicitement exprimées concernant son souhait d'utilisation du système (comme par exemple ses préférences sur l'ambiance lumineuse d'un environnement ou sur le paramétrage d'un outil) mais aussi des informations non directement formulées sur son utilisation effective du système (comme la fréquence et la nature de ses déplacements dans un bâtiment). À cause des ressources limitées des objets de l'environnement, la **conservation** des données et leur **modification** sont sans doute des préoccupations mineures en intelligence ambiante. En revanche, la fréquence des interactions fait de la **transmission** un axe de travail important pour la protection des données personnelles. L'**information** de l'utilisateur et son **consentement** sont le plus souvent sous-entendus, et le problème de la **justification** des traitements se pose à divers degrés en fonction de la nature plus ou moins sensible des informations transmises. Bien que les environnement d'informatique ambiante soient en général développés pour assister l'utilisateur, ce dernier est mis dans une situation où il ne contrôle que partiellement l'utilisation de ses données personnelles, et peut ne même pas avoir conscience de leur existence.

1.3.3.3 Problématiques dans le domaine du commerce électronique

Dans le cadre du commerce électronique, que l'utilisateur cherche à profiter d'un service numérique purement immatériel (comme l'accès à un contenu) ou à acquérir un bien de consommation, il devra fournir un certain nombre de données au fournisseur du service. En effet, ce dernier a bien souvent l'obligation légale de **conserver** et de tenir à jour (**modification**) pendant un temps déterminé les informations permettant d'identifier ses clients, pour des raisons de traçabilité et de comptabilité. L'utilisateur est donc susceptible de dévoiler des informations sur son identité, sa localisation géographique, ses données bancaires, ses préférences relatives au service ou au produit commandé, son profil d'utilisation du service commercial, et a priori toute autre information exigée par le fournisseur de service. Les utilisateurs peuvent avoir des avis différents quant à la pertinence (**justification**) de ces informations en regard du service ou du produit demandé. L'utilisateur dispose en général d'une déclaration du service commercial sur les **informations** relatives au traitement, mais bien souvent elles ne sont pas prises en considération (par manque de temps, d'intérêt ou par défaut de lisibilité) et le **consentement** de l'utilisateur est conditionné uniquement par son besoin d'accéder au service. Enfin, le service pouvant requérir la collaboration de plusieurs intervenants, certaines données doivent souvent pouvoir être **transmises**, ne serait-ce qu'à une société de livraison ou à une banque.

1.4 Technologies de protection des données personnelles

Des propositions de plus en plus nombreuses sont présentées comme améliorant la protection de la vie privée. On trouve dans cette approche des travaux « atomiques », traitant un aspect du problème en fournissant un outil informatique ou méthodologique (à l'image du standard *Platform for Privacy Preferences* du W3C [Wor06]), ou encore des propositions composites se reposant sur de telles briques fonctionnelles. Cette dernière catégorie recouvre des travaux plus ou moins ambitieux, allant de la spécification de profil utilisateur sécurisé (comme dans la proposition de Stéphanie Riché et Gavin Brebner [RB03]) aux infrastructures intégrées portées par les projets européens PRIME [Eur08] ou PISA [Bor00], par exemple.

Nous nous proposons d'analyser les quelques outils et principes que l'on retrouve couramment dans les propositions existantes et qui diffèrent du simple contrôle d'accès, non spécifique à la protection des données personnelles.

1.4.1 *Platform for Privacy Preferences*

Le standard P3P du *World Wide Web Consortium* [Wor06] est un outil désormais incontournable de la communication des sites web sur leur politique de protection des données personnelles. P3P est une spécification de documents XML décrivant les politiques de traitement des données personnelles déclarées par un site web. Ces documents sont conçus pour être accessibles par un navigateur à partir de la page d'accueil du site.

L'objectif de ce projet est de rationnaliser la manière dont les sites web communiquent sur leurs traitements. Les données présentes dans un document P3P couvrent les aspects suivants :

- L'identité de l'entité collectant les données ;
- La nature des données collectées ;
- La destination (ou justification) de la collecte de données ;
- L'identification des données pouvant être partagées avec des tiers ;
- L'identification de ces tiers ;
- La possibilité offerte ou non aux utilisateurs de modifier la manière dont leurs données sont traitées ;
- Les méthodes de résolution des conflits éventuels (et le ressort juridique compétent) ;
- La durée de rétention de chacune des informations collectées ;
- Un lien vers une version de la politique lisible par un humain.

Il faut bien comprendre ici, et les documents du W3C le soulignent, que P3P n'impose aucune politique minimale, il ne fait que fournir le moyen de l'exprimer. De plus, P3P ne permet pas de vérifier que la politique est effectivement appliquée par le site web en question. P3P a pour seul objectif (comme précisé dans les spécifications) de résoudre le problème de l'information de l'utilisateur, à l'exclusion des cinq autres axes de la protection des données personnelles.

On peut toutefois noter que les diverses extensions à P3P permettent également de traiter partiellement le problème du **consentement** de l'utilisateur. En effet, le langage APPEL [Wor02] permet de spécifier des préférences du côté de l'utilisateur. Ainsi, le navigateur est capable de détecter automatiquement (via des moteurs fournis par le W3C) si une politique P3P est conforme aux préférences APPEL, le fait étant alors considéré comme un consentement a priori de l'utilisateur. Ce système est par exemple utilisé dans le cas simple de la décision d'acceptation d'un *cookie* par un navigateur.

Les concepteurs de systèmes de protection des données personnelles ont tout intérêt à s'appuyer sur le standard P3P, ou en tout cas à demeurer interopérable avec lui. En effet, il permet de résoudre de manière simple le problème de l'information de l'utilisateur, en étant capable

de décrire les divers aspects relatifs au traitement des données. Si les listes de choix prédéfinies pour la spécification du type de traitement, de leur justification ou du type de données restent limitées, elles sont extensibles par le biais de schémas XML personnalisés. De nombreux schémas spécifiques sont déjà disponible sur le web, comme par exemple celui traduisant le langage ECML, pour le traitement des données relatives au commerce électronique [Wor99]. De plus, P3P est déjà largement utilisé par les sites web pour leur communication, et de nombreux outils sont capables de manipuler le formalisme d'une manière ou d'une autre. Toutes ces raisons poussent à favoriser au maximum l'intéropérabilité avec P3P, préférentiellement à d'autres langages de politiques comme SPARCLE [KS02] ou *XACML Privacy Policies* [Org05], moins génériques et moins répandus.

Il faut toutefois rester conscients des limitations de P3P. Tout d'abord, la restriction à un rôle d'**information** (et éventuellement de **consentement**). Enfin et surtout, P3P exprime la politique d'un site web indépendamment de tous les types de réglementations que nous avons pu identifier. Un utilisateur n'a alors aucun moyen de savoir si ces politiques respectent telle loi ou telle directive. Il reste donc ici un travail d'information et de raisonnement à effectuer.

1.4.2 *Sticky policies*

Comme nous venons de le voir, des outils comme P3P n'assurent pas réellement la protection des données. Une fois qu'une politique de traitement est déterminée, il faut donc que les processus de traitement, de transmission et de stockage des données se chargent de l'appliquer. Une approche courante (et commune à de nombreuses propositions) consiste à attacher aux informations sensibles les méta-données de description de la politique de sécurité associée, les applications s'engageant à respecter cette « politique collante ». Ces *sticky policies* ont été introduites par Günter Karjoth et Matthias Schunter en 2002 [KS02].

La proposition, dans son principe, attache des règles aux données personnelles, qui ne peuvent être manipulées par l'application que si ces règles sont respectées. On retrouve par exemple cette idée dans l'architecture intégrée du projet PRIME [Eur08, ACC+06] ou dans l'architecture proposée parallèlement par Marco Casassa Mont *et al.* [CPB03]. Si le concept est intuitif, pratique et adapté à la distribution des applications et des données (permettant un premier pas vers une réelle **protection étendue**), il ne donne cependant (dans sa version de base) aucune garantie à l'utilisateur quant au respect de la politique par une application distante. Il nous faudra donc détailler comment comment les *sticky policies* peuvent être utilisées de manière à réellement contraindre l'utilisation des données à distance. Les différents types d'utilisation de ce concept devront être analysés en fonction des garanties qu'ils fournissent à un observateur distant et au propriétaire des données en particulier. C'est ce que nous ferons plus en détail dans le chapitre 5.

Il convient également d'observer, en ce qui concerne cette famille de propositions, que la source des dites politiques n'est pas toujours spécifiée. Bien souvent, elle est issue de négociations entre les parties ou directement déduite de politiques déclaratives de type P3P. Ce mode de fonctionnement ne permet donc pas nativement de prendre en compte les contraintes réglementaires ou légales applicables aux traitements, que ce soit à la création de la politique ou bien au moment du traitement de l'information. Nous sommes donc toujours ici en besoin d'un outil adapté pour manipuler et prendre en compte les contraintes issues des réglementations.

1.4.3 Gestion déportée des données sensibles

Des propositions ont également été faites pour permettre aux utilisateurs de profiter de services en ligne tout en évitant à ces derniers de pouvoir tracer leurs activités. C'est un type d'application qui relève donc davantage de l'accès anonyme aux services et des autorisations préservant la vie privée, autres dimensions de la protection de la vie privée décrites dans les sections 1.2.3 et 1.2.4. Néanmoins, certaines de ces propositions se rapportent plus particulièrement à la gestion des données personnelles de l'utilisateur dans ces scénarios. C'est le cas notamment du protocole IDsec [Int], établi par l'IETF. Il consiste en la déportation de la gestion des données utilisateur sur un serveur spécialisé, mettant en œuvre des mécanismes de contrôle d'accès sophistiqués, visant s'assurer du bien-fondé des différentes demandes d'accès au profil qui lui sont faites.

En préalable au déroulement d'une transaction entre l'utilisateur et le fournisseur de service, l'utilisateur s'identifie auprès du serveur gestionnaire de son profil, qui en retour lui procure un certificat de session (qui servira de jeton d'accès temporaire). Ce certificat est ensuite transmis au service, qui le présente au gestionnaire de profil, accompagné d'une requête concernant le profil de l'utilisateur et d'un certificat de créance sur ses propres propriétés techniques. Le gestionnaire de profil valide le certificat de session et examine le certificat de créance. Si le gestionnaire est satisfait par ce certificat, c'est-à-dire si la correspondance entre la requête et les propriétés, quelles qu'elles soient, du fournisseur de service correspondent aux exigences connues de l'utilisateur, alors les informations de profil sont transmises au service.

Les approches de ce type ont apporté des idées intéressantes, notamment dans le cadre de la gestion des identités virtuelles, mais souffrent de limitations discriminantes. Tout d'abord, la localisation des données personnelles de l'utilisateur sur un serveur délocalisé et clairement identifié pose un problème de sécurité mis en avant par les concepteurs mêmes d'IDsec : le serveur gestionnaire, dépositaire de nombreuses données potentiellement sensibles, devient en effet une cible privilégiée pour des attaques informatiques. Cet aspect du problème milite fortement en faveur d'une gestion des données personnelles directement par leurs propriétaires, de manière distribuée. De plus, ce protocole ne s'inquiète que de l'accès initial aux données et ne fournit aucun moyen pour assurer leur **protection étendue**. Enfin, les possibilités offertes à l'utilisateur de spécifier des propriétés techniques à vérifier pour qu'un fournisseur de service puisse accéder à telle ou telle partie de son profil personnel sont très limitées en termes d'expressivité. En effet, pour traiter réellement de protection des données personnelles, il faudrait pouvoir définir des politiques capables de se référer aux six axes définis dans la section 1.3.1.

1.4.4 Agents utilisateurs

Certaines propositions émanant du domaine des systèmes multi-agents impliquent des agents artificiels [Fer95] dans la protection des données personnelles des utilisateurs. Dans ce contexte, les agents désignent des entités logicielles capables d'interagir de manière autonome avec d'autres entités ainsi qu'avec l'environnement dans lequel elles sont situées. Ces agents peuvent être des briques logicielles destinées à mettre en œuvre le traitement en lui-même, comme dans l'architecture proposée par John J. Borking [Bor00]. Ils peuvent également se présenter sous la forme d'agents mobiles et se déplacer avec les données. Dans des travaux comme ceux de Stéphanie Riché, Gavin Brebner et Mickey Glitter [RBG02, RB03], ces agents sont des assistants logiciels au service d'un utilisateur placé au centre de l'application. Ces agents personnels ou **agents utilisateurs** sont alors chargés de surveiller et de contrôler l'utilisation qui est faite des données, de manière que cette utilisation reste conforme avec la politique établie.

Cette approche permet de répondre au problème majeur posé par la gestion déportée des données personnelles par une reprise de contrôle de l'utilisateur sur ses informations, tout en décentralisant les fonctionnalités de raisonnement dans des entités autonomes. L'agent ou les agents utilisateurs permettent d'interfacer les relations entre l'utilisateur humain et les différents services et application, en lui confiant la responsabilité de certaines décisions (comme par exemple dans le cas du consentement a priori). Dans cette perspective, c'est l'agent utilisateur qui met en œuvre les différents mécanismes (principalement de contrôle d'accès) assurant la sécurité des données personnelles de l'utilisateur.

Le paradigme des agents utilisateurs est donc séduisant à plusieurs titres. Si le principe de l'agent n'est pas en soi un outil de protection des données personnelles, il peut être chargé de leur mise en œuvre. Les travaux existants dans le domaine des agents autonomes et des systèmes multi-agents permettent en outre de bénéficier d'outils théoriques et pratiques qui font des agents des entités capables d'un grand pouvoir d'adaptation et de planification. Les agents sont traditionnellement dotés de capacités de raisonnement plus ou moins poussées, qui pourraient nous permettre de combler le vide laissé par les propositions existantes : en effet, encore une fois les systèmes proposés ne sont pas capable de s'adapter aux diverses réglementations existantes en matière de protection des données personnelles, la seule source pour la création de politiques étant l'utilisateur.

1.5 Discussion

Cet aperçu des problématiques inhérentes à la protection de la vie privée et des données personnelles en particulier amène diverses réflexions sur les solutions existantes et les travaux restant à conduire en la matière.

1.5.1 Représentation et raisonnement

L'étude de la sémantique du problème de la protection des données personnelles, de son contexte réglementaire et des technologies actuellement utilisées pour l'assurer mettent en avant un besoin (et un manque) fondamental : celui de permettre, au niveau des solutions techniques, une représentation des réglementations en vigueur, un raisonnement sur ces réglementations et une adaptation dynamique à celles-ci. Nous avons vu en effet que les solutions proposées se référaient à des politiques de sécurité qui n'étaient pas directement reliées aux contraintes légales ou réglementaires pesant sur le contexte d'exécution du traitement, alors que ces contraintes ont une sémantique riche, précise et surtout variable. Suivant la localisation géographique des traitement, l'identité des tiers et l'autorité sous laquelle les traitements sont effectués, divers textes légaux, réglementaires ou contractuels peuvent entrer en jeu ou pas.

De plus, il est nécessaire que les outils mis en œuvre, quels qu'ils soient, disposent d'un moyen de représentation des données qui permettent de les mettre en relation avec lesdites réglementations. Il doit être possible de raisonner sur la nature de ces données, leur propriétaire, leur destination et les contraintes qui pèsent sur elles. Les propositions que nous allons formuler devront donc être centrées autour de cette question du représentation et du raisonnement, qui font majoritairement défaut aux solutions actuelles.

1.5.2 Le problème de la confiance

Un autre sujet qui nous semble important à la lecture des propositions existantes est celui de la confiance. L'objectif commun des propositions citées est de permettre à l'utilisateur de

protéger au mieux ses données personnelles. Cependant, dès lors que ces données sont confiées à des agents tiers, la conviction que l'utilisateur peut avoir que la dite protection soit assurée repose sur la confiance qu'il a dans les agents en question (et éventuellement sur leurs interlocuteurs). Ces agents étant précisément les entités à qui profiterait, le plus souvent, une violation de la politique de protection des données, la confiance par défaut de l'utilisateur dans un environnement ouvert doit rester très faible.

Il apparaît intuitivement que suivant les propositions, des niveaux de contraintes différents sont imposés aux dits agents, conduisant ainsi à des niveaux de confiance différents de l'utilisateur. En effet, plus il est difficile à un agent tiers de contrevenir à une politique de traitement des données et plus l'utilisateur peut être convaincu que cette dernière sera respectée. Il apparaît nécessaire d'évaluer ce qui permet à une solution technique d'élever ce niveau de confiance, afin d'être en mesure de concevoir des architectures les plus satisfaisantes possibles. Il serait également intéressant de pouvoir déporter la confiance de l'utilisateur sur une entité qui ne soit pas l'agent tiers auquel sont confiées les données, afin que cette confiance ne puisse être entâchée par l'intérêt de ce dernier à contrevenir aux politiques de traitement.

1.5.3 Agents humains et agents artificiels

Les problématiques que nous venons de présenter concernent essentiellement les utilisateurs humains du système, et les solutions proposées sont orientées par cette considération. De ce fait, les propositions que nous avons décrites comprennent peu d'automatisations de nature à mettre en relation les exigences de la problématique avec les techniques utilisées. L'association entre ces agents humains et des agents artificiels, servant d'interface avec les applications et les services, permettrait d'automatiser nombre de procédures et de décisions. Au vu de l'observation des technologies mettant en œuvre des agents utilisateurs, ce choix de conception nous paraît particulièrement adapté de par la délégation qu'il constitue en termes de raisonnement. Il convient donc de s'intéresser à ce que le paradigme des agents artificiels (et notamment cognitifs) peut apporter pour la protection des données personnelles.

Chapitre 2

Agents cognitifs et normes

L'objectif que nous poursuivons est la réalisation d'un outil logiciel capable d'assister un utilisateur humain dans sa gestion des données personnelles (les siennes ou celles d'autrui) dans le but de les protéger au mieux. Cet outil devra donc prendre en compte d'une part la ou les réglementations qui s'appliquent en la matière dans son contexte d'exécution, d'autre part les méthodes et protocoles à sa disposition pour assurer les propriétés de la protection des données personnelles. Ce logiciel devra donc faire preuve d'autonomie, de facultés d'adaptation, de prise en compte de son environnement d'exécution et de capacités de communication avec l'utilisateur humain auquel il est associé comme avec d'autres entités logicielles. Il semble donc naturel d'appeler cet outil un agent artificiel, puisqu'il en a toutes les caractéristiques fondamentales. De plus, la nécessité dans laquelle il se trouve de raisonner sur les données comme sur les réglementations pour mettre en place des solutions de manière proactive nous oriente naturellement vers les agents cognitifs, capables de raisonnement, de planification, d'anticipation, de stratégie, plutôt que vers des agents purement réactifs se contentant de répondre à des stimuli externes.

2.1 Modèles d'agents

Les travaux de recherche dans le domaine des systèmes multi-agents comportent de nombreuses propositions de modèles pour les agents autonomes, et notamment cognitifs. Le modèle BDI est sans doute le plus connu et le plus utilisé. Il trouve des déclinaisons dans la formalisation logique du modèle mental des agents, ainsi que dans des modèles procéduraux de développement d'agents logiciels.

2.1.1 Le modèle BDI

BDI (pour *Belief - Desire - Intention*) est une famille répandue de modèles cognitifs. Les concepts BDI ont été introduits par Michael E. Bratman [Bra87] et principalement développés dans un formalisme logique par Philip R. Cohen et Hector J. Levesque, dont la communication *Intention is choice with commitment* [CL90] sert de fondation à de nombreux travaux sur les composants conceptuels des modèle cognitifs.

Un agent BDI travaille sur une base de **croyances**, qui constitue sa représentation (potentiellement partielle ou erronée) du monde, un ensemble de **désirs**, qui sont pour lui des buts désirables, et enfin un ensemble d'**intentions**, en relation avec les buts sur lesquels il s'est engagé. Les désirs d'un agent BDI peuvent éventuellement être incompatibles avec les croyances ou bien avec d'autres désirs. Un agent BDI peut conserver des désirs qu'il sait inaccessibles. Les

intentions, en revanche, doivent être compatibles entre elles et avec les croyances de l'agent, et l'agent doit se croire capable de les réaliser, faute de quoi il doit y renoncer.

La notion d'intention est centrale dans le modèle BDI. Elle a également été sujette à dispute sur sa nature philosophique et sa construction logique, et de nombreuses approches complémentaires sont venues préciser la proposition originelle de Cohen et Levesque. Ce modèle cognitif purement logique a en effet suscité de nombreuses extensions.

Le modèle proposé en 1993 par Yoav Shoham [Sho93] utilise par exemple, outre les croyances, les notions d'**obligation** et d'**engagement** (**commitment**) pour fonder les décisions d'un agent. Andreas Herzig et Dominique Longin repartent des concepts proposés par Cohen et Levesque et utilisent les notions de buts et de buts persistants pour construire des intentions de manière cohérente [HL04]. Ils introduisent notamment la notion de **choix** pour représenter les préférences de l'agent. Dans un autre modèle, Carole Adam *et al* proposent une extension permettant de traiter des émotions au sein du modèle cognitif [AEG+06], utilisant des concepts comme l'**idéalité** ou l'**agréabilité** d'un état.

Les modèles cognitifs de type BDI s'avèrent intéressant pour une étude comme la nôtre, car ils permettent d'exprimer de manière explicite, et compréhensible pour un humain, les notions externes à l'agent. Les raisons qui poussent un agent à avoir tel ou tel comportement vis-à-vis des données personnelles, comme par exemple les réglementations en vigueur ou les informations sur la nature des données et des traitements, peuvent donc être représentées explicitement. On bénéficie de l'avantage principal du raisonnement symbolique, à savoir la capacité à expliquer et à justifier ses agissements.

2.1.2 Le modèle AgentSpeak

Le modèle d'agent AgentSpeak, proposé par Anand S. Rao [Rao96], est un exemple de modèle calculatoire visant à la mise en œuvre du modèle conceptuel de BDI. Le langage associé permet de représenter les croyances, les désirs (en fait des **buts** associés à des **plans d'exécution**) et les intentions (qui sont les plans en cours d'exécution). Un plan est dans ce modèle une pile d'**actions** associée à un but. Les actions peuvent être directement exécutables par l'agent ou représenter l'ajout d'un sous-but associé à un sous-plan.

Le cycle d'exécution d'un agent AgentSpeak est représenté figure 2.1. De manière simplifiée, ce cycle est structuré en trois étapes :

- L'agent effectue une **perception**, qui consiste en la récupération d'un **événement** (message interne ou externe) dans une file d'attente. Cet événement peut représenter l'ajout d'un but ou sa suppression. Il peut également représenter l'ajout ou le retrait d'une croyance, par exemple à la suite de la mesure d'un élément de l'environnement. L'agent utilise cette perception pour mettre à jour sa base de croyances ou sa file d'événements à traiter.
- L'agent sélectionne un événement (si la file est non vide). Si l'événement traite de croyances, la base de croyances est remise à jour. S'il traite d'un but, il essaie de l'unifier avec la tête d'un **plan d'exécution** contenu dans sa bibliothèque de plans. En cas de réussite, le plan est ajouté aux intentions, c'est-à-dire à l'ensemble des plans actifs.
- L'agent sélectionne un des plans actifs et effectue l'action en tête de pile. Cette action peut générer un événement interne (si c'est un sous-but), une modification de la base de croyances ou une action d'interaction avec l'environnement (si l'agent est équipé d'actuateurs) ou avec d'autres agents (typiquement, un envoi de message).

Le modèle AgentSpeak se distingue, parmi les différents avatars de BDI, par sa facilité de mise en œuvre technique due à l'éloignement d'avec des modèles purement logiques. Le modèle

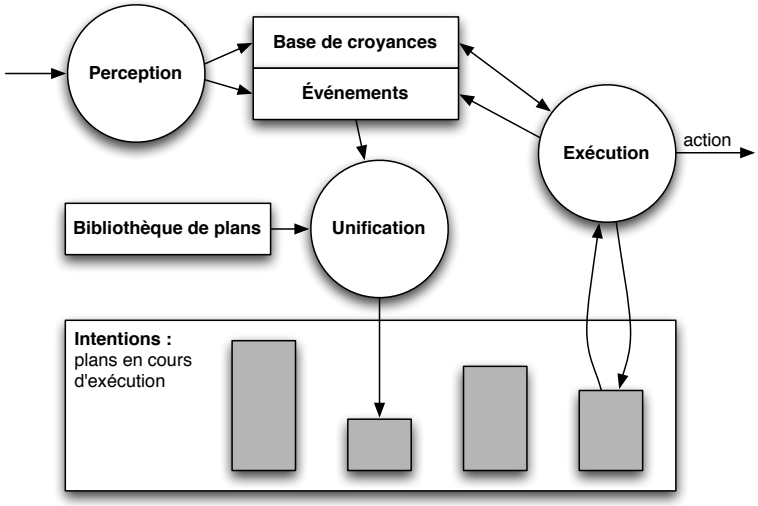

FIG. 2.1 – Boucle d'exécution du modèle AgentSpeak

d'agent y est bien défini, à la fois doté de capacités cognitives fondamentales (aisément extensible via la sémantique des buts) et d'un schéma d'exécution séquentiel et déterministe.

La plate-forme de développement Jason, développée par Rafael H. Bordini et Jomi Fred Hübner, est une mise en œuvre directe du modèle AgentSpeak [BH06]. Jason dispose d'extensions pratiques au modèles assez intéressantes, comme la possibilité d'attacher des méta-données aux croyances afin de conditionner leur utilisation par les plans. Ce type de fonctionnalité permet de mettre en place de manière assez simple un contrôle d'accès sur certaines croyances, par exemple celles identifiées comme des données personnelles.

2.1.3 Les modèles PRS, dMars et JACK

D'autres modèles de type procédural sont apparus parallèlement à AgentSpeak, avec des concepts de mise en œuvre proches. Un des plus connus est le modèle PRS (pour *Procedural Reasoning System*), conçu par Michael P. Georgeff et Amy L. Lansky [GL87], qui a évolué pour aboutir au modèle dMars, proposé en 1997 par Mark d'Inverno *et al* [dKLW97]. Ces modèles sont plus complexes qu'AgentSpeak, bien qu'utilisant le même type de boucle d'exécution. En particulier, ils autorisent la mise en œuvre de méta-raisonnement (utilisation de stratégies) lors de la sélection d'événements ou de l'unification avec un plan.

La plate-forme de développement JACK [AOS], développée et commercialisée par la société AOS, est une mise en œuvre et une évolution du modèle dMars. JACK fournit un modèle d'agent BDI procédural et multi-thread, fondé sur les notions de croyances, de buts, de plans et d'événements. Il permet le développement d'agents ou de modèles d'agents cognitifs spécialisés, comme par exemple le modèle CoJACK développé par la même société. La plate-forme JACK s'appuie sur une extension du langage Java pour fournir des outils de nature logique, comme la possibilité pour un agent de maintenir la valeur de vérité d'une assertion, concept pratique pour représenter le respect d'une réglementation ou d'une obligation. Les bases de croyances disposent également de méthodes de rappel (*callback*) personnalisables, permettant de conditionner de manière très fine l'utilisation d'une croyance à l'exécution d'un plan de rappel, autorisant ainsi la mise en place de mesures de sécurité extensibles.

Du point de vue de sa valorisation, le logiciel a également l'avantage de bénéficier d'une communauté d'utilisateurs tant académiques que gouvernementaux ou industriels (comme par exemple les gouvernements australien et canadien, ou encore Dassault Aviation [DM02]), motivant le dynamisme du développement et de la maintenance de la plate-forme, ainsi que la pérennité du modèle économique sous-jacent.

Pour toutes ces raisons scientifiques, techniques et pratiques, nous avons retenu la plate-forme JACK plutôt qu'une autre pour nos travaux de développement sur la protection des données personnelles dans les systèmes multi-agents.

2.2 Systèmes normatifs

Un agent chargé de protéger les données personnelles, quel que soit son modèle, doit faire partie d'un système au sein duquel son comportement pourra être mis en regard des réglementations. Le champ scientifique traitant des **systèmes normatifs** étudie notamment cette correspondance entre les comportements souhaités et les comportements réels.

2.2.1 Caractériser les comportements appropriés

En protection des données personnelles comme dans d'autres domaines, la réglementation et les diverses contraintes qui régissent le processus peuvent être considérées comme une pression sociale exercée sur des individus afin de les amener à un comportement ou un état conforme à une règle. Les prescriptions sociales qui s'appliquent aux individus (que nous nommerons agents de manière générique, qu'ils soient humains ou artificiels) sont alors appelées **normes**. Le dictionnaire de l'Académie française définit le terme « norme » de la manière suivante :

> « Type, état, comportement qui peut être pris pour référence ; modèle, principe directeur qu'on tire de l'observation du plus grand nombre. »

C'est bien évidemment une définition générique, mais qui s'entend bien dans le domaine des systèmes normatifs comme la caractérisation d'un comportement adéquat. Ces normes sont potentiellement de natures variées, établies de diverses manières, perçues et respectées de manière différente par chaque agent.

2.2.2 Les différents types de normes

Raimo Tuomela a établi en 1995 une classification des différents types de normes, suivant leur nature, leur perception et leur acceptation par les agents du système [Tuo95]. Il distingue :

- les **r-normes** (règles), qui s'imposent aux agents parce qu'il y a un accord entre eux pour les respecter (ce sont par exemple les lois et les réglementations) ;
- les **s-normes** (normes sociales), que les agents respectent à cause de leur conscience sociale, parce qu'ils savent que les autres agents s'attendent à les voir les respecter (c'est par exemple le type de norme qui fait que la plupart des gens ne publient pas leur carnet d'adresses sur leur site internet) ;
- les **m-normes** (normes morales), que les agents respectent parce que leur conscience individuelle le leur dicte (c'est par exemple ce qui pousse une personne à ne pas épier un ami en train de taper son code de carte bancaire) ;
- les **p-normes** (normes de prudence), que les agents respectent parce qu'elles constituent pour eux la stratégie rationnelle à suivre (c'est le type de norme qui pousse un individu à ne pas divulguer un secret gênant concernant un tiers de qui il attend un service).

Bien évidemment, une norme en particulier peut correspondre simultanément à plusieurs de ces catégories, pour un même agent.

Dans le cadre des systèmes multi-agents, nous disposons d'autres outils pour représenter les p-normes, qui concernent davantage le comportement guidé par les buts des agents. Les r-normes sont les plus évidemment reliées au type de problème qui nous intéresse : elles sont émises par un accord qui constitue une autorité reconnue (comme le contrat social d'un État ou un contrat privé entre agents) qui les impose à tout ou partie des agents. Les m-normes peuvent également être considérées de la même manière, si l'on estime que l'agent lui-même représente en l'occurence cette autorité. Les s-normes, fortement liées aux deux précédentes, ne rentrent en ligne de compte que dans les scénarios où le comportement de l'agent est observable par les autres, auxquels cas elles peuvent être considérées de la même manière que des r-normes ou des m-normes en introduisant une autorité virtuelle.

Dans ce contexte, nous retiendrons la définition suivante :

Définition 2.1 (Norme)**.** Une norme est la spécification d'un état ou d'un comportement jugé adéquat, dans un contexte donné, par un agent ou un ensemble d'agents.

2.2.3 Systèmes normatifs et systèmes multi-agents

Disposant de la notion de norme, on peut dériver le concept de **système normatif**, qui est un système social comportant à la fois des normes et des agents, la relation entre les unes et les autres différant suivant les auteurs. La notion de système normatif a été largement transposée dans le domaine de l'informatique et de l'intelligence artificielle (par des auteurs comme Andrew J. I. Jones et Marek J. Sergot notamment [JS93]), qui est le point de vue qui nous intéresse ici.

En préface de leur ouvrage édité sur les systèmes normatifs [MW93], John-Jules Charles Meyer et Roel J. Wieringa proposent pour l'expression « système normatif » une définition assez générique et quasiment auto-référentielle :

> "[Normative systems are] systems in the behavior of which norms play a role and which need normative concepts in order to be described or specified."

Le paradigme des systèmes multi-agents a par la suite rapidement été intégré à la notion de système normatif en informatique, de par la naturelle adéquation de la métaphore sociale qu'il représente. Pour autant, la notion de **système multi-agent normatif** ne se résume pas à l'introduction dans le système d'entités nommés agents, elle fait jouer à plein leur caractéristique fondamentale d'autonomie pour enrichir cette relation encore floue entre les normes et le fonctionnement du système normatif.

2.2.4 Systèmes d'agents normatifs et enrégimentation

Ågotnes *et al*, donnent une interprétation (qui peut paraître assez restrictive) de la relation entre normes et fonctionnement du système normatif dans leur définition d'un système multi-agent normatif [ÅvdHR+07] :

> "[A normative multi-agent system is] a set of constraints on the behaviour of agents in the system."

Ici l'on voit poindre, si l'on considère ces contraintes comme strictes, l'aspect le plus sécuritaire possible de la notion de système multi-agent normatif, à savoir l'**enrégimentation**[1] des agents.

[1] Bien que le terme « enrégimentation » n'existe pas en français, nous prendrons la liberté de l'utiliser comme substantif naturel du verbe « enrégimenter », auquel nous donnerons l'acception particulière précisée ici.

Dans un système enrégimenté, on rend toute violation des normes strictement impossible aux agents, par voie programmatique ou par toute autre contrainte de leur environnement d'exécution. Ce type de système devient donc un **système d'agents normatifs** (cas particulier de système multi-agent normatif), tel que décrit par Tiberiu Stratulat [Str02]. C'est un systèmes dans le quel les agents sont normatifs, c'est-à-dire qu'ils respectent strictement les normes.

2.2.5 Subjectivité et violations

Guido Boella, Leendert van der Torre et Harko Verhagen proposent une définition beaucoup plus souple d'un système multi-agent normatif [BvV06] :

> "A normative multiagent system is a multiagent system together with normative systems in which agents on the one hand can decide whether to follow the explicitly represented norms, and on the other the normative systems specify how and in which extent the agents can modify the norms."

Cette nouvelle définition introduit deux concepts fondamentaux : les normes sont dynamiques et elles peuvent être violées.

Dans les systèmes multi-agents normatifs les plus riches, les agents disposent de suffisamment d'autonomie et de capacités cognitives pour considérer les normes et décider ou non de les suivre (ou d'adopter toute autre position intermédiaire). Néanmoins, tous les agents interprètent les normes de la même manière : qu'ils décident ou non de suivre une norme, cette dernière a la même signification pour tous. Les agents (tous ou seulement certains) peuvent également édicter de nouvelles normes ou modifier les existantes. Un système multi-agent normatif est donc essentiellement dynamique et laisse une grande part à l'autonomie des agents.

Nous retiendrons comme définition du système multi-agent normatif une adaptation (introduite dans une précédente communication [PD08c]) de celle de Boella *et al*, évitant de se reposer directement sur le concept de système normatif et détaillant l'autonomie des agents :

Définition 2.2 (Système multi-agent normatif)**.** Un système multi-agent normatif est un système multi-agent associé à un ensemble de normes sur le comportement des agents, permettant le choix individuel des normes à suivre, l'évolution de ces normes, la représentation et le traitement des violations.

L'autonomie ainsi introduite implique, nous le voyons, la possibilité de violer les normes. Il appartient donc à un système multi-agent normatif de caractériser ces violations (en fonction du formalisme retenu pour spécifier le système) et éventuellement de leur associer des sanctions. Ici se pose un problème majeur si l'on veut appliquer le paradigme des systèmes multi-agents normatifs à la protection des données personnelles. En effet, si l'on identifie le cas général d'une application distribuée ouverte (par exemple une collaboration entre agents utilisateurs et agents de service sur Internet) à un système normatif, il faut, pour pouvoir appliquer des sanctions, que les violations puissent être détectées par une autorité reconnue. Or, dans le cas qui nous intéresse, la violation d'une norme est souvent silencieuse (c'est-à-dire potentiellement indétectable par d'autres agents que le contrevenant). Si l'on prend par exemple le cas simple de la rétention d'information, aucun agent du système ne peut avoir de conviction ferme sur le fait qu'un autre agent a supprimé une donnée particulière.

Les notions de subjectivité et d'autonomie posent également le problème fondamental de l'interprétation des normes. Même dans le cas où ces normes sont édictées dans un formalisme compréhensible par tous les agents (au niveau de sa syntaxe et des mécanismes de raisonnement pouvant lui être appliqués), la signification profonde des éléments du langage normatif (le

lien qu'ils entretiennent avec le monde physique ou les actions des agents) n'est pas forcément universellement partagée par les agents du système. Un élément sémantique du langage pourra faire référence à tel jeu d'actions pour un des agents et à tel autre pour un deuxième agent. Ce phénomène augmente encore la subjectivité de l'évaluation dans les systèmes ouverts, que peuvent rejoindre des agents développés et contrôlés par des entités ou utilisateurs indépendants ayant interprété le langage de normes de manières différentes.

La plupart des applications pouvant être considérées comme des mises en œuvre théoriques ou logicielles du concept de système multi-agent normatif (comme par exemple la plate-forme MOISE+, proposée par Jomi Fred Hübner, Jaime Simão Sichman et Olivier Boissier [HaSB02]) s'appuient sur une possible introspection des agents par une autorité centrale pour appliquer des sanctions. Cela est évidemment impossible dans le cas d'un système ouvert, sauf à mettre en place des dispositifs bien trop intrusifs pour rester acceptables aux individus comme aux organisations, comme nous l'avons détaillé dans une précédente communication [CPBD07]. Il nous paraît par conséquent délicat, dans le cadre de la protection des données personnelles, de considérer le recours aux sanctions comme un moyen efficace de lutter contre les violations. Il nous faudra donc prendre en compte le fait que les agents tiers seront très faiblement incités à respecter toutes les normes du système.

D'autre part, la possibilité que plusieurs entités puissent être à l'origine de nouvelles normes, ainsi que le caractère dynamique de ces normes, amène à se poser la question de la cohérence de l'ensemble des normes du système. Il nous faudra prendre en considération (au niveau de l'agent ou du système) le fait que dans le cas général, cette cohérence peut ne pas être assurée.

2.2.6 Centralisation et distribution

Un autre problème survient en ce qui concerne l'application que nous voudrions faire des systèmes normatifs (nous avons abordé ce problème dans [CPBD07]). Dans la quasi-totalité (sinon la stricte totalité) des modèles de systèmes multi-agents normatifs, le système normatif en lui-même désigne une autorité centrale, celle-là même chargée de s'assurer de l'application des normes et d'infliger des sanctions. Cette autorité est alors garante de la cohérence de l'ensemble des normes. Encore une fois, si nous transposons cette architecture à une application ouverte à l'échelle d'Internet, il devient difficile de décider quelle institution pourrait jouer le rôle d'une telle autorité. Les agents en interaction peuvent dépendre d'autorités étatiques différentes, de législations différentes, de sociétés différentes, ils peuvent agir sous la contrainte de contrats différents. Les autorités sont multiples et non partagées, et pour cette raison le modèle centralisé ne peut convenir à ce type d'application.

Dans le contexte qui nous intéresse, les normes devront donc être évaluées au niveau de chaque agent. C'est également au niveau de l'agent que doit se faire l'évaluation de la cohérence de l'ensemble de normes. Ceci est dû à l'autonomie des agents dans le système, mais également au fait que les normes qui s'appliquent sont différentes pour chaque agent et qu'il ne peut y avoir d'autorité centrale. De plus, à cause du problème d'introspection déjà évoqué, le recours à des autorités de contrôle multiples ne peut être satisfaisant. Pour ces raison, la caractérisation des violations doit être locale. L'évaluation d'une violation ne peut être menée, dans le pire cas, que par l'agent en infraction, les violations d'agents tiers ne pouvant être la plupart du temps que supposées. Cette évaluation est d'autant plus subjective qu'elle est subordonnée à la gestion par l'agent d'une potentielle incohérence dans les normes qui lui sont imposées.

2.2.7 Nécessité d'un formalisme

Un agent en charge de la gestion et de la protection de données personnelles doit donc être capable de raisonner de manière efficace non seulement sur les concepts propres à son modèle cognitif, mais aussi sur les notions de norme, d'autorité, de violation. Dans cette perspective, nous examinons maintenant les formalismes, fondés sur la logique modale, qui ont fourni des modèles pour la mise en œuvre d'agents cognitifs BDI comme pour le raisonnement normatif.

2.3 Éléments de logique modale

Le modèle BDI de Cohen et Levesque [CL90], comme nombre de ses extensions et d'autres contributions dans le domaine des modèles cognitifs, sont exprimés dans le formalisme de la logique modale. La bonne compréhension des propositions que nous allons formuler nécessite l'introduction des notions de base de la logique modale, des outils et des formalismes que nous utilisons[2]. D'une manière générale, les logiques modales ont été et sont toujours un instrument formel privilégié pour la conception de modèles d'agents cognitifs ou de structuration des interactions multi-agents [Woo01a].

Une logique modale propositionnelle [Che80, BdV01] est l'extension d'un langage propositionnel, par opposition par exemple aux logiques modales du premier ordre. Par la suite et sauf mention contraire, le terme « logique modale » désignera une logique modale propositionnelle. D'une manière générique nous noterons $\mathcal{L}$ le langage propositionnel de base et ferons référence à ses propositions par les caractères p, q et à suivre. Les formules bien formées d'une logique modale seront représentées par des lettres grecques minuscules (φ, ψ, ρ).

2.3.1 Modalités

Une logique modale étend un langage $\mathcal{L}$ de propositions logiques par des **modalités**, qui représentent des quantifications de nature universelle ou existentielle sur des variables implicites (non représentées dans les formules). La modalité universelle générique se note généralement $\Box$ (« *box* »), et son dual existentiel $\Diamond$ (« *diamond* »). Dans le cas le plus simple (logique monomodale), une formule bien formée φ d'une logique modale respecte la grammaire (2.1), où p est une proposition du langage de base $\mathcal{L}$ (les opérateurs $\wedge$ et $\rightarrow$ seront définis comme à l'accoutumée, on s'autorise également l'utilisation de $\top$ pour « vrai » et de $\bot$ pour « faux »). La formule (2.2) définit le dual $\Diamond$ d'une modalité $\Box$.

$$\varphi = p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Box\varphi \tag{2.1}$$

$$\Diamond\varphi \stackrel{def}{=} \neg\Box\neg\varphi \tag{2.2}$$

Dans la suite nous considérerons les modalités existentielles comme des abréviations du langage, mais il est possible de considérer au contraire $\Box\varphi$ comme une abréviation de $\neg\Diamond\neg\varphi$. Suivant les applications, la modalité générique $\Box$ est différenciée, spécialisée pour correspondre à une notion porteuse de sens comme par exemple la nécessité, (notée $\Box$ en logique aléthique), la connaissance (K en logique épistémique), la croyance (Bel en logique doxastique), l'obligation

[2] Les sections 2.3 et 2.4 ont été rédigées parallèlement à des contributions significatives de notre part aux articles de la Wikipédia francophone sur la logique modale [Wik08b], la logique déontique [Wik08a], la sémantique de Kripke [Wik08d] les formules de Sahlqvist [Wik08c]. Ceci explique certaines similarités de structure et d'éléments d'illustrations.

(*Ob* en logique déontique)... Les modalités sont donc des opérateurs exprimant un concept plus ou moins complexe à propos de la formule sur laquelle ils sont appliqués. Les logiques multi-modales comportent plusieurs de ces couples de modalités, qui peuvent représenter des concepts distincts.

Le caractère universel ou existentiel d'une modalité se retrouve dans la traduction d'une formule modale en logique du premier ordre (« traduction standard »). Les atomes propositionnels y deviennent des prédicats unaires, et des relations entre variables du premier ordre y sont introduites. Les formules (2.3) et (2.4) sont des exemples de traductions standard que que nous n'expliciterons pas ici, car elles font référence à la sémantique des logiques modales[3]. Elles nous servent uniquement à illustrer le fait que les logiques modales sont bien des fragments de la logique du premier ordre.

$$trad_x(\Box p) = \forall y(R(x,y) \to P(y)) \tag{2.3}$$

$$trad_x(\Box\Diamond(p \vee q)) = \forall y(R(x,y) \to \exists z(R(y,z) \wedge (P(z) \vee Q(z)))) \tag{2.4}$$

2.3.2 Axiomatiques

Les logiques modales dites « normales » sont les plus utilisées, à cause notamment de la représentation intuitive de leur sémantique. Ces logiques sont caractérisées par la règle d'inférence (RN) (dite règle de nécessitation) et par l'axiome[4] (K)[5]. La règle (RN) permet de déduire, à partir d'un théorème φ, le théorème $\Box\varphi$. L'axiome (K) exprime la propriété de distributivité de la modalité universelle par rapport à l'implication logique. Les logiques modales normales héritent également des règles d'inférence (*modus ponens*, instanciation universelle) et des tautologies de la logique propositionnelle.

$$\frac{\varphi}{\Box\varphi} \tag{RN}$$

$$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi) \tag{K}$$

Un système de logique modale normale est généralement noté $K\xi_1 \ldots \xi_n$, où $\xi_1 \ldots \xi_n$ sont les axiomes introduits en sus de l'axiome (K). Par exemple, le système de logique modale *KD45*, utilisé en logique doxastique, comprend les axiomes (K), (D), (4) et (5).

$$\Box\varphi \to \Diamond\varphi \tag{D}$$

$$\Box\varphi \to \Box\Box\varphi \tag{4}$$

$$\Diamond\varphi \to \Box\Diamond\varphi \tag{5}$$

Le système *KT5*, lui, fera appel aux axiomes (K) et (5), ainsi qu'à l'axiome (T) :

$$\Box\varphi \to \varphi \tag{T}$$

[3]Pour les lecteurs déjà familiers avec la sémantique de Kripke, la traduction standard $trad_x$ d'une formule modale se fait pour un monde donné, représenté par la variable du premier ordre x. La traduction fait appel à une relation R entre variables du premier ordre qui correspond à la relation d'accessibilité $\mathcal{R}$ entre les mondes du modèle.

[4] Par souci de lisibilité, nous utiliserons librement le terme « axiome » en lieu et place de « schéma axiomatique ».

[5] L'axiome (K) (pour « Saul Kripke ») ne devra pas être confondu avec la modalité K (pour « *knows* ») de la logique épistémique.

2.3.3 Sémantique de Kripke

Les formules d'une logique modale normale (monadique) sont interprétées sur des structures introduites par Saul Aaron Kripke autour de 1960 [Kri59, Kri63a, Kri63b] et fondées sur des concepts proches de ceux introduits par Hintikka [Hin62] ou Kanger [Kan57], notamment.

2.3.3.1 Modèles de Kripke

Dans un modèle de Kripke, une formule est interprétée pour un « monde » donné. Un monde peut par exemple représenter un état du système, un instant du temps ou toute autre réalité. Pour chaque monde, chaque atome propositionnel de $\mathcal{L}$ est défini comme étant vrai ou faux. Les mondes sont reliés entre eux par une relation binaire dite d'accessibilité. La valeur de vérité d'une formule modale dans un monde donné est ensuite définie à partir de la valeur de vérité des formules dans les mondes accessibles. En effet, $\Box\varphi$ est vraie dans un monde donné si et seulement si φ est vraie dans tous les mondes accessibles, et $\Diamond\varphi$ est vraie dans un monde donné si et seulement si φ est vraie dans au moins un monde accessible. C'est une représentation relationnelle de la traduction standard des modalités universelles et existentielles en logique du premier ordre. La valeur de vérité des formules est évaluée par induction sur la structure de la formule, les opérateurs non modaux étant traités localement[6].

Ainsi, si l'on considère un agent traitant des données personnelles, on peut imaginer une modalité *Regl*, où *Regl* φ signifie qu'une réglementation impose que φ soit vraie. Son expression relationnelle dit que pour tout état de l'agent respectant les réglementations (soit tout état accessible par une série d'actions autorisées), φ est vraie. La représentation modale permet ici d'encapsuler une notion complexe dans une formule plus simple à manipuler, même si la nature même de la notion d'accessibilité (les actions autorisées) est pour l'instant cachée.

La caractérisation formelle de la sémantique d'une logique monomodale normale est la suivante.

Définition 2.3 (Modèle de Kripke)**.** Un **modèle de Kripke** est un triplet $\mathcal{M} = \{\mathcal{W}, \mathcal{R}, h\}$, où $\mathcal{W} = \{w_i\}$ est l'ensemble des **mondes possibles**, $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$ est la **relation d'accessibilité** et $h : atom(\mathcal{L}) \longrightarrow \wp(\mathcal{W})$ est la **fonction de valuation**[7].

2.3.3.2 Interprétation des formules

La fonction de valuation, appliquée à un atome propositionnel, donne l'ensemble des mondes auxquels cet atome est vrai. La notation $\mathcal{M}, w \models \varphi$ se lit « la formule φ est vraie au monde w du modèle $\mathcal{M}$ ». L'opérateur de modélisation $\models$ est défini par induction sur la structure des formules. Les formules suivantes s'entendent pour tout monde w de tout modèle de Kripke $\mathcal{M}$ et traduisent formellement les concepts déjà introduits.

[6] C'est-à-dire qu'une négation est vraie en un monde donné si et seulement si la formule niée n'est pas vraie en ce même monde et qu'une disjonction est vraie en un monde donné si et seulement si l'un au moins de ses termes est vrai en ce même monde.

[7] Ici, $\wp(\mathcal{W})$ est l'ensemble des parties de $\mathcal{W}$ et $atom(\mathcal{L})$ est l'ensemble des atomes propositionnels du langage $\mathcal{L}$.

$$\mathcal{M}, w \models \top \tag{2.5}$$
$$\forall p \in atom(\mathcal{L}), \mathcal{M}, w \models p \text{ si et seulement si } w \in h(p) \tag{2.6}$$
$$\mathcal{M}, w \models \neg\varphi \text{ si et seulement si } \mathcal{M}, w \not\models \varphi \tag{2.7}$$
$$\mathcal{M}, w \models \varphi \vee \psi \text{ si et seulement si } \mathcal{M}, w \models \varphi \text{ ou } \mathcal{M}, w \models \psi \tag{2.8}$$
$$\mathcal{M}, w \models \Box\varphi \text{ si et seulement si } \forall w' \in \mathcal{W}, w\mathcal{R}w' \text{ implique } \mathcal{M}, w' \models \varphi \tag{2.9}$$

La figure 2.2 représente un exemple de modèle de Kripke, et les formules (2.10) à (2.15) expriment des valeurs de vérité en différents mondes du modèle.

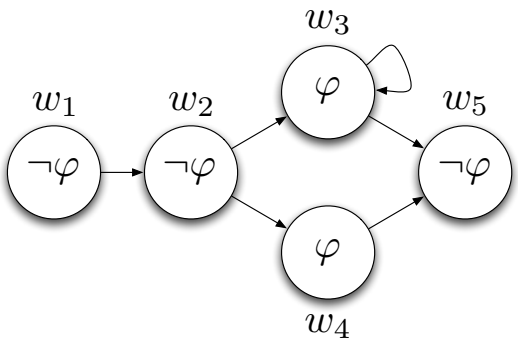

FIG. 2.2 – Exemple de modèle de Kripke

$$\mathcal{M}, w_1 \models \neg\varphi \tag{2.10}$$
$$\mathcal{M}, w_1 \models \Box\Box\varphi \tag{2.11}$$
$$\mathcal{M}, w_2 \models \Box\varphi \tag{2.12}$$
$$\mathcal{M}, w_3 \models \Diamond\varphi \tag{2.13}$$
$$\mathcal{M}, w_5 \models \Box\neg\varphi \tag{2.14}$$
$$\mathcal{M}, w_5 \models \Box\varphi \tag{2.15}$$

2.3.3.3 Théorèmes de correspondance

Les propriétés de la relation d'accessibilité $\mathcal{R}$ correspondant à une modalité donnée sont en étroite relation avec les axiomes de cette modalité, afin d'assurer la correction et la complétude du langage modal par rapport à sa représentation sémantique. Cette correspondance entre axiomes et propriétés relationnelles a été établie par Henrik Sahlqvist et constitue un théorème fondamental de la logique modale [Sah75]. Le tableau 2.1 récapitule les plus courantes de ces correspondances, impliquant des axiomes et des propriétés auxquels nous serons amenés à faire référence par la suite. Ainsi, le système *KD45* donné plus haut en exemple correspond aux modèles de Kripke dans lesquels la relation $\mathcal{R}$ est sérielle, transitive et euclidienne.

2.3.3.4 Discussion

La sémantique de Kripke a l'avantage de donner une signification claire et bien définie aux logiques modales normales, en caractérisant le mécanisme des modalités. On peut ainsi s'en

Axiome		Propriété relationnelle	
(D)	$\Box\varphi \rightarrow \Diamond\varphi$	Caractère sériel	$\forall w \in \mathcal{W}, \exists w' \in \mathcal{W}, w\mathcal{R}w'$
(T)	$\Box\varphi \rightarrow \varphi$	Réflexivité	$\forall w \in \mathcal{W}, w\mathcal{R}w$
(4)	$\Box\varphi \rightarrow \Box\Box\varphi$	Transitivité	$\forall w, w', w'' \in \mathcal{W}$, $w\mathcal{R}w' \wedge w'\mathcal{R}w'' \rightarrow w\mathcal{R}w''$
(5)	$\Diamond\varphi \rightarrow \Box\Diamond\varphi$	Caractère euclidien	$\forall w, w', w'' \in \mathcal{W}$, $w\mathcal{R}w' \wedge w\mathcal{R}w'' \rightarrow w'\mathcal{R}w''$
$(\Box M)$	$\Box(\Box\varphi \rightarrow \varphi)$	Pseudo-réflexivité	$\forall w, w' \in \mathcal{W}, w\mathcal{R}w' \rightarrow w'\mathcal{R}w'$

Tab. 2.1 – Correspondances entre axiomes d'une logique et propriétés de la relation d'accessibilité $\mathcal{R}$

servir pour introduire une modalité représentant, pour un agent, toute une classe d'actions, d'événements ou de raisonnements, caractérisée par le fait que les états atteints sont désirables, utiles, souhaitables, autorisés, interdits... Un des inconvénients majeurs de cette sémantique, notamment pointé par Sven Ove Hansson [Han02] est qu'elle prend appui sur le caractère désirable (ou tout autre sens que l'on souhaite donner à la modalité) du monde accessible, ce qui n'est pas toujours le plus utile. Hansson a proposé des sémantiques alternatives travaillant sur la désirabilité des formules évaluées dans les mondes accessibles (et non plus sur la désirabilité des mondes eux-mêmes), sans toutefois aboutir à la cohésion générale qui rend les sémantiques de Kripke si séduisantes. Cette cohésion est issue de la correspondance entre axiomes et relation d'accessibilité, qui permet de relier le raisonnement logique, mécanique, de l'agent aux caractéristiques sémantiques de la notion modale. Dans le cas des modalités d'obligation notamment, la correspondance telle qu'elle apparaît dans la sémantique de Kripke permet de travailler sur des axiomes et des relations cohérents et intuitifs pour le concept représenté, alors qu'une proposition comme celle de Hansson induit une axiomatique partiellement contre-intuitive et impropre à représenter le raisonnement d'un agent sur des normes.

2.3.4 Logiques temporelles linéaires

Les normes traitant de la protection des données personnelles traitent souvent de concepts liés au temps. Les exemples les plus simples en sont sans doute les limitations sur la durée de rétention, qui imposent que telle ou telle donnée soit supprimée dans un temps imparti. Il est donc nécessaire pour notre agent de pouvoir raisonner sur les concepts temporels afin de pouvoir appréhender ce type de norme avec efficacité.

Les logiques temporelles constituent une famille de logiques modales normales, où les modalités représentent des notions liées au passé, au futur ou à la précédence temporelle d'événements. Dans ces logiques, les mondes représentent des instants reliés par une relation d'accessibilité définissant le sens d'écoulement du temps. Ces mondes ou instants sont généralement notés t_i (appartenant à un ensemble $\mathcal{T}$) plutôt que w_i. Dans les logiques temporelles linéaires (LTL), les instants sont ordonnés suivant une unique chaîne qui va du passé vers le futur. Ces logiques servent à décrire de manière formelles des propriétés sur les successions temporelles d'événements.

Les logiques modales du temps ont été introduites par Arthur Norman Prior [Pri57, Pri67], et les logiques temporelles linéaires en particulier se fondent principalement sur les travaux d'Amir Pnueli [Pnu77]. Un grand nombre d'améliorations et de propositions ont été apportées depuis ces travaux fondateurs, et les formalismes présentés ici visent à représenter le consensus courant

concernant les dénominations de « logique temporelle » et de « logique temporelle linéaire ». Une logique temporelle est notamment caractérisée par son **flot temporel**, soit le couple $(\mathcal{T}, C_t(\mathcal{R}))$ de l'ensemble des instants et de la fermeture transitive d'une relation d'accessibilité particulière qui associe un instant à son successeur dans le temps. On notera $<$ la fermeture transitive de $\mathcal{R}$ pour signifier son caractère total et le fait qu'elle peut permettre de comparer des instants distincts sur l'axe du temps. Ce n'est toutefois pas nécessairement un ordre ni un préordre. Dans les logiques auxquelles nous nous intéressons ici, le flot de temps est discret et sa relation de base ($\mathcal{R}$) est linéaire. Il peut également être sériel (infini vers le futur) et/ou non borné à gauche (infini vers le passé, *left-unbounded* en anglais). Comme nous l'avons vu, les caractéristiques du flot temporel, en tant que propriétés d'une relation d'accessibilité, sont directement liées aux axiomes des différentes modalités utilisées pour travailler sur le temps, modalités que nous allons maintenant détailler. Nous allons en effet avoir besoin de décrire le futur, le passé, l'antécédence dans le temps d'un événement sur un autre, les durées...

2.3.4.1 La modalité *neXt*

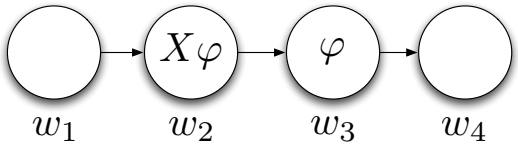

FIG. 2.3 – La modalité neXt en LTL

La modalité X (pour *neXt*) est la plus simple des modalités temporelles. Elle permet d'accéder à l'instant suivant directement l'instant présent. Son principe est illustré par la figure 2.3. La relation d'accessibilité associée à X est la relation $\mathcal{R}$ à laquelle fait référence le flot de temps du modèle. Le plus souvent, cette relation est sérielle, ce qui impose l'axiome (X-D). En LTL, elle est également linéaire à gauche comme à droite (chaque instant a au plus un seul passé et un seul avenir possibles), ce qui signifie notamment que la modalité universelle X est son propre dual existentiel (axiome (X-Alt)).

$$X\varphi \rightarrow \neg X \neg \varphi \tag{X-D}$$

$$\neg X \neg \varphi \rightarrow X\varphi \tag{X-Alt}$$

Dans le cas où le flot de temps est non borné à gauche, on peut aisément définir un opérateur converse X^{-1} aux propriétés similaires.

2.3.4.2 Les modalités G, F, H et P

La modalité G (pour *is Going to be true*) est la modalité universelle traitant du futur : elle exprime le fait qu'une formule sera vraie à tout instant du futur. La modalité F (pour *Future*) est le dual de G, elle exprime le fait qu'une formule sera vraie à un instant au moins dans le futur. Le principe des modalités F et G est illustré par la figure 2.4. Les modalités H (pour *Has been true*) et P (pour *Past*) sont les modalités universelle et existentielle, respectivement, traitant du passé. Suivant les auteurs, futur et passé englobent ou non le présent. Nous prendrons comme convention que les modalités strictes G, F, H et P excluent le présent mais que les modalités

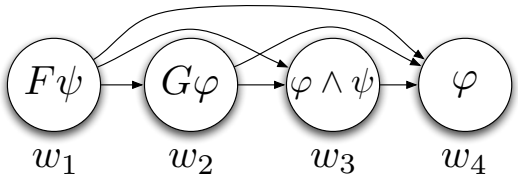

FIG. 2.4 – Raisonnement sur le futur en LTL : les modalités G et F

affaiblies G^-, F^-, H^- et P^- l'incluent. Nous aurions pu faire le choix inverse, mais les modalités affaiblies peuvent être déduites des modalités strictes, alors que l'inverse est impossible.

$$G^-\varphi \stackrel{def}{=} G\varphi \wedge \varphi \tag{G^-}$$

$$H^-\varphi \stackrel{def}{=} H\varphi \wedge \varphi \tag{H^-}$$

Ainsi, la relation correspondant à la modalité G (assortie de son dual F) est $<$, la fermeture transitive de $\mathcal{R}$. L'axiomatique de G comprend donc (G-D), qui traduit le caractère sériel de $<$, et (G-4), qui correspond à sa transitivité (cf tableau 2.1).

$$G\varphi \rightarrow F\varphi \tag{G-D}$$

$$G\varphi \rightarrow GG\varphi \tag{G-4}$$

En logique temporelle linéaire, l'axiome supplémentaire (G-.3) assure la linéarité vers le futur.

$$(F\varphi \wedge F\psi) \rightarrow F(\varphi \wedge F\psi) \vee F(\varphi \wedge \psi) \vee F(F\varphi \wedge \psi) \tag{G-.3}$$

La modalité H et son dual P sont axiomatisés de manière similaire à G et F. Le bon fonctionnement du raisonnement conjoint sur le futur et le passé implique également l'introduction des axiomes de conversion (GP) et (HF), exprimant le fait que le présent est d'une part le passé de tout futur et d'autre part le futur de tout passé.

$$\varphi \rightarrow GP\varphi \tag{GP}$$

$$\varphi \rightarrow HF\varphi \tag{HF}$$

2.3.4.3 Les modalités *Since* et *Until*

On franchit un grand pas en terme d'expressivité en rajoutant à une logique temporelle les opérateurs $\mathcal{U}$ (*Until*) et $\mathcal{S}$ (*Since*). $\mathcal{U}$ est une modalité dyadique exprimant le fait qu'une première formule va demeurer vraie dans le futur jusqu'à ce qu'une seconde formule (que nous appellerons formule cible) devienne vraie à son tour.

$\mathcal{S}$ est la modalité symétrique dans le passé, exprimant le fait qu'une première formule a été vraie depuis la dernière fois qu'une seconde formule a été vraie. Encore une fois et toujours pour favoriser l'opérateur le plus expressif, nous considérons que ces modalités s'entendent au sens strict, c'est-à-dire d'une part que le présent n'est pas inclus dans le futur ni dans le passé et d'autre part que la formule cible doit effectivement devenir vraie à un instant du futur (ou du

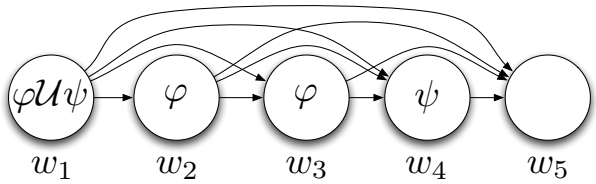

FIG. 2.5 – La modalité $\mathcal{U}$ (*Until*)

passé). En d'autres termes, la première formule n'a pas à être vraie à l'instant présent, et la formule cible doit le devenir dans un futur excluant le présent. Il est éventuellement possible de définir des modalités affaiblies $\mathcal{U}^-$ et $\mathcal{S}^-$ incluant le présent. La sémantique des modalités strictes, exprimées en fonctions de la relation $<$ sur des instants t_i d'un modèle $\mathcal{M}$, est donnée par les formules (2.16) et (2.17) et illustrée (pour la modalité $\mathcal{U}$ uniquement) par la figure 2.5.

$$\mathcal{M}, t \models \varphi\ \mathcal{U}\ \psi \text{ si et seulement si } \exists t' \in \mathcal{T}, \begin{cases} t < t', \\ \mathcal{M}, t' \models \psi \\ \forall t'' \in \mathcal{T}, \text{ si } t < t'' \text{ et } t'' < t' \text{ alors } \mathcal{M}, t'' \models \varphi \end{cases} \tag{2.16}$$

$$\mathcal{M}, t \models \varphi\ \mathcal{S}\ \psi \text{ si et seulement si } \exists t' \in \mathcal{T}, \begin{cases} t' < t, \\ \mathcal{M}, t' \models \psi \\ \forall t'' \in \mathcal{T}, \text{ si } t' < t'' \text{ et } t'' < t \text{ alors } \mathcal{M}, t'' \models \varphi \end{cases} \tag{2.17}$$

Ces deux modalités ont une grande expressivité. Il est impossible de les exprimer en fonction des modalités précédemment introduites, alors que le contraire est possible, de la manière présentée dans les formules (2.18) à (2.20).

$$X\varphi = \bot\ \mathcal{U}\ \varphi \tag{2.18}$$

$$F\varphi = \top\ \mathcal{U}\ \varphi \tag{2.19}$$

$$P\varphi = \top\ \mathcal{S}\ \varphi \tag{2.20}$$

L'axiomatique de $\mathcal{U}$ varie suivant les applications, nous présentons celle qui correspond à un flot temporel isomorphe à $(\mathbb{N}, <)$[8] (pour des raisons de lisibilité, nous utiliserons les autres modalités temporelles en tant qu'abréviations). Dans cette axiomatique, (2.21) et (2.22) sont l'équivalent de l'axiome (K) pour la modalité $\mathcal{U}$, lui assurant une certaine forme de distributivité sur l'implication logique. Les axiomes (2.23) et (2.24) relient la formule cible de la modalité dyadique avec les formules vraies dans l'intervalle de temps considéré, et assurent le maintien de la valeur de vérité du *until* entre l'instant présent et l'instant cible. L'axiome (2.25) assure une propriété de transitivité. Les axiomes suivants conditionnent les propriétés du flot de temps : (2.26) en assure la linéarité, ($\mathcal{U}$-D) le caractère discret, ($\mathcal{U}$-W) la cohérence de son ordonnancement et ($\mathcal{U}$-N) son extension dans le futur.

[8]Pour plus de détails sur cette axiomatique, sa complétude expressive et sa complétude sémantique, voir la définition 7.13, le lemme 7.14 et le théorème 7.20 de [BdV01], pages 430-433.

$$G(\varphi \to \psi) \to ((\rho\, \mathcal{U}\, \varphi) \to (\rho\, \mathcal{U}\, \psi)) \tag{2.21}$$
$$G(\varphi \to \psi) \to ((\varphi\, \mathcal{U}\, \rho) \to (\psi\, \mathcal{U}\, \rho)) \tag{2.22}$$
$$((\varphi\, \mathcal{U}\, \psi) \wedge \neg(\rho\, \mathcal{U}\, \psi)) \to (\varphi\, \mathcal{U}\, (\varphi \wedge \neg\rho)) \tag{2.23}$$
$$(\varphi\, \mathcal{U}\, \psi) \to ((\varphi \wedge (\varphi\, \mathcal{U}\, \psi))\, \mathcal{U}\, \psi) \tag{2.24}$$
$$(\varphi\, \mathcal{U}\, (\varphi \wedge (\varphi\, \mathcal{U}\, \psi))) \to (\varphi\, \mathcal{U}\, \psi) \tag{2.25}$$
$$(\varphi\, \mathcal{U}\, \psi) \wedge (\rho\, \mathcal{U}\, \sigma) \to \begin{pmatrix} ((\varphi \wedge \rho)\, \mathcal{U}\, (\psi \wedge \sigma)) \\ \vee((\varphi \wedge \rho)\, \mathcal{U}\, (\psi \wedge \rho)) \\ \vee((\varphi \wedge \rho)\, \mathcal{U}\, (\varphi \wedge \sigma)) \end{pmatrix} \tag{2.26}$$
$$F\top \to (\bot\, \mathcal{U}\, \top) \tag{$\mathcal{U}$-D}$$
$$F\varphi \to ((\neg\varphi)\, \mathcal{U}\, \varphi) \tag{$\mathcal{U}$-W}$$
$$F\top \tag{$\mathcal{U}$-N}$$

L'axiomatique de la modalité $\mathcal{S}$ est définie de manière symétrique à celle de $\mathcal{U}$ et augmentée de ($\mathcal{S}$-L) dans le cas d'un flot de temps isomorphe à $(\mathbb{N}, <)$, afin d'assurer l'existence d'un « point de départ » pour ce dernier. L'utilisation conjointe de $\mathcal{U}$ et $\mathcal{S}$ nécessite également l'introduction de l'axiome de conversion (2.27).

$$H\bot \vee (PH\bot) \tag{$\mathcal{S}$-L}$$
$$\varphi \wedge \psi\, \mathcal{U}\, \rho \to \psi\, \mathcal{U}\, (\rho \wedge (\psi\, \mathcal{S}\, \varphi)) \tag{2.27}$$

2.3.4.4 Principales logiques temporelles linéaires

De nombreuses logiques ont été proposées pour travailler sur les principes de bases de LTL. Suivant les exigences du domaine d'application, des ensembles de modalités différents sont choisis, amenant à des logiques plus ou moins expressives.

La logique K_t (ou *tense logic*) de Prior est la plus ancienne et probablement une des plus simples [Pri57]. Elle fait usage que des modalités G et H (et de leurs modalités duales), mais avec une axiomatique limitée à leurs axiomes de conversion (GP) et (HF) (et à l'axiome (K) appliqué à chaque modalité). Cette axiomatique est fortement complète pour les flots de temps bidirectionnels[9], mais n'assure ni leur transitivité ni leur caractère sériel. C'est donc une logique minimaliste pour traiter de la notion de temps.

La logique initialement proposée par Pnueli en 1977 est destinée à décrire les caractéristiques temporelles (invariants, équité, vivacité...) des programmes informatiques aussi bien séquentiels que concurrents ou cycliques [Pnu77]. N'ayant pas besoin de décrire le passé, on se contente de la modalité G et de son dual F. Manna et Pnueli étendent en 1981 cette logique (toujours en se limitant au futur) avec la modalité dyadique $\mathcal{U}$, mettant en évidence son pouvoir d'expression [MP81].

Pour les logiques temporelles linéaires traitant uniquement du futur, il est courant d'utiliser uniquement $\mathcal{U}$, ou encore la modalité affaiblie $\mathcal{U}^-$ assortie d'une modalité *neXt*.

La logique composée uniquement des modalités strictes $\mathcal{U}$ et $\mathcal{S}$ mérite qu'on y prête attention, à cause de sa grande expressivité : sur la classe particulière de flots de temps Dedekind-complets (incluant $(\mathbb{R}, <)$ et $(\mathbb{N}, <)$), cette logique a la même expressivité que la logique monadique du premier ordre (augmentée des relations binaires $=$ et $<$)[10]. Par contre, il a été plusieurs fois

[9] Pour plus de détails, voir pages 205-206 de [BdV01].
[10] Théorème 7.12, page 430 de [BdV01]

remarqué que si l'on pouvait conserver une complexité relativement faible (typiquement, la PSPACE-complétude) du problème de satisfaisabilité dans une logique temporelle utilisant $\mathcal{U}$, l'ajout d'une modalité $\mathcal{S}$ (plutôt que H) pour traiter du passé peut être handicapant du point de vue de la complexité calculatoire, comme le défend Mark Reynolds [Rey03].

Ces considérations nous amèneraient à choisir une logique de type $\mathcal{U}\ \mathcal{S}$ pour son expressivité, mais à craindre pour sa complexité. S'il s'avère que nous pouvons représenter les normes en matière de protection des données personnelles en évitant d'utiliser $\mathcal{S}$ (ce qui est envisageable puisque les normes servent a priori davantage à l'agent à raisonner sur ses actions futures qu'à analyser le passé), il pourra s'avérer avantageux de ne pas l'inclure dans le langage.

2.4 Logique déontique et politiques de sécurité

2.4.1 Logique déontique standard (SDL)

2.4.1.1 Présentation

La logique déontique standard (ou SDL pour *Standard Deontic Logic*) est également une logique modale normale, où la modalité universelle représente la notion d'obligation ou de devoir. Une version primitive de la logique déontique est introduite par Georg Henrik von Wright en 1951 [von51], mais SDL procède de plusieurs améliorations apportées par la communauté et par von Wright lui-même.

La notion d'obligation est représentée en SDL par la modalité universelle Ob, $Ob\ \varphi$ exprimant le caractère obligatoire d'une formule φ. Son dual est la permission ($Per\ \varphi$ signifiant que la négation de φ n'est pas obligatoire). Par le jeu des négations et de la dualité entre modalités on définit également le caractère interdit ($For\ \varphi$, la négation de φ est obligatoire), optionnel ($Opt\ \varphi$, φ n'est pas obligatoire) ou gratuit ($Gra\ \varphi$, φ n'est ni obligatoire ni interdit) d'une formule.

$$Per\ \varphi \stackrel{def}{=} \neg Ob \neg \varphi \tag{2.28}$$

$$For\ \varphi \stackrel{def}{=} Ob \neg \varphi \tag{2.29}$$

$$Opt\ \varphi \stackrel{def}{=} \neg Ob\ \varphi \tag{2.30}$$

$$Gra\ \varphi \stackrel{def}{=} \neg Ob\ \varphi \wedge \neg Ob \neg \varphi \tag{2.31}$$

2.4.1.2 Axiomatique

La modalité de base Ob procède d'une axiomatique de type KD. L'axiome (Ob-K) exprime la distributivité de l'obligation sur l'implication : si une implication logique est obligatoire, alors l'obligation de l'antécédent implique l'implication du conséquent. L'axiome (Ob-D), lui, incarne un principe de cohérence des formules déontiques : l'obligation implique la permission. Ainsi, l'obligation et l'interdiction simultanées d'une même formule (situation connue sous le nom de « dilemme moral ») conduisent à l'inconsistance logique du système.

$$Ob(\varphi \rightarrow \psi) \rightarrow (Ob\ \varphi \rightarrow Ob\ \psi) \tag{Ob-K}$$

$$Ob\ \varphi \rightarrow \neg Ob \neg \varphi \tag{Ob-D}$$

2.4.1.3 Sémantique

La logique déontique standard est interprétée sur des modèles de Kripke dont la relation d'accessibilité est sérielle (à cause de l'axiome (Ob-D)), c'est-à-dire que depuis tout monde il existe au moins un monde accessible. La figure 2.6 représente un exemple de modèle pour un système SDL. On peut notamment remarquer la capacité de la logique à représenter les violations : au monde w_1, $Ob\ \varphi$ est vraie mais φ est fausse. Par contre, dans tous les mondes accessibles depuis w_1 (à savoir w_2 et w_3), φ est vraie par définition de la sémantique. w_2 et w_3 sont donc des mondes respectant les obligations de w_1, ce qui n'implique pas que les mêmes obligations soient en vigueur dans ces mondes. Dans w_2 par exemple, φ est vraie (donc les obligations de w_1 sont bien respectées) mais interdite. Le monde w_2 est donc en violation avec ses propres obligations (il ne peut pas y avoir de mise en relation de w_2 avec lui-même), alors que w_3 respecte à la fois les obligations de w_1 et de w_2. Dans w_4, aucune obligation (ni interdiction) n'est vraie, toutes les formules y sont donc permises et optionnelles, donc gratuites par définition. Ce monde est donc en relation à la fois avec des mondes où φ est vraie et avec des mondes ou φ est fausse.

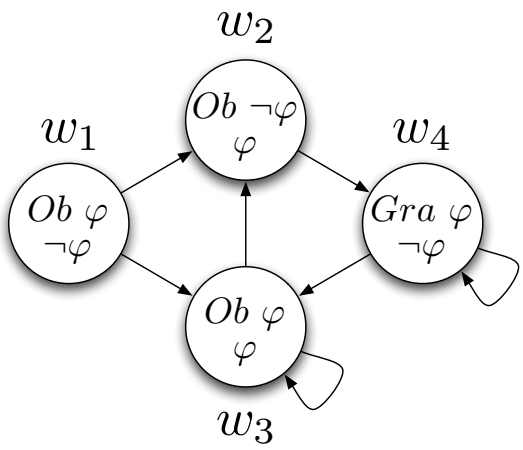

FIG. 2.6 – Exemple de modèle SDL

2.4.2 Politiques de sécurité

Malgré les paradoxes (décrits dans l'annexe A) dont souffre la logique déontique standard, celle-ci reste un outil privilégié à cause de sa correspondance avec les concepts philosophiques de l'obligation et du caractère intuitif et aisément représentable de sa sémantique. Toutefois, les handicaps formels la confinent souvent à un rôle de représentation et de formalisation, notamment dans le cadre des travaux sur les politiques de sécurité. SDL ou l'une de ses variantes sont en effet souvent utilisées pour représenter ces politiques, plus que pour raisonner conjointement sur les politiques et l'état réel du monde.

De nombreux auteurs [CJ94, Cup93a, Cup93b, GMP90, JS92, MM92, Ort96, CC97, CC98] ont étudié les apports et les limitations de la logique déontique pour le travail sur les politiques de sécurité. L'intérêt initial et fondamental est d'apporter un formalisme mathématique dans un domaine originellement régi par des réglementations en langage naturel, donc potentiellement informel. Les spécifications même d'une politique de sécurité peuvent ainsi être manipulées grâce à des outils mathématiques. Cette démarche de limitation des ambiguïtés du langage ne semble

pas être en soi un argument donnant un avantage particulier à la logique déontique par rapport à d'autres formalismes logiques. Par contre, Rodolphe Ortalo notamment remarque que la facilité avec laquelle les ensembles de formules déontiques peuvent être conceptualisés en s'appuyant sur la sémantique relationnelle, les diverses représentations qui peuvent en être faites, sont apparues comme un outil de communication formidable entre les différents acteurs de la sécurité d'un système [Ort96]. Selon Ortalo toujours, le fait d'exprimer les politiques de sécurité en logique déontique permet une flexibilité plus grande dans la conception du système lui-même, flexibilité qu'il oppose notamment aux exigences de bas niveau d'un système de type Bell-LaPadula [BL76].

Dans le cadre de la protection des données personnelles, l'ensemble des normes applicables dans un contexte donné peut être représenté de manière formelle et considéré comme une politique de sécurité pour un agent ou pour une organisation. La logique déontique peut alors être utilisée dans ce rôle de représentation, indépendamment des possibles applications de ladite politique. Néanmoins, un ensemble des normes contraignant le comportement des agents ne pourra constituer une politique acceptable que s'il est cohérent.

Il a par ailleurs été souvent remarqué que l'utilisation de la logique déontique pour la gestion des politiques nécessitait une meilleure adaptabilité du formalisme, afin de représenter l'évolution de ces politiques au cours de l'exécution du système. Le second principal reproche que l'on fait à la trop simpliste logique déontique est son incapacité à résoudre nativement les dilemmes, quand bien même on pourrait les représenter par différenciation des modalités, pour arriver à un ensemble de normes cohérent pouvant constituer une politique de sécurité. Nous allons maintenant voir comment l'introduction de logiques temporelles d'une part, et la mise en place de procédures de fusion d'autre part, ont été proposées pour répondre à ces deux défaillances identifiées de la logique déontique.

2.4.3 Logiques déontiques et logiques temporelles

La notion d'obligation ou d'interdiction est souvent associée à des contraintes temporelles. Dans le cadre de la protection des données personnelles, ce besoin d'associer les obligations à des notions de dates, de délais ou de précédence se fait particulièrement sentir. Si l'on prend l'exemple de la rétention de données personnelles, il faut généralement exprimer une obligation (de supprimer cette information) associée à une échéance. Dans le cas de l'information de l'utilisateur, on a une obligation sur la précédence temporelle de plusieurs formules.

De nombreuses propositions ont été faites pour associer aux notions de la logique déontique les possibilités offertes par les logiques temporelles, notamment linéaires. Une telle association permet de mieux caractériser la notion de violation d'une obligation, ou encore d'exprimer des obligations comportant des durées de validité ou des dates limites. La conception d'une logique temporelle déontique passe par le choix des modalités temporelles à inclure dans la logique déontique. On disposera donc dans la logique cible de la modalité d'obligation Ob, ainsi que d'un sous-ensemble des modalités temporelles X, G / G^-, H / H^-, $\mathcal{U}$ / $\mathcal{U}^-$ et $\mathcal{S}$ / $\mathcal{S}^-$. Par la suite, des questions de syntaxe se posent : autorise-t-on un libre mélange des modalités ou impose-t-on une structure figée ? Le libre mélange permet plus d'expressivité, mais des restrictions syntaxiques justifiées peuvent permettre de réduire la complexité de la logique. Au niveau axiomatique, le principal point de choix réside dans les axiomes de conversion. Si l'on considère par exemple qu'une obligation sur le maintien d'une formule est équivalente au maintien d'une obligation sur une formule (équation (2.32)), on dispose d'un outil pour transformer les chaînes de modalités (et éventuellement les réduire), mais on complexifie le problème de satisfaisabilité de la logique.

$$G\ Ob\ \varphi \leftrightarrow Ob\ G\ \varphi \tag{2.32}$$

Nous présentons ici quelques exemples représentatifs de la diversité des développements et des points de vue en matière de logiques déontiques temporelles.

2.4.3.1 Chellas 1980

En 1980, Brian F. Chellas propose l'introduction de modalités temporelles dans la logique déontique standard [Che80]. Dans la sémantique associée, les mondes représentent des exécutions complètes du système, depuis sa création. Ce choix sémantique est assez classique dans le domaine des modèles cognitifs. Il est notamment utilisé par Cohen et Levesque [CL90] pour la sémantique de la modalité de croyance. Chellas utilise donc un symbole de modélisation prenant en compte le modèle, le monde mais également l'instant t du temps. Ayant introduit une relation de compatibilité historique à l'instant t (deux mondes w et w' sont compatibles au temps t, soit $w \sim_t w'$ si et seulement s'ils sont indiscernables pour tout instant strictement antérieur), l'auteur utilise une relation d'accessibilité déontique $\mathcal{R}_t$ indicée (et conditionnée) par cet instant. En effet, un monde ne peut être déontiquement accessible depuis un autre à l'instant t que s'il lui est historiquement compatible au même instant, comme exprimé par la formule ($\mathcal{R}_t$).

Une logique de ce type est difficilement utilisable pour la conception d'un agent, car ce dernier devrait raisonner sur les concepts déontiques en prenant en compte l'intégralité de sa propre exécution, depuis son lancement, ainsi que les actions qu'il prévoit de mener dans le futur. Les actions passées de l'agent, qui peuvent être des erreurs de jugement ou des violations, ont alors une influence énorme sur ses prises de décisions, qui peuvent rapidement s'avérer difficiles.

$$w\mathcal{R}_t w' \rightarrow w \sim_t w' \qquad (\mathcal{R}_t)$$

La sémantique de la modalité déontique est donc restreinte en conséquence : la valeur de vérité de $Ob\ \varphi$ en un monde w et un instant t ne dépend que des mondes historiquement compatibles avec w jusqu'à l'instant t. Cette conception de la relation d'accessibilité soulève un problème d'ordre philosophique sur la nature de cette relation : doit-elle identifier tous les couples monde/instant qui respectent les obligations du monde/instant courant, ou bien seulement, parmi ces possibles, ceux qui représentent une opportunité pour l'agent ? En effet, dans le formalisme de Chellas, un monde/instant non atteignable par l'agent (au sens où l'historique permettant d'arriver à ce monde/instant est distinct de l'historique déjà parcouru par l'agent) ne peut être déontiquement accessible. Ce n'est pas le cas dans d'autres formalismes, où l'agent a ainsi la possibilité de mesurer ses « erreurs » passées au vu de ses obligations présentes. Par analogie, dans les modèles cognitifs de type BDI, un agent peut avoir un désir inatteignable. Celui-ci ne sera jamais transformé en intention, mais il est néanmoins représentable par l'agent.

Chellas introduit ensuite les modalités de nécessité et de possibilité historiques (que nous noterons respectivement, $\boxdot$ et $\diamondplus$) : une formule est historiquement nécessaire si et seulement si elle est vraie dans tout monde/instant historiquement compatible, une formule est historiquement possible si et seulement s'il existe un monde/instant historiquement compatible où est vraie. La nécessité historique est la modalité qui correspond à l'union des relations $\sim_t$, qui est une relation d'équivalence ($\boxdot$ a donc une axiomatique KT5). Les relations entre obligation et nécessité historique s'axiomatisent comme suit :

$$\boxdot\varphi \rightarrow Ob\ \varphi \tag{2.33}$$
$$Ob\ \varphi \rightarrow \diamondminus\varphi \tag{2.34}$$
$$\boxdot\varphi \leftrightarrow Ob\,\boxdot\varphi \tag{2.35}$$
$$\diamondminus\varphi \leftrightarrow Ob\diamondminus\varphi \tag{2.36}$$

L'axiome (2.33) est l'expression de la contrainte sémantique ($\mathcal{R}_t$), (2.34) découle de la sérialité de l'obligation. Les deux dernières formules expriment le fait que si une formule est historiquement nécessaire (resp. possible), alors cette nécessité (resp. possibilité) est obligatoire.

La logique proposée par Chellas nous paraît très limitée aujourd'hui. En effet, les concepts temporels sur lesquels elle s'appuie ne prennent pas en compte les développements de la logique temporelle linéaire (alors naissante), et notamment le raisonnement explicite sur le futur et le passé. Néanmoins, la notion de nécessité historique (ainsi que la notion dérivée de déterminisme historique) a souvent été reprise dans des systèmes logiques plus élaborés.

2.4.3.2 La logique NTL

La *Normative Temporal Logic* (NTL) est un exemple de logique temporelle et déontique moderne, présentée en 2007 par Thomas Ågotnes, Wiebe van der Hoek, Juan A. Rodríguez-Aguilar, Carles Sierra et Michael Wooldridge [ÅvdHR+07]. La logique NTL n'est pas fondée sur la logique temporelle linéaire, mais sur la logique temporelle arborescente (logique CTL). Les modalités temporelles Xet $\mathcal{U}$ utilisées dans NTL[11] s'entendent donc pour l'ensemble des futurs possibles. En logique CTL, cette situation correspond à l'utilisation de la modalité préfixe A.

L'opérateur d'obligation de la logique NTL est contextualisé en fonction du système normatif, plusieurs d'entre eux pouvant être considérés simultanément. Ainsi, Ob_η est l'opérateur du système normatif η et est complètement distinct d'un opérateur $Ob_{\eta'}$. Cette logique permet donc de représenter les dilemmes, à condition qu'ils soient répartis sur plusieurs systèmes normatifs. Les modalités d'obligation étant issues de SDL, elles héritent des paradoxes décrits dans l'annexe A (bien que leur impact ne soit pas caractérisé). Le symbole η représente également une relation d'accessibilité spécifiant quelles transitions entre mondes sont interdites ou autorisées par le système normatif. Si l'on considère l'intersection de η avec une relation d'accessibilité $\mathcal{R}$ définie par les contraintes du système, on peut déterminer l'ensemble des transitions qu'un agent peut emprunter en respectant les normes d'un système donné.

Concernant les relations entre logique déontique et logique temporelle, l'imbrication des deux est strictement fixée : un opérateur déontique s'applique sur une modalité temporelle. On ne rencontre pas de modalité temporelle sans qu'une modalité déontique y soit directement appliquée, et une modalité déontique s'applique toujours directement à une modalité temporelle. Cette restriction syntaxique permet de limiter la complexité calculatoire de la logique à la NP-complétude.

La logique NTL permet la comparaison de systèmes normatifs sur le plan de leur rigueur ou de leur permissivité, en raisonnant sur les inclusions entre relations d'accessibilité. En travaillant sur les intersections entre ces relations, un agent peut choisir de respecter les normes de plusieurs systèmes simultanément, mais la logique ne garantit pas alors qu'il lui restera des options possibles (en d'autres termes, l'intersection $\eta_1 \cap \eta_2 \cap \mathcal{R}$ peut être vide). Cette logique

[11] Les notations utilisées dans la communication originale sont différentes, mais pour des raisons de cohérence nous nous reposerons sur les formalismes déjà introduits.

peut donc être considérée comme un premier pas vers une gestion des dilemmes entre systèmes normatifs, en permettant de les représenter mais pas toujours de les arbitrer.

Cette proposition soulève la question de l'utilisation d'une logique du temps arborescente. Il faut alors garder à l'esprit que ce sont des normes que nous souhaitons représenter, et que ces normes traduisent des réglementations qui disent ce qui doit être fait ou pas. Une représentation arborescente du temps permettrait d'analyser le comportement possible d'un agent en décrivant ses futurs possibles (certains respectant une norme, d'autres la violant), mais la norme en elle-même ne traite que de l'histoire effective, celle qui a eu et qui aura lieu, à l'exclusion de toute autre. Le fait que l'agent raisonne prioritairement sur ses obligations nous permet donc de nous limiter à un temps linéaire, plutôt que d'utiliser un formalisme plus adapté à la description et à l'analyse externe d'un système. On notera d'autre part que la logique NTL telle qu'elle est formalisée ici permet de représenter les conflits entre normes, mais pas de les résoudre : l'agent peut se retrouver dans une situation où il ne sait pas ce qu'il doit faire.

2.4.3.3 Åqvist 2004

Lennart Åqvist présente en 2004 une étude détaillée des méthodes usitées pour introduire le temps dans une logique déontique [Åqv04]. Il distingue notamment les approches utilisant une **indexation par le temps**, à la manière de Chellas par exemple [Che80], des approches utilisant les **opérateurs temporels** comme celle de Ågotnes *et al* [ÅvdHR+07]. Åqvist caractérise les logiques modales **multidimensionnelles** comme celles qui, à la manière de celle de Chellas, s'interprètent sur une structure de coordonnées à deux dimensions (les structures de type $\mathcal{T} \times \mathcal{W}$) : les instants et les mondes (ou histoires).

Afin de lier langage et sémantique et d'utiliser dans les formules les notions d'instant et de monde, Åqvist utilise dans le formalisme qu'il met en place des *constantes systématiques de cadre* représentant l'une ou l'autre coordonnée. Sont également introduites des modalités qui permettent de raisonner, en un monde/instant, sur les formules vraies à tout instant du même monde, en tout monde au même instant, en tout monde à tout instant, ou en un monde donné à un instant donné. Ces connecteurs sont beaucoup plus expressifs que l'utilisations des modalités de la logique temporelle.

Åqvist introduit ensuite dans ce système des modalités dyadiques d'obligations conditionnelles, pour lesquelles $Ob_\psi\varphi$ (souvent noté $Ob(\varphi/\psi)$ par ailleurs) signifie « si ψ est vraie, alors il est obligatoire que φ ».

De la proposition d'Åqvist nous retiendrons principalement le raisonnement bi-dimensionnel entre axes déontique et temporels, ainsi que la notion d'instant plus aisément manipulable que les exécutions de Chellas. Les principes introduits ici nous permettront de raisonner sur des dates (de manière absolue ou relative), concept récurrent dans le domaine de la protection des données personnelles.

2.4.3.4 Balbiani 2005

Philippe Balbiani propose en 2005 une logique mêlant les notions de nécessité, d'obligation, de temps et d'action [Bal05]. Le temps est exprimé à l'aide d'un opérateur *neXt* et de son converse. L'auteur discute également de l'introduction de modalités temporelles plus expressives comme F, P, H et G, ainsi que du surcoût qu'elles induisent en termes de complexité calculatoire (passage de NP-complet à PSPACE-difficile, en admettant la conjecture NP $\neq$ PSPACE). La notion de temps semble donc ici réduite à sa plus simple expression. La logique déontique utilisée est SDL et la logique dynamique est une variante de la classique *Propositional Dynamic Logic*

permettant de raisonner sur la durée des actions. La sémantique de cette logique, fondée sur des histoires construites à l'aide d'une fonction reliant la date de début d'un processus à sa date d'arrêt n'est que faiblement apparentée aux structures de Kripke classiques.

Balbiani aborde le problème des obligations émises par des autorités indépendantes et propose de forcer l'absence de conflits entre celles-ci grâce à un axiome (formule (2.37)) liant les différentes modalités déontiques associées. Cette formule dit que l'existence d'obligations implique la permission par toute autorité de la conjonction de toutes les formules sur lesquelles portent des obligations. L'équivalent sémantique de cet axiome est la contrainte suivant laquelle l'intersection de toutes les relations d'accessibilité déontiques doit être sérielle. De plus, l'auteur inclut toute relation déontique dans la relation aléthique du modèle, considérant que toute obligation doit nécessairement pouvoir être respectée.

$$\bigwedge_{i=1}^{n} Ob_i\ \varphi_i \rightarrow Per_{n+1} \bigwedge_{i=1}^{n} \varphi_i \tag{2.37}$$

On trouve dans cet axiome un outil essentiel pour la nécessaire gestion des conflits, qui peuvent rendre un ensemble de normes incohérent. Nous pourrons l'utiliser dans une logique déontique multimodale pour stigmatiser la présence ou l'absence de conflits entre différentes sources de normes.

2.4.3.5 Obligations avec échéances

On peut considérer qu'exprimer des obligations en logique déontique reste vain si l'on n'attache pas des contraintes temporelles à chaque obligation. Pour illustrer cette idée, prenons l'exemple d'un agent qui aurait l'obligation d'effacer une certaine donnée personnelle, mais sans contrainte de temps. Il y a deux manières de considérer cette situation : ou bien c'est une obligation immédiate et l'agent est en violation s'il n'obéit pas à l'instant présent (ce qui est un point de vue rarement tenable), ou bien c'est une obligation de le faire « un jour ». Dans ce second cas, l'agent ne se trouve jamais en situation de violation, car il peut toujours prétendre qu'il prévoit de détruire les données à une date future. Ainsi, son comportement ne peut jamais lui être reproché. Ce problème est d'autant plus présent en protection des données personnelles que les réglementations font systématiquement référence à des contraintes de temps.

Pour remédier à ce problème et représenter les obligations de manière plus réaliste, plusieurs auteurs ont fait des propositions d'opérateurs mixtes, déontiques et temporels, d'obligations avec échéances. Nous proposons d'examiner rapidement trois de ces propositions, partant d'une idée fondamentale identique : il faut utiliser l'opérateur temporel $\mathcal{U}$ (ou un équivalent) de manière à maintenir une obligation sur une formule temporelle jusqu'à ce qu'une échéance arrive ou que l'obligation soit respectée. Si l'obligation n'est toujours pas respectée à l'échéance, on considère alors qu'il y a une violation.

Frank Dignum, Jan Broersen, Virginia Dignum et John-Jules Meyer proposent un modèle d'obligation avec échéances qui n'est pas fondé sur une logique déontique [DBDM04]. En effet, la notion d'obligation avec échéance est définie parallèlement avec celle de violation, ces deux concepts étant liés par des contraintes temporelles pouvant être exprimées en LTL. Le principal défaut de cet opérateur semble être qu'une obligation avec échéance peut être dérivée dès que les conditions sont réunies pour qu'elle soit respectée, sans qu'il y ait eu une quelconque initiative d'une quelconque autorité pour exprimer une obligation. Ceci est dû au fait que l'obligation n'est pas un opérateur primaire dans le système utilisé par Dignum *et al.*

Robert Demolombe utilise une logique dans laquelle l'obligation avec échéance est définie à partir d'un opérateur de type *stit* (pour *See To It That*, faire en sorte que), capable de représenter

les obligations, et de modalités temporelles classiques [Dem05]. En transposant l'opérateur de Demolombe dans une logique déontique et temporelle, on constate qu'il ne souffre pas du même problème d'apparition « automatique » des obligations que celui de Dignum *et al.*

Julien Brunel, Jean-Paul Bodeveix et Mamoun Filali étendent un formalisme fondé sur un produit de SDL et de LTL, en utilisant des opérateurs indexés par des durées (plutôt que par des dates dans les autres formalismes) [BBF06]. Brunel *et al* utilisent alors les opérateurs F et $\mathcal{U}$ ainsi modifiés pour maintenir une obligation pendant une période donnée et caractériser sa violation. Ce formalisme permet de raisonner explicitement sur des durées, ce qui autorise une grande expressivité, mais qui a l'inconvénient de nécessiter l'introduction dans le langage de quantificateurs explicites sur ces durées. L'opérateur d'obligation avec échéance correspondant ne peut donc être exprimé directement dans une logique déontique et temporelle classique.

Afin de traiter efficacement les obligations avec échéances dans le cadre de protection des données personnelles, nous devrons, lorsque nous disposerons d'un formalisme logique, déterminer quelles sont les caractéristiques qu'un tel opérateur doit avoir dans notre contexte. Les trois opérateurs existants seront alors évalués en détail au vu de ces exigences.

2.4.4 Fusion de normes et gestion de l'inconsistance

Dans un cadre normatif ouvert comme celui qui nous intéresse, les normes proviennent de sources variées et indépendantes. Ces différences peuvent par exemple être dues au fait que l'agent joue plusieurs rôles simultanément dans un même système normatif ou qu'il appartient à plusieurs systèmes normatifs distincts. En conséquence, ces normes peuvent être incompatibles entre elles. Dans le cas général, l'ensemble des formules logiques constituant les normes est logiquement inconsistant, c'est-à-dire qu'il contient des contradictions permettant, en vertu du principe *ex contradictione sequitur quodlibet* (2.38), de déduire n'importe quelle formule. De tels ensembles de normes ne peuvent donc pas constituer une recommandation saine pour l'agent[12]. Il est par conséquent nécessaire d'établir des stratégies pour déduire d'un ensemble inconsistant de normes, un modèle de comportement cohérent. C'est l'un des points de travail importants identifiés en 1996 par Rodolphe Ortalo [Ort96] : amener les systèmes artificiels à gérer l'inconsistance aussi bien que les systèmes humains, qui s'en accomodent dans le cadre des politiques de sécurité exprimées en langage naturel.

$$\varphi, \neg\varphi \vdash \psi \tag{2.38}$$

L'outil généralement avancé pour ce faire est la **fusion de normes**. Celle-ci consiste à construire un ensemble de normes consistant à partir d'un ensemble inconsistant, de plusieurs ensembles inconsistants ou de plusieurs ensembles consistants inconsistants entre eux. Pour cela, on choisit de privilégier certaines normes par rapport à d'autres : certaines formules seront conservées dans l'ensemble final, d'autres seront écartées. Lors de cette étape on utilise une relation de préférence, qui peut être définie de diverses manières. Cette préférence peut porter sur les formules soumises à l'obligation (qu'elles soient des états ou des actions), sur les autorités qui émettent les formules (comme par exemple dans les méthodes proposées par Laurence Cholvy et Frédéric Cuppens [CC98]), sur les rôles joués par l'agent (comme proposé également par Cholvy et Cuppens [CC97]) sur les mondes possibles dans lesquelles ces formules sont vraies (comme dans la sémantique proposée en 2002 par Hansson pour la logique déontique [Han02]).

[12] Dans nombre de systèmes de logique déontique, comme celui de Balbiani [Bal05], ce problème de contradiction entre systèmes normatifs est évacué par construction : les conflits ne sont pas permis, il n'est pas possible de les représenter ni de raisonner dessus.

Cette sélection de normes peut également être fondées sur la spécificité des formules soumises à l'obligation (au sens où φ est plus spécifique que ψ lorsque $\varphi \rightarrow \psi$), comme en logique déontique défaisable (et notamment dans le système élaboré par L. Thorne McCarty et présenté par van der Torre [vdT94]).

Le principe de la fusion de normes peut être comparé aux travaux concernant le raisonnement sur des bases de connaissances incohérentes, comme ceux de Gerhard K. Kraetzschmar [Kra97]. Il s'en distingue (outre par la sémantique normative) en ceci que certaines normes sont supprimées ou désactivées pour atteindre un ensemble cohérent, alors que des propositions comme celles de Julia Rose Galliers [Gal90], par exemple, mettent en place un raisonnement sur des ensembles comportant des contradictions. Nous serons amenés par la suite à détailler plus avant quelques mécanismes pour la résolution des inconsistances normatives. Nous verrons que les propositions existantes dans le domaine de la fusion de normes avec préférences en logique déontique présentent l'inconvénient de considérer trop facilement qu'un ensemble de normes est inconsistant, et donc intenable pour un agent.

2.4.5 Formalismes alternatifs

2.4.5.1 Obligations, états et actions

On pourra remarquer que les modalités d'obligation de la logique déontique portent sur des formules, représentant une assertion sur l'état du monde qui devrait être vraie. Dans sa proposition originale [von51], von Wright faisait porter les obligations sur des actions. La convergence des formalismes effectuée par la communauté de la logique philosophique pour aboutir à SDL a abandonné cette idée, pour utiliser des formules plus abstraites et susceptibles de représenter n'importe quel concept. Par la suite, de nombreux auteurs sont revenus sur cette distinction entre « obligation d'être » et « obligation de faire », fondamentale d'un point de vue philosophique. Cette réflexion, régulièrement réactualisée (notamment par Ortalo [Ort96]), a conduit à des formalismes mêlant logique déontique standard et logique dynamique (comme proposé par exemple par Philippe Balbiani [Bal05]), ou encore à des opérateurs spécialisés mettant en avant la responsabilité d'un agent dans l'avénement de la valeur de vérité d'une formule. C'est le cas des divers opérateurs *stit*, comme l'opérateur E_i de Kanger [Kan71], l'*achievement stit* élaboré par Nuel Belnap et Michael Perloff [BP88] ou le *deliberative stit* proposé par Franz von Kutschera [von86] et John F. Horty [Hor89], qui peuvent éventuellement servir à représenter des obligations [HB95]. Ces formalismes sont difficilement acceptable dans le cas de figure qui nous intéresse, car ils interdisent la représentation d'obligations que l'agent n'a pas la capacité de respecter. Or ce type d'obligation peut être présent dans les réglementations sur la protection des données personnelles (par exemple à cause de conflits entre plusieurs réglementations) et doit pouvoir être considéré par l'agent.

2.4.5.2 Représentations alternatives du temps

La représentation du temps et de la précédence des états (ou des actions) a également nombre de représentations alternatives. James F. Allen, par exemple, introduit les opérateurs dyadiques *before*, *meets*, *overlaps*... pour raisonner sur les relations entre intervalles de temps et ainsi manipuler des événements, des états (ou *fluents*) et des processus [All84]. Les approches fondées sur la réification des propositions temporelles (comme celle d'Allen ou celle de Drew V. McDermott [McD82]) utilisent des prédicats comme *holds* ou *occurs*, qui caractérisent l'apparition d'une formule à un instant donné ou pendant un intervalle donné. Le calcul des événements de Robert A. Kowalski et Marek J. Sergot [KS86] ou le calcul des situations de John M. McCarthy et Patrick

J. Hayes [MH69] sont deux autres exemples de formalismes, fondés sur la logique du premier ordre, pour représenter les caractéristiques temporelles et les propriétés des événements.

Ce type de représentation du temps peut paraître pratique, à cause de la séparation entre les termes propositionnels et les considérations purement temporelles. Elle est une étape intéressante pour le passage au premier ordre, cependant, à un niveau purement conceptuel, certains auteurs trouvent artificielle cette séparation sémantique entre les événements et les notions temporelles.

2.4.5.3 Paraconsistance

Le problème de représentation du caractère conflictuel des ensembles de normes, s'il peut être résolu très simplement par l'indexation des modalités d'obligation, peut également être traité à la source en acceptant l'existence de conflits au sein du système. Les logiques paraconsistantes, introduites par Graham Priest et Koji Tanaka [PT04], et notamment les travaux de Jean-Yves Béziau [Bé02, Bé06], permettent de représenter des systèmes logiques dans lesquels il existe des contradictions, exprimées à l'aide d'opérateurs de négation spécifiques, et qui pourtant ne sont pas triviaux (ou encore non explosifs, c'est-à-dire qu'ils ne permettent pas de dériver n'importe quelle formule). Antoine Seilles, notamment, a étudié l'intérêt de ce formalisme dans le cadre de la représentation informatique des textes de loi et des disputes judiciaires [Sei07]. Néanmoins, la paraconsistance, si elle permet d'éviter l'écueil du *ex contradictione sequitur quodlibet*, ne fournit pas nativement de méthode de résolution des conflits exprimés.

2.4.5.4 Défaisabilité

Enfin, face aux stratégies de fusion de normes (par voie logique ou par voie procédurale), introduites pour arbitrer les conflits au sein d'un ensemble de normes, on peut opposer une adaptation déontique de la logique défaisable, dont Donald Nute a été un des principaux contributeurs [CNV97, Nut97, Nut01]. La logique défaisable permet de raisonner sur des règles logiques strictes (représentant les contraintes naturelles et inviolables d'un système), des règles logiques susceptibles d'être défaites (représentant des règles générales) et des défaiseurs capables de supplanter ces règles (représentant des exceptions aux règles générales). Cependant, si les mécanismes de défaisabilité semblent s'adapter de manière assez satisfaisante à certains types d'arbitrages, il convient d'examiner au cas par cas les tenants et les aboutissants de nature philosophique de l'utilisation de la défaisabilité. En effet, l'introduction de la défaisabilité en logique déontique permet de raisonner sur la notion d'exception à une norme (comme détaillé par van der Torre par exemple [vdT94]), mais cela ne constitue qu'un seul cas d'espèce en matière de conflit déontique. La défaisabilité semble en effet un processus peu adapté pour effectuer des arbitrages entre des systèmes normatifs différents, ou pour raisonner sur des préférences d'une manière très générale. Si le mécanisme de défaisabilité est intéressant, constituant d'ailleurs un cas particulier de la fusion avec préférences, c'est donc principalement pour des raisons sémantiques qu'il ne nous semble pas judicieux de l'appliquer à des ensembles de réglementations hétérogènes.

2.5 Synthèse et discussion

Les travaux existants formulent des propositions utiles au raisonnement en termes de protection des données personnelles. Les développements en matière de modèles cognitifs fournissent des outils théoriques et pratiques prenant en charge certains aspects du raisonnement de l'agent, notamment en matière des buts qu'il se donne (ou qu'on lui donne) à suivre. Les travaux dans le domaine des systèmes normatifs permettent d'étendre ces modèles cognitifs et de situer les

agents dans des organisations régies par des règles exprimées à l'aide de formalismes logiques riches et expressifs. Il est possible de travailler avec des normes prenant en compte les relations entre obligations et temps. Les conflits entre normes peuvent également être détectés et gérés de manière plus ou moins satisfaisante pour permettre à l'agent d'en tirer des directives applicables.

Il est possible d'adapter ces outils à des normes régissant le traitement des données personnelles, mais aucun des travaux existants ne prend en compte cette spécialisation. C'est une telle adaptation que nous proposons ici, même si cette dernière n'est pas immédiate. En effet, suivant le sens que l'on donne aux normes pour l'agent, la notion de conflit, par exemple, peut être discutée. Notre objectif est de construire un agent qui respecte au mieux ses obligations, même si ces dernières comportent des incohérences. Cet agent ne fait donc pas entrer en ligne de compte ses intérêts stratégiques propres dans sa gestion des informations personnelles. La thèse que nous défendons sous-tend que l'utilisation d'un tel agent pourrait garantir le meilleur respect possible des ensembles de normes applicables, ou en tout cas de caractériser ce respect (ou non-respect). Dans cette perspective, la notion de permission, en particulier, doit être considérée avec une attention toute particuière. En effet, une permission valide dans un système normatif particulier ne devrait pas pouvoir servir de « prétexte » à l'agent pour violer une autre norme jugée moins importante, tant qu'il a une possibilité de respecter cette obligation. C'est donc l'existence d'obligations ou d'interdictions qu'il faut considérer en priorité en cas de conflit logique.

Nous allons maintenant définir le modèle de cet agent conscient des problématiques de vie privée, en concevant son système de raisonnement logique (fondé sur les avancées techniques que nous avons identifiées comme les plus adaptées) en prenant en compte ce rôle particulier de la permission.

Deuxième partie

Outils théoriques pour la protection de la vie privée

Chapitre 3

L'agent *PAw*, un agent conscient de la protection des données personnelles

Les propositions en matière de protection des données personnelles mettent en exergue un certain nombre de principes et de bonnes pratiques, mais il nous semble que deux aspects, principalement, demandent à être traités de manière spécifique. Premièrement, les démarches d'automatisation de la gestion sont assez rares et encore embryonnaires, alors que les travaux en matière de personnalisation et d'organisation des applications distribuées autour de l'utilisateur en montrent le besoin, comme le pointe par exemple Alfred Kobsa [Kob07]. D'autre part, la protection étendue des données est absente de nombre de solutions existantes, même si certaines, comme par exemple celles utilisant les *sticky policies* [KS02, CPB03], en montrent le souci.

En réponse à ce constat, la thèse que nous défendons apporte une contribution à cet état de l'art en proposant des outils formels et techniques pour un agent artificiel assistant (l'agent PAw, ou *Privacy-Aware Agent*), lié à son agent humain utilisateur et chargé d'interfacer ses interactions avec des applications de service tout en prenant en charge la protection de ses données personnelles.

3.1 Systèmes multi-agents centrés utilisateur

La recherche sur les systèmes multi-agents a évolué au cours des dernières années pour mettre un accent particulier sur le rôle de l'utilisateur dans la compréhension et la conception du système. La notion, essentiellement applicative, de systèmes multi-agents centrés utilisateurs [MAG06] désigne notamment les architectures dans lesquelles les utilisateurs humains développent une interaction privilégiée avec certains agents artificiels du système, qui leur servent d'interface. Ce type de relation a parfois été modélisé à l'aide de systèmes multi-agents hybrides, comportant des agents artificiels et des agents humains considérés au même niveau d'abstraction [Chi04]. Parmi les travaux traitant de la place de l'utilisateur humain dans le système multi-agent, on peut notamment citer l'analyse d'Yvon Haradji, Nils Ferrand et Haijiang Li [HFL04] ou encore, sur des aspects plus techniques, les architectures multi-agents centrées sur l'utilisateur proposées par Stéphanie Riché, Gavin Brebner et Mickey Gittler [RBG02, RB03].

Dans le cadre de notre travail, nous entendons le terme de « centrage utilisateur » de la manière suivante :

Définition 3.1 (Centrage utilisateur)**.** Le centrage utilisateur est un parti pris de concevoir et de considérer un système, ses fonctionnalités et ses caractéristiques en fonction des seuls bénéfices, risques et diverses répercussions sur un utilisateur humain de ce système.

Le point de vue de cet utilisateur prévaut donc, dans cette conception, sur celui des autres intervenants. Dans le cadre d'un agent utilisateur chargé de traiter et de protéger des données personnelles au sein d'un système multi-agent, cela signifie entre autres que la protection des données appartenant à l'utilisateur humain propriétaire de l'agent est considérée comme plus importante que les performances d'une application ou que l'utilité (notamment stratégique ou commerciale) perçue par d'autres agents.

Le centrage utilisateur tel que nous l'entendons dans nos travaux constitue donc un parti pris très fort, qui nous fait observer un système depuis un unique point de vue (qui peut ne pas être celui de tous les acteurs du système). Les intérêts de ce choix sont principalement de deux ordres. D'une part, il permet d'aborder de manière poussée des questions de libertés individuelles qui entrent en conflit avec les intérêts de la majorité des agents d'un système et qui seraient donc menacées d'être mises de côté dans le cadre d'une approche plus globale. D'autre part, il permet en quelque sorte de simuler l'égoïsme d'un agent du système (au sens de la prévalence des buts individuels sur les intérêts collectifs), qui ne peut raisonner efficacement qu'en se fondant sur ses informations et intérêts propres et/ou en considérant que les intentions des autres agents sont potentiellement dangereuses pour lui.

Bien entendu, c'est également un point de vue partial et partiel qui reste insuffisant pour la construction d'un modèle global. Il fait écho à d'autres points de vue défendant d'autres agents et d'autres intérêts. Néanmoins les développements sur ces autres points de vue (performance et sécurité des services, personnalisation, prospection commerciale personnalisée...) nous semblent pour l'instant largement plus présents dans le débat scientifique, technique et social que les analyses détaillées du centrage utilisateur tel que nous l'entendons.

3.2 Gestion des profils d'utilisateurs

Le centrage sur l'utilisateur d'une application, qu'elle soit distribuée ou centralisée, fondée ou pas sur un paradigme multi-agent, appelle souvent une personnalisation de l'expérience applicative de l'utilisateur. Cette personnalisation s'opère par une adaptation des outils informatiques utilisant des données décrivant les préférences ou les usages de l'utilisateur ciblé. Ces données peuvent être renseignées par l'utilisateur humain lui-même ou bien collectées directement par l'application. Dans tous les cas, le problème de la protection de ces données se pose. Même si la personnalisation en elle-même ne rentre pas dans le cadre de nos travaux, il convient de s'intéresser au champ scientifique de la personnalisation si l'on veut traiter conjointement de protection des données personnelles et de centrage utilisateur.

3.2.1 Modèles existants

Nous proposons tout d'abord un aperçu du type de propositions existant dans le domaine, en analysant comment ces propositions traitent le problème de la protection des données personnelles.

3.2.1.1 Personnalisation et modèles d'utilisateurs

De nombreux travaux existent dans le domaine de la modélisation de l'utilisateur sous forme d'un profil géré par voie informatique. Dans le domaine de la recherche d'information

[ACI04, GSCM07] par exemple, l'intérêt est de personnaliser l'information fournie à l'utilisateur en fonction de ses préférences, qu'elles soient exprimées ou inférées. Les profils peuvent également servir à générer des propositions commerciales ciblées, à personnaliser l'apparence d'un site web, à construire des réseaux sociaux...

En la matière, comme le soulignent Ioannidis et Koutrika [IK05], il convient de distinguer d'un point de vue terminologique le **modèle d'utilisateur** du **profil d'utilisateur**. Le modèle d'utilisateur est l'ensemble des dimensions suivant lesquelles l'utilisateur va être représenté, alors que le profil est une instance de modèle, la représentation d'un utilisateur en particulier.

La constitution de ces profils d'utilisateurs, les méthodes de collecte ou de production des données, l'évaluation de la qualité des informations du profil sont un domaine de recherche à part entière, qui concerne les spécialistes de la personnalisation [ACI04, MLd03, CDA99]. Ici nous nous intéressons aux profils d'utilisateurs avec un point de vue extérieur, afin d'en analyser les besoins et l'existant en termes de protection du caractère privé des données.

3.2.1.2 Le modèle générique de Bouzeghoub et Kostadinov

Amato et Straccia [AS99] ont proposé en 1999 un état des lieux des propositions en matières de modèles d'utilisateurs et en ont dérivé un modèle d'utilisateur synthétique. Bouzeghoub et Kostadinov [BK05] ont repris et enrichi l'analyse et le modèle qui en découle. Ces modèles restent fortement teintés par l'orientation des auteurs vers les applications de recherche d'information, malgré un souci exprimé de généricité. Ils décrivent l'utilisateur suivant les six dimensions suivantes :

- **Données personnelles :** représente les informations d'identification et les caractéristiques personnelles de l'utilisateur. Dans [AS99], les auteurs s'appuient pour cet axe sur le « schéma utilisateur » de P3P [Wor06] ;
- **Domaines d'intérêt :** constitue une spécification (en matière de contenu, de source, de structure...) des informations susceptibles d'intéresser l'utilisateur ;
- **Préférences de livraison :** peut désigner, suivant les applications, les informations nécessaires à une livraison physique (adresse, numéro de téléphone, dates et horaires de disponibilité...) ou les modes de livraison numérique (formatage des informations, préférences d'interface, de fréquence...) ;
- **Historique :** « journalise » les interactions de l'utilisateur avec le système, à partir desquelles peuvent également être déduites des informations se rapportant aux cinq autres dimensions sémantiques (notamment les domaines d'intérêt) ;
- **Qualité :** contient les informations sur la qualité de service désirée par l'utilisateur, et sur les évaluations faites par lui de ladite qualité ;
- **Sécurité :** spécifie le caractère privé et le niveau de sécurité des éléments du profil. Dans [AS99] elle est réduite au minimum, à savoir une liste d'hôtes autorisés à accéder au profil. Dans d'autres modèles [RBG02], ce peut être par exemple un ensemble de listes de contrôle d'accès (*Access Control Lists*).

Au sein du profil, les données élémentaires peuvent être représentées de diverses manières, mais dans [AS99] comme dans [BK05] les auteurs retiennent que les structures les plus souvent utilisées sont les **tuples** (souvent des paires attributs-valeurs ou des triplets RDF), les **règles** logiques ou sémantiques ou les **journaux** d'historiques. Ces données élémentaires sont elles-mêmes structurées pour former le profil. Elle sont organisées, suivant les cas, en **arbres**, en **vecteurs**, en **graphes** ou de manière linéaire. À priori, la représentation et la structuration des données peut être une contrainte pour leur protection, déterminant ainsi les mécanismes de sécurité potentiellement utilisables pour garantir la protection des données personnelles contenues dans le

profil.

3.2.1.3 La sécurité dans les modèles d'utilisateurs

Il est possible de classer les différents modèles d'utilisateurs dans l'ordre croissant de l'importance qu'ils accordent à la sécurité des informations et à la protection des données personnelles.

Modèles sans protection intégrée

La plupart des modèles d'utilisateurs ont une conception très clairement orienté vers des impératifs de richesse et de pertinence de l'information contenue, mais sont par ailleurs fort dépourvus en matière de sécurité.

De nombreux modèles ne comprennent absolument aucun mécanisme de protection. C'est le cas des travaux de Tamine et Bahsoun [TB06], NGuyen *et al* [NDB05], DesJardins *et al* [dW05, dEW06], Shearin et Lieberman [SL01], Schiaffino et Amandi [SA00], Li et Yao [LY02]. Tous ces modèles manipulent des données personnelles potentiellement sensibles, mais aucun d'eux ne met en œuvre de dispositif de protection de ces données.

Modèles gérant une représentation des informations de sécurité

D'autres modèles se situent à un niveau légèrement supérieur, en intégrant (parfois même en étendant) les outils de la spécification P3P [Wor06]. On retrouve dans cette catégorie les modèles d'Amato et Straccia [AS99], Bouzeghoub et Kostadinov [BK05] et Cingil *et al* [CDA99]. Dans ce dernier modèle notamment, Cingil *et al* développent des entités mandataires P3P pour assurer, de manière relativement indépendante du profil, la communication sur les politiques de protection des données personnelles entre client et serveur. Ces travaux sont néanmoins limités par la nature même de P3P, qui ne s'intéresse qu'aux deux premiers axes de la protection des données personnelles, à savoir le consentement et l'information de l'utilisateur.

Ces derniers modèles souffrent d'un défaut significatif. En effet, ils intègrent les informations de protection du profil au même niveau que les éléments du profil eux-mêmes. C'est assez comparable à ce que serait un système de fichiers qui représenterait les informations de contrôle d'accès dans un simple fichier : pour connaître ses droits à accéder à une information, un processus doit alors accéder sans restriction particulière à une information de même niveau. Il est donc tout-à-fait nécessaire de représenter les méta-informations de sécurité différemment des informations du profil. Les travaux suivants respectent ce principe.

Modèles mettant en œuvre une protection réelle des données

Peu de modèles montrent à la fois un réel souci de la protection des données personnelles et une représentation saine des informations de sécurité correspondantes. Nous mettrons plus particulièrement les propositions de Stéphanie Riché et Gavin Brebner [RBG02] et le framework ePerson [DRB+03] de Dickinson *et al.*

Le modèle de Riché et Brebner [RBG02] se fonde délibérément sur une philosophie de centrage sur l'utilisateur, en prenant compte dès le départ des contraintes de protection des données personnelles. Les méta-informations de sécurité, de niveau de confidentialité et de sensibilité à la réplication sont représentées dans un système de politiques arborescent stocké de manière distribuée sur les différents terminaux de l'utilisateur, parallèlement aux données elles-mêmes. Cette approche se distingue des précédentes en ceci qu'elle intègre les fonctionnalités de contrôle d'accès directement dans le gestionnaire de profil, ce qui constitue un premier pas vers une protection efficace des données personnelles.

La plate-forme framework ePerson [DRB+03], développé par Dickinson *et al*, met également en pratique un contrôle d'accès géré par l'utilisateur et une séparation claire entre données de profil et données de sécurité. Plusieurs concepts intéressants émergent de ces travaux, comme celui de « transfert minimal d'information », que l'on retrouve en substance dans les travaux réglementaires de l'Union Européenne [The02, Art04]. Ce principe veut que seule l'information absolument nécessaire à une transaction ou à un service doive être dévoilée. Dans cette optique, la plate-forme intègre des méthodes de négociation sur les informations transmises, la possibilité de falsifier automatiquement des informations pour protéger la vie privée de l'utilisateur ainsi que les concepts d'identités virtuelles anonymes et de créances abstraites (déjà utilisés dans plusieurs systèmes de protection de la vie privée, comme [Pea02a] ou [Int]).

3.2.2 S'abstraire de la couche de gestion des profils

Les modèles d'utilisateurs sont nombreux et aux structures variées. Nous avons vu que les méthodes de protection des données qu'ils contiennent étaient également de natures diverses, plus ou moins efficaces et parfois inexistantes. Il pourrait donc être intéressant pour un agent utilisateur de s'abstraire de cette couche de gestion des profils en insérant une couche intermédiaire masquant l'organisation des données.

Si une telle couche effectue une sérialisation des informations du profil, en attachant à une information donnée les informations attenant à son organisation (l'adresse dans un arbre ou un graphe par exemple), alors la protection de cette information sérialisée équivaut à la protection de l'information représentée dans le profil. Il est ainsi aisé d'imaginer un protocole utilisant un format XML générique pour la transmission des données entre agents, quel que soit le modèle d'utilisateur sous-jacent. La couche fonctionnelle chargée de la protection de ces données n'aura alors plus à considérer le modèle spécifiquement utilisé mais uniquement une valeur sérialisée. Bien entendu, il faut tout de même que l'agent destinataire de ces données connaisse le modèle d'utilisateur et la méthode de sérialisation utilisés.

Pour qu'une telle couche intermédiaire fonctionne efficacement, il faut toutefois que les méta-informations de nature ontologique ou relatives à la protection des données dans le profil soient traduites dans un formalisme lui aussi standardisé. Nous proposerons dans la section 4.2.1 une manière de représenter ces informations de manière raisonnée et efficace.

Dans la suite de nos travaux, nous décidons donc de ne pas réutiliser un modèle d'utilisateur particulier. Nous établirons nos propositions sans préjuger de l'organisation des informations dans un éventuel profil et en supposant que ledit profil ne protège pas efficacement les données qu'il contient. Nous faisons l'hypothèse qu'il est possible d'utiliser un modèle d'utilisateur particulier à condition de lui adosser une couche d'abstraction adaptée.

3.3 Modèle d'un *Privacy-Aware Agent*

Nous détaillons maintenant les axes de la thèse que nous défendons. La contribution qui la constitue est articulée autour de l'architecture de l'agent *PAw* (pour *Privacy-Aware*), un agent cognitif conscient de son contexte normatif en matière de protection des données personnelles. L'objectif fondamental que nous poursuivons dans la conception de cet agent est de lui permettre de gérer les données personnelles qui lui sont confiées de la meilleure manière possible, en regard des diverses réglementations qui régissent son activité. Utilisant le modèle d'agent BDI procédural JACK [AOS], l'agent PAw inclut des modules spécialisés mettant en œuvre des fonctionnalités de raisonnement logique sur les réglementations à respecter, ainsi que des outils de représentation et de raisonnement sur les outils techniques à mettre en œuvre pour cela.

Nous présentons tout d'abord les fonctionnalités que nous jugeons nécessaire, au vu de nos études et observations, pour aboutir à un agent personnel gérant correctement les données personnelles, puis nous proposons une intégration de ces fonctionnalités dans un modèle d'agent. La conception et la mise en œuvre détaillées de ces fonctionnalités seront développées dans les chapitres suivants.

3.3.1 Les composantes de la gestion PAw

Un agent PAw doit assurer un certain nombre de fonctionnalités essentielles. Comme nous l'avons déterminé dans la section 1.5.1, il doit notamment être capable de représenter explicitement les données personnelles (appartenant à son propriétaire ou à un autre utilisateur) ainsi que les différentes réglementations qui s'appliquent à celles-ci, afin de pouvoir agir conformément au cadre normatif auquel il est soumis. Nous avons vu dans le chapitre 2 que ces réglementations pouvaient provenir de sources hétérogènes et par conséquent comporter des incohérences. Un agent PAw doit donc être capable de raisonner en dépit de ces incohérences afin de déduire un ensemble de normes cohérent à respecter, servant de politique de sécurité. Il lui faut enfin asservir sa gestion des données personnelles à ces normes et faire en sorte d'étendre la protection des données personnelles aux autres agents du système.

3.3.1.1 Perception du contexte normatif

L'agent PAw dispose tout d'abord d'outils de communication lui permettant de se procurer les réglementations émises par les diverses autorités pouvant influer sur son comportement. Par le biais de requêtes directes à ces autorités, l'agent obtient l'expression de ces normes exprimées dans un langage formel de nature logique. Nous faisons ce choix afin de permettre à l'agent PAw de bénéficier, pour son raisonnement sur les réglementations, des mécanismes d'inférence de la logique déontique, spécifiquement adaptés à la notion de norme, détaillés dans le chapitre 2 et plus particulièrement dans la section 2.4. Nous utilisons ici un formalisme particulier, la logique DLP (pour *Deontic Logic for Privacy*) que nous allons développer de manière approfondie dans le chapitre suivant.

L'agent PAw est également capable de récupérer, depuis l'interface web d'un agent de service, une politique P3P [Wor06] pour en extraire des informations dans le formalisme DLP, de manière à pouvoir les mettre en relation avec les normes dont il a connaissance.

3.3.1.2 Raisonnement sur le contexte normatif

Disposant de ces multiples (et possiblement hétérogènes) ensembles de normes, l'agent PAw met en œuvre des procédures particulières pour vérifier certaines propriétés (notamment de syntaxe et de cohérence individuelle) sur les normes reçues et les autorités individuelles qui les ont émises. Puis les divers ensembles de normes sont assemblés pour former un tout cohérent. En effet, les sources de normes dans un environnement ouvert sont multiples, et nous avons vu dans la section 2.4.4 qu'un ensemble de normes devait être rendu logiquement consistant afin de constituer une politique de sécurité acceptable. Pour cela, l'agent PAw doit détecter les éventuels conflits qui existent entre ces ensembles et les arbitrer de manière à obtenir au final une collection de normes qui lui permette de diriger son comportement de manière non ambiguë.

Par la suite, l'agent devra être capable de décider si une ou plusieurs de ces normes concernent les actions qu'il souhaite mener, en fonction de celles qu'il a déjà effectuées ou observées.

3.3.1.3 Protection locale des données personnelles

Une fois établie une base de normes cohérente sur la base des réglementations, l'agent PAw doit avoir les moyens de s'assurer que les données personnelles qu'il est amené à traiter n'échappent pas à ces normes. C'est, comme nous venons de le dire, le cœur de notre proposition : l'agent PAw, de par ses spécifications, doit respecter au mieux les réglementations et donc la politique qu'il en a tirée. Nous supposons ici que l'agent est d'ores et déjà capable de distinguer les informations à caractère personnel des autres données informatiques (l'inférence automatique du caractère personnel dépassant le cadre de notre travail).

Nous choisissons de placer les informations personnelles dans une structure de données à part au sein de l'agent. En effet, ces données doivent être traitées différemment des autres, car les éléments de politique s'y appliquant doivent être au préalable examinés. Cette structure de données doit être conçue de telle manière que l'agent lui-même ne puisse y accéder sans qu'une vérification des normes ait lieu. Une telle organisation permet ainsi d'assurer une protection *locale* des données personnelles, limitée à l'enceinte logicielle de l'agent PAw.

3.3.1.4 Raisonnement sur les protocoles applicatifs

Les fonctionnalités présentées pour l'instant ne permettent à l'agent PAw que de protéger les données personnelles dans le cadre local de ses propres actions. Or, nous avons vu dans la section 1.3.2 que pour assurer la protection de la vie privée de l'utilisateur, la protection *étendue* des données personnelles est une nécessité lorsque ces données sont partagées au sein d'une application distribuée. Dans ce contexte, il faut donc que l'agent PAw soit capable de garantir que les normes sont respectées non seulement par lui mais également par tous les agents qui seront amenés à manipuler les données dont il a la garde. Dans cet objectif d'amélioration de la protection étendue des données personnelles, nous dotons l'agent PAw de connaissances spécifiques sur divers protocoles et architectures d'applications, qu'il nous faudra analyser en détail dans le chapitre 5. Ces connaissances, associées à l'ensemble de normes auquel il se soumet, lui permettent de prendre des décisions éclairées lorsque la nécessité survient de communiquer des informations personnelles à d'autres agents afin qu'elles soient traitées à l'échelle d'une application répartie. L'agent PAw a ainsi par exemple la possibilité de déterminer si une architecture est davantage favorable qu'une autre au respect des normes ou si la transaction dans son ensemble est inacceptable.

3.3.2 Architecture de l'agent PAw

Nous choisissons pour la réalisation de l'agent PAw d'utiliser le modèle d'agent JACK, présenté dans la section 2.1.3 comme mettant en œuvre les principes des agents BDI de manière procédurale. Nous justifions ce choix par la présence dans la plate-forme de développement de fonctionnalités utiles à la mise en œuvre de la protection locale des données personnelles (comme nous le verrons dans le chapitre 6) ainsi qu'au maintien de bases de croyances cohérentes.

Les fonctionnalités de l'agent PAw sont organisées en couches, suivant une architecture guidée par le modèle d'agent sous-tendu par la plate-forme JACK. On dispose nativement d'une couche de **croyances**, dans laquelle sont stockées toutes les informations non strictement statiques nécessaires aux fonctionnement de l'agent, une couche de **buts** BDI, qui sont des événements spéciaux auxquels l'agent fait en sorte de répondre d'une manière ou d'une autre, et une couche de **plans** qui constituent les procédures d'action spécifiques de l'agent. La figure 3.1 représente l'architecture de l'agent PAw, que nous allons détailler plus avant.

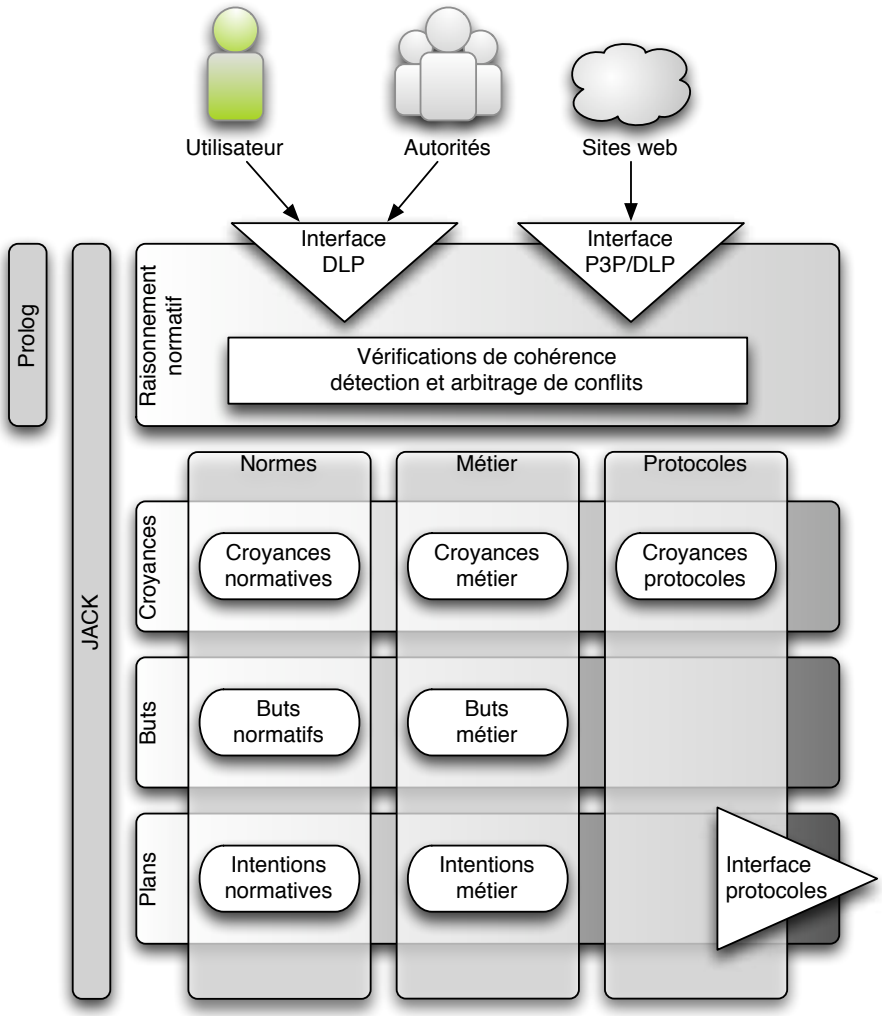

FIG. 3.1 – Architecture de l'agent PAw

3.3.2.1 Couche de raisonnement normatif

Au-dessus des trois couches préexistantes, nous ajoutons artificiellement une couche de **raisonnement normatif** à laquelle nous confions les fonctionnalités de perception et de raisonnement sur le contexte normatif de l'agent PAw. Cette couche fournit en sortie des ensembles de croyances « normatives » qui alimentent la couche de croyances de l'agent. Nous choisissons ainsi d'isoler les fonctionnalités de raisonnement sur le contexte applicatif que nous venons d'introduire, car elles ont pour objet la production d'une politique qui sera consommée par les autres fonctionnalités de l'agent PAw. La couche de raisonnement normatif fait appel à des mécanismes natifs du modèle d'agent JACK (croyances, buts, plans, communication) que nous représentons à part, et qui interagissent avec une machine virtuelle Prolog utilisée pour le raisonnement logique. C'est cette couche de raisonnement qui met en œuvre la logique DLP que nous allons détailler dans le chapitre suivant.

3.3.2.2 Couche de croyances

Les croyances manipulées par l'agent PAw sont de plusieurs ordres :

- Les **croyances normatives** sont celles qui sont fournies par la couche de raisonnement normatif. Elles sont notamment constituées d'un ensemble de normes cohérent issu de l'arbitrage des conflits (la politique de sécurité applicable par l'agent), de l'ensemble des normes désactivées par ce même mécanisme ainsi que d'autres informations annexes que nous détaillerons par la suite. Nous représentons ces informations dans l'agent PAw sous la forme de croyances à cause de leur caractère dynamique et de la nécessaire interprétation des normes effectuée par l'agent, qui fait de la politique de sécurité générée une information ne correspondant pas forcément à une vision du monde partagée par les autres agents. Nous isolons toutefois ces informations des autres croyances (de manière à nous affranchir des problèmes évoqués dans la section 3.2.1.3), car elles doivent être considérées séparément, et notamment avant chaque manipulation de données à caractère personnel.
- Les **croyances métier** sont les données nécessaires au fonctionnement quotidien de l'agent PAw, les croyances que devrait manipuler de toute manière un agent dans la même situation même s'il ne s'intéressait pas à la protection des données personnelles. Parmi ces croyances, certaines sont identifiées comme ayant un caractère personnel. Ces dernières sont donc isolées et traitées différemment, mais rentrent tout de même dans la catégorie des croyances « métier ».
- Les **croyances protocoles**, relativement statiques car fournies à l'agent à son démarrage, constituent sa base de connaissances relative aux protocoles et architectures d'application qu'il est susceptible d'utiliser pour collaborer avec d'autres agents.

3.3.2.3 Couche de buts

Les buts de l'agent PAw sont représentés par des événements (messages internes ou externes) d'un type particulier, représentant une « mission ». L'agent est capable de répondre à ces buts en leur associant un plan. Nous choisissons de diviser les buts de l'agent en deux familles : les **buts métier** qui traitent de ses missions applicatives en tant qu'agent personnel et les **buts normatifs**, engendrés par une analyse des croyances normatives sur la protection des données personnelles. Nous permettons ainsi de distinguer les activités métier de l'agent, qui peuvent être déclenchées par des événements extérieurs, du contrôle normatif qu'il s'impose, ce dernier étant subordonné à des événements essentiellement internes, créés en fonction des croyances normatives. Les buts normatifs vont influer sur la manière dont l'agent traite les buts métier, en conditionnant les actions menées par les plans associés pour faire en sorte que les normes actives soient respectées.

3.3.2.4 Couche de plans

Les plans JACK de l'agent représentent, lorsqu'ils sont activés, les intentions d'un agent BDI. C'est la couche la plus basse en terme d'abstraction cognitive, elle décrit les méthodes de fonctionnement concrètes de l'agent. Ici encore les intentions se divisent entre **intentions normatives** et **intentions métier**, les premières contraignant les secondes. Des ensembles de plans spécialisés, les plans « protocoles », sont également en charge de la communication avec l'extérieur, lorsque cette communication met en jeu des données personnelles. Ces plans sont appelés en relation avec les connaissances de l'agent sur la caractérisation des protocoles et les prises de décision qui en découlent, mais en eux-mêmes ils constituent plus un mécanisme d'interface qu'un concept cognitif.

3.3.3 Discussion sur le modèle

Comme nous l'avons vu, la structure de l'agent PAw est fondée sur les travaux relatifs aux modèles BDI et profite principalement des principes du modèle PRS. L'agent PAw manipulant, dans sa couche de raisonnement normatif, des concepts logiques, il pourrait être justifié de considérer un modèle d'agent purement logique. En effet, de nombreux travaux proposent des modèles d'agents cognitifs fondés par exemple sur des logiques multi-modales couvrant un ou plusieurs aspects du raisonnement. Nous avons pourtant choisi de travailler sur un modèle procédural pour deux raisons :

- Tout d'abord, le faible rapport observé entre les implémentations de modèles logiques et les modèles logiques théoriques est une illustration de la difficulté qu'il y a à mettre en œuvre des modèles intégrés en logique modale. Cela se fait nécessairement au prix d'une réduction du modèle à une réalisation moins expressive, n'autorisant pas l'exploitation de la pleine profondeur d'inférence du modèle théorique.
- Enfin et surtout, nous souhaitons mettre en avant le fait que les fonctionnalités que nous proposons pour l'agent PAw sont en principe indépendantes du modèle d'agent choisi. Notre contribution ici n'est en effet pas un modèle d'agent en soi, mais une collection d'outils, organisés de manière cohérente, permettant de doter un modèle d'agent donné de certaines fonctionnalités. Si la couche de raisonnement logique appelle naturellement à une mise en œuvre de nature logique (impliquant donc une certaine perte d'expressivité entre modèle et implémentation, comme nous l'avons dit plus haut), celle-ci peut s'intégrer dans un modèle procédural à condition que les interfaces soient correctement définies.

Le choix que nous faisons est ainsi principalement un choix de démonstration. La mise en œuvre de nos propositions est facilitée par le modèle d'agent JACK, mais ce dernier n'est pas une nécessité absolue. Les fonctionnalités de l'agent PAw pourraient tout aussi bien être intégrées à un modèle purement théorique, à un agent développé intégralement en Prolog, à un autre modèle procédural ou encore à un modèle d'agent fonctionnel.

Chapitre 4

Appropriation des normes par l'agent PAw

Nous nous intéressons maintenant à la couche supérieure du modèle de l'agent PAw. Elle constitue l'interface de l'agent avec les normes existant dans son environnement. Il peut ainsi s'informer de l'ensemble des normes auxquelles il se réfère, les organiser et raisonner en fonction de ces normes. Le composant principal de cette couche de conception est un moteur de raisonnement logique exploitant une *logique déontique pour la vie privée* (*Deontic Logic for Privacy* ou logique DLP) et lui permettant d'adapter sa gestion des informations personnelles en fonction de ce que nous nommerons son *contexte normatif*.

4.1 Le contexte normatif

L'agent PAw s'exécute dans un contexte bien défini, mais dynamique, constitué des normes auxquelles il doit se référer. Pour chaque transaction ou interaction avec un agent tiers, l'agent PAw doit évaluer ce contexte pour adapter son comportement en conséquence. Nous appellerons *contexte normatif* l'ensemble de ces normes (identifiées par leurs entités sources). Le contexte normatif varie en fonction du **temps**, mais aussi de la localisation physique de l'agent ou de ses interactions courantes avec d'autres entités (comme par exemple le fait qu'une transaction s'opère dans le cadre d'un contrat avec un agent tiers), c'est-à-dire sa **situation** dans son environnement.

Définition 4.1 (Contexte normatif)**.** Le contexte normatif de l'agent est l'ensemble des normes qui lui sont opposables à un instant et dans une situation donnés.

Dans cette définition, nous entendons par le terme « normes opposables » le fait qu'une entité extérieure puisse s'attendre à ce que l'agent connaisse ces normes et éventuellement les respecte. Il faut donc que ces normes soient mises à disposition par l'entité qui les émet (ou par un mandataire), que nous appelerons une *autorité normative*.

4.1.1 Autorités normatives

4.1.1.1 Organisations et institutions

La notion d'autorité dans les systèmes multi-agents normatifs est généralement représentée par la notion d'**institution**. Pour John Rogers Searle [Sea69], deux types de normes existent, qui permettent de fonder une institution à partir d'un groupe d'agents : des **normes constitutives**

et des **normes régulatrices**. Les normes constitutives ne sont pas des normes au sens de la classification de Tuomela [Tuo95], mais plutôt les spécifications d'une **organisation**, décrivant les conditions d'appartenance d'un agent au système. Dans des modèles d'organisations plus fins, comme Agent-Groupe-Rôle (AGR, conçu notamment par Jacques Ferber et Olivier Gutknecht [FG98, Gut01]) ou ses descendants, ces normes peuvent décrire les conditions pour qu'un agent appartienne à un groupe donné ou joue un rôle particulier dans l'organisation. Les normes régulatrices, quant à elles, sont celles qui régulent effectivement le système. Elles décrivent le fonctionnement désirable des agents membres de l'institution (éventuellement en fonction de leurs groupes et rôles). Ce sont ces dernières qui correspondent véritablement à ce que nous appelons « normes » jusqu'à présent.

L'institution constitue donc une autorité de référence pour les agents qui s'en réclament. Les deux notions d'organisation et d'institution ainsi présentées sont souples et relativement génériques. Le concept d'institution englobe clairement le fait pour une entité (agent, groupe d'agents, organisation) le fait de pouvoir être responsable de l'émission et de l'application d'un ensemble de normes, ce qui constitue une base intéressante pour la notion d'autorité.

4.1.1.2 Définition d'une autorité normative

La notion d'institution étant bien plus expressive qu'il ne nous est nécessaire, nous limiterons la notion d'autorité normative à la responsabilité vis-à-vis des normes, dans l'optique d'une meilleure généricité du concept.

Définition 4.2 (Autorité normative)**.** Une autorité normative est un agent ou un ensemble d'agents qui édicte des normes afin que d'autres agents aient conscience de l'existence de ces normes et puissent les appliquer.

On peut également définir la notion d'*autorité racine*, ou *autorité primaire*, en utilisant la notion d'autonomie : on peut considérer que l'agent qui constituera l'autorité normative primaire d'une norme est celui qui émet cette norme sans qu'elle lui soit imposée par une autorité tierce. Néanmoins, nous considérons qu'une autorité normative au sens large peut se voir imposer ses normes, ce qui permet la délégation d'autorité au sein d'une organisation ou d'une institution.

Il faut également distinguer la notion d'autorité normative de celle de dépôt de normes. Un dépôt de normes, s'il assure la fonction de mise à disposition des normes, n'en assume pas forcément la responsabilité, la notion d'autorité lui fait donc défaut.

4.1.1.3 Exemples d'autorités normatives

Suivant notre définition, toute entité responsable de la création et/ou de la propagation de règles peut être considérée comme une autorité normative.

Ainsi, les **États**, à cause de leur système législatif, constituent un exemple archétypal d'autorité normative. Les normes en sont les lois, décrets et autres textes, et l'entité responsable en est alors une organisation d'agents fort complexe. Les normes édictées par un État sont publiques, opposables à tous les agents (ce qui n'enlève rien à la possibilité qui leur est laissée de violer ces normes). Éventuellement, l'entité étatique peut être divisée en de nombreuses autorités normatives en fonction des juridictions et domaines de responsabilités.

Une **entreprise** (ou une **association**) peut également être une autorité normative, à cause de ses statuts et de son réglement intérieur, qui doivent être considérés comme des normes puisqu'ils définissent dans une certaine mesure la normalité d'un comportement au sein de l'organisation. De la même manière que pour un état, l'entreprise peut être divisée en plusieurs autorités normatives organisées suivant sa hiérarchie.

Un **contrat** ou un accord entre plusieurs parties constitue également une autorité normative dont les responsables sont les différents signataires. En effet, un contrat contient en général un ensemble de contraintes sur le comportement des signataires.

4.1.2 Hétérogénéité du contexte

Lors d'une transaction ou d'une collaboration avec un autre agent, l'agent PAw dépend d'un contexte normatif composite, dans lequel il doit respecter les normes émanant de plusieurs autorités normatives indépendantes.

Tout d'abord, l'agent PAw, durant son exécution, est situé géographiquement dans son environnement. Il dépend d'un certain nombre de ressorts juridiques et donc d'autorités normatives, et ce même si l'agent est mobile. Ainsi, l'agent PAw doit avoir conscience des textes de loi édictés par le pays de son exécution courante, qui va constituer une première autorité normative. D'autres États peuvent également entrer en ligne de compte en tant qu'autorités dans le contexte normatif. En effet, si l'agent PAw interagit avec un agent tiers situé dans une autre juridiction, il peut également avoir à prendre en compte une deuxième autorité normative législative de niveau national ou international.

L'agent PAw peut éventuellement appartenir, directement ou indirectement, à une entreprise. Parmi les multiples autorités normatives qui y sont associées, l'agent PAw aura conscience d'un certain nombre d'entre elles (par exemple, le supérieur hiérarchique direct dont il dépend et le réglement intérieur de l'entreprise). De la même manière, il peut avoir à considérer les normes de plusieurs contrats. Ce peut être le cas par exemple si les relations entre l'entreprise dont il dépend et une organisation tierce sont formalisées par des accords explicites.

Enfin, dernière autorité normative pour l'agent, l'utilisateur humain lui-même, son propriétaire, qui peut lui fournir un ensemble de préférences normatives concernant la protection des données personnelles. Il est normal et assez naturel de considérer que l'agent artificiel, qui agit comme mandataire de son utilisateur humain, reconnaisse ce dernier comme une autorité normative. Nous proposons donc, dans le cadre du modèle de l'agent PAw, d'introduire l'autorité normative `self`, spécifique à chaque agent, représentant les préférences personnelles de son propriétaire humain.

Le contexte normatif d'un agent PAw est donc intrinsèquement composite, hétérogène. La diversité des autorités permet de décrire des relations normatives riches, mais pose la question fondamentale de la cohérence de ce contexte normatif.

4.1.3 Cohérence du contexte

Les différentes autorités normatives auxquelles est soumis l'agent PAw sont a priori faillibles et indépendantes les unes des autres. Il peut donc arriver deux types de problèmes : une autorité normative peut émettre un ensemble de normes logiquement inconsistant, ou bien plusieurs autorités normatives peuvent émettre des normes qui entrent en conflit les unes avec les autres.

4.1.3.1 Incohérence à l'échelle de l'autorité normative

Il est envisageable qu'une autorité normative particulière émette des normes qui soient incohérentes entre elles. Considérons par exemple le cas des deux normes suivantes, émises par une seule autorité normative et ayant les mêmes conditions d'application :

> Après la livraison d'un produit chez un client, le numéro de digicode du client doit être conservé pendant deux mois (pour faciliter les livraisons futures).

> Après la livraison d'un produit chez un client, le numéro de digicode du client doit être supprimé immédiatement (pour préserver la vie privée du client).

Dans cet exemple simpliste, un agent PAw soumis à cette autorité normative devrait, pour respecter toutes les normes, à la fois supprimer et conserver la même information. L'incohérence est ici évidente, mais elle peut être plus profondément enfouie : ce pourraient être les conséquences des actions obligatoires qui soient incompatibles entre elles, et non pas les actions qui soient la négation l'une de l'autre. D'autre part, en raison même de la profondeur d'inférence à laquelle se trouve l'incompatibilité, l'incohérence peut concerner plus de deux normes : on peut imaginer un ensemble de n normes parfaitement cohérentes entre elles, mais qui présentent un conflit à l'ajout d'une $n+1^{\text{ème}}$ norme. La détection de ces incohérences dépend donc des facultés d'inférence de l'agent PAw.

Dans tous les cas, nous considérons ici que l'autorité normative n'est pas « raisonnable » et qu'elle émet des normes incohérentes. Dans cette situation, l'agent PAw doit pouvoir identifier le problème et ignorer temporairement l'autorité normative défaillante, jusqu'à ce qu'elle rétablisse la cohérence dans les normes qu'elle émet. Il est bon de noter que c'est ici une hypothèse simplificatrice ainsi qu'un poids particulier qui pèse sur l'entité chargée de traduire les normes du langage naturel en expressions formelles. En effet, il est courant que dans les sociétés humaines les normes émises par une même autorité soient incompatibles, à une profondeur d'inférence plus ou moins importante. La scission d'une autorité normative unique en plusieurs autorités distinctes peut permettre de diminuer l'impact de cette hypothèse sur l'adaptabilité de l'agent PAw, en se ramenant dans le cas où l'incohérence n'est pas interne à une autorité normative mais conséquente à la juxtaposition de plusieurs d'entre elles.

4.1.3.2 Incohérence entre plusieurs autorités normatives

Le second cas que nous avons à traiter concerne les incohérences qui surviennent entre plusieurs autorités normatives. Prenons le cas d'un agent PAw appartenant à une entreprise. Considérons deux autorités normatives en particulier, la loi du pays et le réglement intérieur de l'entreprise, édictant les deux normes (fictives) suivantes, respectivement :

> Il est interdit de conserver le numéro de carte de crédit d'un client après une transaction commerciale.

> Après la vente d'un produit à un nouveau client, son numéro de carte de crédit doit être conservé pendant une semaine.

Lorsque l'agent PAw traite avec un nouveau client, les deux normes s'appliquent et il se trouve face à un dilemme. Cependant, aucune des normes émises par les deux autorités n'est en soi incohérente, ce n'est que leur conjonction qui pose problème. Dans ce cas, l'agent PAw doit être capable de prendre la décision de respecter l'une des deux normes et pas l'autre.

Il nous faut donc mettre en place une méthode d'« arbitrage » de cette incohérence, permettant à l'agent PAw de désactiver certaines normes et d'en conserver d'autres, afin de décider d'un comportement respectant « au mieux » un contexte normatif a priori intenable. Ici, il peut paraître naturel de raisonner en se fondant sur les autorités normatives : l'agent trouve-t-il plus important de satisfaire la loi ou le réglement de l'entreprise ? Un agent humain, placé dans la même situation, pourrait choisir l'une ou l'autre autorité, selon ses valeurs ou préférences personnelles. Il nous faut donc prendre en considération cette **subjectivité** en élaborant une méthode d'arbitrage des incohérences.

Le conflit de normes que nous avons présenté est relativement simple, mais encore une fois les subtilités que nous avons déjà mises en lumière sont à considérer : l'incohérence peut concerner plus que deux normes, et cette incohérence peut être moins évidente qu'une simple contradiction.

4.1.4 Dynamique des normes

Suivant les applications dans lesquelles sera déployé l'agent PAw, celui-ci est susceptible de changer de contexte normatif très fréquemment, par exemple s'il est mobile au niveau international, s'il communique avec des agents sous le couvert de différents contrats ou bien s'il est en charge à la fois d'affaires professionnelles et personnelles pour son utilisateur humain. Le contenu des normes édictées par les autorités normatives peut également varier au cours du temps. Au cours du temps et de ses diverses activités, l'agent PAw verra ce contexte normatif évoluer, parce qu'il sera soumis à des autorités normatives différentes ou encore que les autorités normatives elles-mêmes modifieront l'ensemble de normes qu'elles édictent. Le contexte normatif est donc essentiellement dynamique, dépendant de l'espace, du temps et des interactions. Pour les raisons que nous allons détailler, il n'est donc pas raisonnable de considérer que la prise en compte des normes par l'agent PAw est calculée une fois pour toutes au lancement du système.

4.1.4.1 Modification des normes édictées par une autorité donnée

Pour chaque autorité normative reconnue par l'agent PAw, il faut considérer la possibilité que les normes associées soient modifiés : de nouvelles lois et réglementations peuvent être créées, d'anciennes abrogées, les directives d'une hiérarchie peuvent être modifiées. Ces préférences de l'utilisateur, qui comme nous l'avons vu constituent aussi une autorité normative pour l'agent PAw, peuvent également évoluer. Les instructions de l'utilisateur peuvent s'affiner au cours du temps, et donc comme pour toute autre autorité normative, les normes correspondantes sont dynamiques.

L'agent PAw doit donc pouvoir tenir compte de ce dynamisme en mettant régulièrement à jour la base de normes de chaque autorité normative (via des requêtes programmées auprès des agents les représentant).

4.1.4.2 Création ou suppression d'autorités normatives

Il peut également arriver qu'au cours de l'exécution de l'agent PAw, l'ensemble des autorités normatives à considérer soit modifié. L'utilisateur peut décider de ne plus reconnaître telle ou telle autorité, ou au contraire en introduire une nouvelle à prendre en compte. C'est en particulier le cas lorsqu'un contrat est signé entre plusieurs agents : il vient alors constituer une nouvelle autorité normative pour chacun d'entre eux. Lorsqu'un tel contrat arrive à terme, l'autorité en question disparaît pour les agents.

L'ensemble des autorités normatives doit donc pouvoir être modifié au cours de l'exécution de l'agent, entraînant à chaque fois un nouveau calcul de l'ensemble des normes à considéder.

4.1.4.3 Commutation de contexte

Il peut également être intéressant de considérer le cas où les normes édictées par chaque autorité sont stables, où l'ensemble des autorités est stable, mais où l'agent se place de lui-même dans un contexte différent. On peut prendre l'exemple d'un agent, localisé sur un terminal mobile, que son propriétaire utilise, suivant les cas, dans un cadre familial ou dans un cadre professionnel. Dans le cadre familial, les autorités normatives pourraient par exemple être les préférences de

l'utilisateur et la loi du pays. Au contexte normatif du cadre professionnel s'ajouteraient le supérieur hiérarchique de l'utilisateur humain, le réglement intérieur de l'entreprise et un certain nombre de contrats. De plus, si l'on imagine que l'utilisateur est fréquemment en déplacement dans un pays étranger, la loi et les réglementations locales viendront s'ajouter au contexte. Pour chacun de ses trois contextes, l'agent doit calculer l'ensemble de normes à suivre. Il serait donc intéressant de pouvoir précalculer et mémoriser un certain nombre de contextes, indépendants les uns des autres (ou évoluant indépendamment les uns des autres, si on les considère maintenant comme dynamiques), entre lesquels l'agent PAw commuterait lors des transitions.

4.1.5 Synthèse des besoins

Il nous faut maintenant concevoir l'outil qui permettra à l'agent PAw de raisonner sur son contexte normatif. Ce formalisme devra être capable de prendre en compte la multiplicité des autorités et la potentielle incohérence du contexte. Pour cela, les normes devront être étroitement liées aux autorités qui les ont édictées, et la représentation des incohérences entre autorités normatives ne doit pas conduire à une inconsistance généralisée du moteur de raisonnement.

Concernant le caractère dynamique du contexte normatif, il paraît raisonnable de considérer que l'outil de raisonnement puisse servir à travailler sur l'état du contexte à un instant donné, sans préjuger d'une future évolution des autorités. L'introduction ou la suppression de nouvelles normes ou de nouvelles autorités utiliseraient ainsi les résultats précédemment calculés. Le cas des commutations de contexte n'a pas nécessairement à être traité au sein du moteur de raisonnement normatif. Les commutations peuvent être mises en œuvre par le basculement vers une autre instance du moteur, en profitant éventuellement des calculs effectués pour un précédent contexte.

4.2 Définition de la logique DLP

Nous présentons maintenant la logique DLP, utilisée dans le modèle de l'agent PAw pour représenter le contexte normatif, raisonner sur des ensembles de normes et en déduire des schémas comportementaux.

La logique DLP est une logique déontique et temporelle, fondée sur la logique déontique standard et la logique temporelle linéaire, syntaxiquement définie par l'utilisation de modalités d'obligation et de modalités temporelles sur un langage spécifique, dont la grammaire décrit les six axes de la protection des données personnelles. Les formules DLP sont donc des normes décrivant des réglementations sur la gestion de ces données.

Dans la suite, nous adopterons dans les formules les conventions de style suivantes :

- Les mots-clés d'un langage (foncteurs de prédicats) sont en italique (exemple : *informContact*) ;
- Les valeurs littérales dans les formules sont dans une police à chasse fixe (exemple : `agent1`) ;
- Les éléments non terminaux d'une grammaire sont en petites majuscules (exemple : DATATYPE) ;
- Les variables non instanciées sont en caractères latins et commencent par une majuscule, par analogie avec Prolog (exemple : AnyAgent) ;
- Les lettres grecques φ et ψ désignent des formules DLP bien formées ;
- La lettre a en indice désigne l'identifiant d'un agent du système ;
- La lettre grecque ν en exposant désigne l'identifiant d'une autorité normative.

4.2.1 Syntaxe du langage de base $\mathcal{L}_{DLP}$

La logique DLP consiste en l'application de modalités sur un langage de base $\mathcal{L}_{DLP}$. Ce langage $\mathcal{L}_{DLP}$ est constitué de **prédicats informatifs**, qui représentent le fait qu'un agent utilisateur soit informé (à un instant donné) par l'agent responsable d'un traitement des spécificités de ce traitement, de **prédicats performatifs**, qui représentent chacun une action (toujours située dans le temps) spécifiquement liée à un traitement mettant en jeu des données personnelles, et de **prédicats d'état** servant à mémoriser les caractéristiques de ces traitements. Tous ces prédicats sont choisis pour décrire les informations nécessaires au raisonnement sur la protection des données personnelles. Il est donc possible d'y retrouver les six axes réglementaires précédemment identifiés.

LDLP-FORMULA = PREDICATE
| LDLP-FORMULA " $\vee$ " LDLP-FORMULA
| "$\neg$" LDLP-FORMULA ;

FIG. 4.1 – Grammaire du langage de base $\mathcal{L}_{DLP}$

La figure 4.1 présente la syntaxe du langage logique, en fonction des opérateurs classiques de disjonction et de négation[1]. La figure 4.2 détaille les prédicats de $\mathcal{L}_{DLP}$, exprimés à l'aide de quelques non-terminaux particuliers :

- A est une chaîne de caractères, identifiant unique d'un agent dans le système ;
- ID est une chaîne de caractères, identifiant unique (produit automatiquement) d'un traitement particulier (mettant en œuvre des données personnelles) ;
- DATAID est une chaîne de caractères, identifiant unique (produit automatiquement) d'une donnée personnelle particulière liée à un traitement identifié ;
- UPDATEID est une chaîne de caractères, identifiant unique (produit automatiquement) d'une requête de mise à jour d'une donnée personnelle ;
- DURATION est une valeur numérique entière (positive) représentant une durée[2] ;
- DATATYPE est une chaîne de caractères prise parmi un ensemble fini, désignant la nature d'une donnée personnelle. L'énumération de ces valeurs comprend notamment celle proposée par les spécifications du langage P3P pour l'élément `<datatype>` d'une politique [Wor06, section 5.5][3]. Par exemple, `user.home-info.telecom.telephone` désigne le numéro de téléphone du domicile d'un utilisateur ;
- ACTIONTYPE est une chaîne de caractères prise parmi un ensemble fini, désignant la nature d'un traitement donné. L'énumération de ces valeurs est l'union de celles proposées par les spécifications du langage P3P pour les éléments `<PURPOSE>` et `<PPURPOSE>` d'une politique [Wor06, section 3.3.5]. Par exemple, `health` se rapporte à un traitement lié aux services de santé, `delivery` à une opération de livraison.

Nous allons maintenant détailler la sémantique de chacun des prédicats constituant les éléments de base du langage $\mathcal{L}_{DLP}$. Ces prédicats sont reliés aux six dimensions réglementaires de la protection des données personnelles, comme le résume le tableau 4.1. Dans la suite, nous

[1] On s'autorisera bien sûr l'utilisation de la conjonction et de l'implication logique, définis comme à l'accoutumée, ainsi que le parenthésage des formules.

[2] Pour ce qui est de la représentation explicite des durées, nous considérerons une granularité d'une heure.

[3] Les types de données spécifiées par d'autres schémas personnalisés, ainsi que l'ensemble des éléments proposés pour l'élément `<CATEGORIES>` de P3P, peuvent également être utilisés.

$$
\begin{aligned}
\text{PREDICATE} &= \text{INFORMATIVE} \mid \text{PERFORMATIVE} \mid \text{STATE} \ ; \\
\text{INFORMATIVE} &= \text{“}\textit{informActionType}\text{(”} \ \text{A, A, ID, ACTIONTYPE} \ \text{“)”} \\
&\mid \text{“}\textit{informForward}\text{(”} \ \text{A, A, ID, DATAID, FW} \ \text{“)”} \\
&\mid \text{“}\textit{informDuration}\text{(”} \ \text{A, A, ID, DATAID, DURATION} \ \text{“)”} \\
&\mid \text{“}\textit{informContact}\text{(”} \ \text{A, A, ID, A} \ \text{“)”} \ ; \\
\text{PERFORMATIVE} &= \text{“}\textit{request}\text{(”} \ \text{A, A, ID, DATATYPE, DATAID} \ \text{“)”} \\
&\mid \text{“}\textit{consentRequest}\text{(”} \ \text{A, A, ID} \ \text{“)”} \\
&\mid \text{“}\textit{consent}\text{(”} \ \text{A, A, ID} \ \text{“)”} \\
&\mid \text{“}\textit{tell}\text{(”} \ \text{A, A, ID, DATAID} \ \text{“)”} \\
&\mid \text{“}\textit{perform}\text{(”} \ \text{A, ID} \ \text{“)”} \\
&\mid \text{“}\textit{updateRequest}\text{(”} \ \text{A, A, ID, DATAID, UPDATEID} \ \text{“)”} \\
&\mid \text{“}\textit{update}\text{(”} \ \text{A, ID, DATAID, UPDATEID} \ \text{“)”} \\
&\mid \text{“}\textit{forgetRequest}\text{(”} \ \text{A, A, ID, DATAID} \ \text{“)”} \\
&\mid \text{“}\textit{forget}\text{(”} \ \text{A, ID, DATAID} \ \text{“)”} \\
&\mid \text{“}\textit{forward}\text{(”} \ \text{ID, DATAID, ID, DATAID} \ \text{“)”} \ ; \\
\text{STATE} &= \text{“}\textit{actionType}\text{(”} \ \text{ID, ACTIONTYPE} \ \text{“)”} \\
&\mid \text{“}\textit{dataType}\text{(”} \ \text{ID, DATAID, DATATYPE} \ \text{“)”} \\
&\mid \text{“}\textit{forwardList}\text{(”} \ \text{ID, DATAID, A} \ \text{“)”} \\
&\mid \text{“}\textit{duration}\text{(”} \ \text{ID, DATAID, DURATION}\text{“)”} \\
&\mid \text{“}\textit{responsible}\text{(”} \ \text{A, ID} \ \text{“)”} \\
&\mid \text{“}\textit{contact}\text{(”} \ \text{A, ID} \ \text{“)”} \\
&\mid \text{“}\textit{owner}\text{(”} \ \text{ID, DATAID, A} \ \text{“)”} \ ; \\
\text{FW} &= \text{“[”}\{\text{A}\}\text{“]”} \ ;
\end{aligned}
$$

FIG. 4.2 – Prédicats du langage $\mathcal{L}_{DLP}$

considérerons un agent de service `a` désirant opérer un traitement `trait1` sur une donnée personnelle `data1`, de type `user.home-info.telecom.telephone`, appartenant à un agent utilisateur `b`.

4.2.1.1 Le prédicat informatif *informActionType*

Le prédicat *informActionType* représente le fait qu'un agent en informe un autre sur la nature d'un traitement identifié, par le biais de son ACTIONTYPE. Comme un traitement peut avoir plusieurs ACTIONTYPE, il peut y avoir plusieurs instances du prédicat pour un même traitement. Ce prédicat correspond à l'axe d'**information** de l'utilisateur, mais également à celui de la **justification**, en cela qu'il exprime la finalité de la collecte et du traitement. La formule (4.1) désigne l'information de l'agent `b` par l'agent `a` du fait que le traitement `trait1` est en rapport avec une action de bienfaisance.

$$\textit{informActionType}(\texttt{a}, \texttt{b}, \texttt{trait1}, \texttt{charity}) \tag{4.1}$$

prédicats	**Axes réglementaires**					
	Information	Consentement	Modification	Justification	Conservation	Transmission
informActionType	•			•		
informForward	•					•
informDuration	•				•	
informContact	•		•			
request	•					
consentRequest		•				
consent		•				
tell						
perform						
updateRequest			•			
update			•			
forgetRequest			•			
forget			•		•	
forward						•
actionType				•		
dataType				•		
forwardList						•
duration					•	
responsible						
contact			•			
owner		•				

TAB. 4.1 – Relations entre axes réglementaires et prédicats de $\mathcal{L}_{DLP}$

4.2.1.2 Le prédicat informatif *informForward*

Le prédicat *informForward* représente le fait qu'un agent en informe un autre de la liste des agents tiers à qui pourra être transmise une donnée personnelle particulière associée à un traitement identifié. La liste des identifiants des agents concernés est notée sous la même forme qu'une liste Prolog. La sémantique de ce prédicat implique que la liste est complète : pour un même traitement et une même donnée, il ne devrait y avoir qu'une instance du prédicat *informForward*. Ce prédicat se rapporte aux axes d'**information** de l'utilisateur et de **transmission des données**. La formule (4.2) désigne l'action de l'agent `a` informant l'agent `b` que la donnée `data1` qu'il sera amené à fournir en vue du traitement `trait1` pourra être transmis aux agents `c` et `d`.

$$informForward(\texttt{a}, \texttt{b}, \texttt{trait1}, \texttt{data1}, [\texttt{c},\texttt{d}]) \tag{4.2}$$

4.2.1.3 Le prédicat informatif *informDuration*

Le prédicat *informDuration* représente le fait qu'un agent en informe un autre de la durée de conservation prévue pour une donnée personnelle associée à un traitement identifié. La granularité temporelle est celle de l'heure. Ce prédicat correspond aux axes d'**information** et de **conservation** des données. La formule (4.3) désigne l'action de l'agent `a` informant l'agent `b` que la donnée `data1` qu'il sera amené à fournir en vue du traitement `trait1` sera conservé pendant trente jours[4].

$$\mathit{informDuration}(\texttt{a}, \texttt{b}, \texttt{trait1}, \texttt{data1}, \texttt{30*24}) \tag{4.3}$$

4.2.1.4 Le prédicat informatif *informContact*

Le prédicat *informContact* représente le fait qu'un agent en informe un autre de l'identité de l'agent à contacter pour faire valoir un droit d'accès ou de modification à l'égard des données attachées à un traitement identifié. Ce prédicat correspond aux axes d'**information** de l'utilisateur et de **modification** des données : il fournit à l'utilisateur les informations nécessaires pour faire appliquer un éventuel droit de rectification et/ou de suppression. La formule (4.4) désigne l'action de l'agent `a` informant l'agent `b` que l'agent à contacter pour le traitement `trait1` est lui-même.

$$\mathit{informContact}(\texttt{a}, \texttt{b}, \texttt{trait1}, \texttt{b}) \tag{4.4}$$

4.2.1.5 Le prédicat performatif *request*

Le prédicat *request* représente la demande d'une donnée particulière (identifiée par son DATAID et spécifiée par son DATATYPE) en vue d'un traitement. Ce prédicat se rapporte principalement à l'axe d'**information** de l'utilisateur, en cela qu'il permet de déclarer le type des données collectées. Par exemple, la formule (4.5) désigne la demande par l'agent `a` du numéro de téléphone personnel de l'agent `b`, en vue d'un traitement `trait1`. Dans le cadre de ce traitement, la donnée sera identifiée par la chaîne `data1`.

$$\mathit{request}(\texttt{a}, \texttt{b}, \texttt{trait1}, \texttt{user.home-info.telecom.telephone}, \texttt{data1}) \tag{4.5}$$

4.2.1.6 Le prédicat performatif *consentRequest*

Le prédicat *consentRequest* représente le fait qu'un agent déclare avoir donné toutes les informations relatives à un traitement et demande le consentement d'un autre agent pour ce même traitement. La sémantique de ce prédicat implique que dans un scénario sain, il survienne après l'ensemble des prédicats d'information et de demandes de données. Ce prédicat correspond à l'axe de **consentement** de l'utilisateur. La formule (4.6) désigne l'action de l'agent `a` demandant le consentement de l'agent `b` pour le traitement `trait1`.

$$\mathit{consentRequest}(\texttt{a}, \texttt{b}, \texttt{trait1}) \tag{4.6}$$

[4] Pour plus de lisibilité, nous nous autorisons la représentation des durées sous la forme d'un produit, bien qu'elles soient codées sous forme de constantes entières.

4.2.1.7 Le prédicat performatif *consent*

Le prédicat *consent* représente le fait qu'un agent déclare son consentement à un autre à propos d'un traitement identifié. Ce prédicat correspond évidemment à l'axe de **consentement** de l'utilisateur. La formule (4.7) désigne l'action de l'agent `b` autorisant l'agent `a` à procéder au traitement `trait1`.

$$consent(\texttt{b}, \texttt{a}, \texttt{trait1}) \tag{4.7}$$

4.2.1.8 Le prédicat performatif *tell*

Le prédicat *tell* représente le fait qu'un agent transmette effectivement une donnée personnelle à un autre en vue d'un traitement convenu. Il est bon de noter que les données en elles-mêmes ne sont pas représentées dans la logique DLP, qui permet uniquement de raisonner sur le fait que la délivrance de l'information identifiée a bien eu lieu. Ce prédicat ne correspond à aucun axe en particulier, il sert de point de repère dans le raisonnement normatif. La formule (4.8) désigne l'action de l'agent `a` transmettant son numéro de téléphone personnel (la donnée `data1` du traitement `trait1`) à l'agent `b`.

$$tell(\texttt{b}, \texttt{a}, \texttt{trait1}, \texttt{data1}) \tag{4.8}$$

4.2.1.9 Le prédicat performatif *perform*

Le prédicat *perform* représente la réalisation d'un traitement par un agent. Ce prédicat ne correspond à aucun axe en particulier, il sert de point de repère dans le raisonnement normatif. La formule (4.9) désigne l'action de l'agent `a` procédant au traitement `trait1`.

$$perform(\texttt{a}, \texttt{trait1}) \tag{4.9}$$

4.2.1.10 Le prédicat performatif *updateRequest*

Le prédicat *updateRequest* représente le fait qu'un agent demande à un autre de mettre à jour une donnée personnelle le concernant. Encore une fois, la nouvelle valeur de la donnée, comme l'ancienne, n'est pas représentée dans le langage. On se contente d'identifier la requête de manière unique. Ce prédicat correspond à l'axe de **modification** des données. La formule (4.10) désigne l'action de l'agent `a` demandant à l'agent `b` de mettre à jour la donnée `data1` liée au traitement `trait1`, la requête étant associée à l'identifiant `request1`.

$$updateRequest(\texttt{b}, \texttt{a}, \texttt{trait1}, \texttt{data1}, \texttt{request1}) \tag{4.10}$$

4.2.1.11 Le prédicat performatif *update*

Le prédicat *update* représente le fait pour un agent de mettre à jour une donnée personnelle, à la suite d'une requête de son propriétaire (auquel cas l'identifiant de la requête est mentionné) ou de son propre chef (auquel cas un nouvel identifiant est généré). Ce prédicat correspond à l'axe de **modification** des données. La formule (4.11) désigne l'action de l'agent `a` mettant à jour l'information `data1` associée au traitement `trait1`, suite à la requête `request1`.

$$update(\texttt{a}, \texttt{trait1}, \texttt{data1}, \texttt{request1}) \tag{4.11}$$

4.2.1.12 Le prédicat performatif *forgetRequest*

Le prédicat *forgetRequest* représente le fait qu'un agent demande à un autre de supprimer une donnée personnelle le concernant. Ce prédicat correspond à l'axe de **modification** des données. La formule (4.12) désigne l'action de l'agent `a` demandant à l'agent `b` de supprimer la donnée `data1` liée au traitement `trait1`.

$$\mathit{forgetRequest}(\texttt{b}, \texttt{a}, \texttt{trait1}, \texttt{data1}) \tag{4.12}$$

4.2.1.13 Le prédicat performatif *forget*

Le prédicat *forget* représente le fait pour un agent de supprimer une donnée personnelle, soit parce que son propriétaire le lui a demandé, soit parce que sa durée de conservation est écoulée, soit pour des raisons qui lui sont propres. Ce prédicat correspond à l'axe de **modification** des données. La formule (4.13) désigne l'action de l'agent `a` supprimant l'information `data1` associée au traitement `trait1`.

$$\mathit{forget}(\texttt{a}, \texttt{trait1}, \texttt{data1}) \tag{4.13}$$

4.2.1.14 Le prédicat performatif *forward*

Le prédicat *forward* représente le fait pour un agent de transmettre une donnée personnelle à un agent tiers, en faisant correspondre une certaine donnée d'un certain traitement à une donnée d'un autre traitement (mis en œuvre par cet agent tiers). Il est intéressant de noter que ce prédicat n'implique pas explicitement les agents responsables des traitements. La transmission d'une donnée personnelle n'est en effet représentée que par la relation s'établissant entre deux traitements gérés par deux agents. On peut ainsi représenter non seulement la transmission d'une donnée personnelle à un tiers, mais également la corrélation d'informations entre deux traitements gérés par le même agent. Ce prédicat correspond à l'axe de **transmission** des données à des tiers. La formule (4.14) désigne la mise en relation, par l'agent responsable du traitement `trait1`, de l'information `data1` du traitement `trait1` avec l'information `data2` dans le traitement `trait2`.

$$\mathit{forward}(\texttt{trait1}, \texttt{data1}, \texttt{trait2}, \texttt{data2}) \tag{4.14}$$

4.2.1.15 Le prédicat d'état *actionType*

Le prédicat *actionType* permet de mémoriser les différents ACTIONTYPE associés à un traitement donné. Il peut donc y avoir plusieurs instances du prédicat pour chaque traitement identifié. Ce prédicat correspond à l'axe de **justification**, en ce sens qu'il exprime la finalité d'un traitement. La formule (4.15) représente le fait que le traitement `trait1` est en rapport avec une action de bienfaisance.

$$\mathit{actionType}(\texttt{trait1}, \texttt{charity}) \tag{4.15}$$

4.2.1.16 Le prédicat d'état *dataType*

Le prédicat *dataType* permet de mémoriser le type d'une donnée personnelle associée à un traitement identifié. Ce prédicat correspond à l'axe de **justification** car il permet d'associer

le type d'une donnée à la finalité d'un traitement. La formule (4.16) représente le fait que la donnée `data1` du traitement `trait1` est un numéro de téléphone personnel.

$$dataType(\texttt{trait1}, \texttt{data1}, \texttt{user.home-info.telecom.telephone}) \tag{4.16}$$

4.2.1.17 Le prédicat d'état *forwardList*

Le prédicat *forwardList* permet de mémoriser les récipiendaires autorisés d'une donnée personnelle. Le prédicat ne désigne qu'un seul agent. Par conséquent, pour chaque donnée de chaque traitement, il y a autant d'instances du prédicat que d'agents récipiendaires. Dans un scénario sain, ces instances correspondent exactement aux récipiendaires déclarés par le biais du prédicat performatif *informForward*. Ce prédicat correspond à l'axe de **transmission** des données. La formule (4.17) représente le fait que l'agent `c` est enregistré comme un récipiendaire autorisé de la donnée `data1` associée au traitement `trait1`.

$$forwardList(\texttt{trait1}, \texttt{data1}, \texttt{c}) \tag{4.17}$$

4.2.1.18 Le prédicat d'état *duration*

Le prédicat *duration* permet de mémoriser la durée de rétention prévue pour une donnée particulière d'un traitement particulier. Dans un scénario sain, la valeur enregistrée par ce prédicat correspond à celle déclarée par le responsable du traitement. Ce prédicat correspond à l'axe de **conservation** des données. La formule (4.18) représente le fait que la donnée `data1` associée au traitement `trait1` soit enregistrée comme devant être conservée pendant trente jours.

$$duration(\texttt{trait1}, \texttt{data1}, \texttt{30*24}) \tag{4.18}$$

4.2.1.19 Le prédicat d'état *responsible*

Le prédicat *responsible* permet de mémoriser l'identité de l'agent responsable d'un traitement (l'agent de service). La formule (4.19) représente le fait que l'agent `a` est responsable du traitement `trait1`.

$$responsible(\texttt{trait1}, \texttt{a}) \tag{4.19}$$

4.2.1.20 Le prédicat d'état *contact*

Le prédicat *contact* permet de mémoriser l'identité de l'agent de contact pour un traitement particulier. Ce prédicat correspond à l'axe de **modification** des données car il identifie l'agent auprès de qui ce droit peut éventuellement être exercé. La formule (4.20) représente le fait que l'agent `a` est l'agent à contacter pour toute requête concernant le traitement `trait1`.

$$contact(\texttt{trait1}, \texttt{a}) \tag{4.20}$$

4.2.1.21 Le prédicat d'état *owner*

Le prédicat *owner* permet de mémoriser le propriétaire d'une donnée personnelle utilisée dans le cadre d'un traitement. Ce prédicat est lié à l'axe de **consentement** de l'utilisateur, en ceci

qu'il identifie l'agent qui a autorité pour donner son consentement. La formule (4.21) représente le fait que l'agent `b` est le propriétaire de l'information `data1` liée au traitement `trait1`.

$$owner(\texttt{trait1}, \texttt{data1}, \texttt{b}) \tag{4.21}$$

4.2.1.22 Identification de $\mathcal{L}_{DLP}$ à un langage propositionnel

Les résultats de la logique modale propositionnelle que nous pourrions utiliser ne sont applicables, par définition, que si le langage de base de la logique est propositionnel. Or, nous avons décrit $\mathcal{L}_{DLP}$ comme étant constitué de prédicats, ce qui peut paraître invalider les conditions d'applications. Cependant, il est possible de considérer ces prédicats comme des représentations pratiques et facilement utilisables d'atomes propositionnels. Par exemple, l'instance du prédicat *consent* proposée dans la formule (4.22) représente l'atome propositionnel fictif (4.23).

$$consent(\texttt{agent1}, \texttt{agent2}, \texttt{transaction1}) \tag{4.22}$$

$$\texttt{consent_agent1_agent2_transaction1} \tag{4.23}$$

Étant donné que chacun des arguments de chacun des prédicats de $\mathcal{L}_{DLP}$ a un domaine dénombrable, l'ensemble des atomes propositionnels représentés par ce langage de base est lui-même dénombrable. D'un point de vue théorique, on peut considérer que l'on travaille sur un langage propositionnel.

Dans les formules DLP on rencontrera souvent des prédicats partiellement instanciés, contenant des variables libres. Ces prédicats peuvent alors être compris comme des schémas propositionnels (sur le même principe que les schémas axiomatiques), représentant n'importe quel atome propositionnel correspondant. Ainsi, $consent(\texttt{agent1}, \texttt{agent2}, \text{TRANSACTION})$ représente les atomes `consent_agent1_agent2_transaction1`, `consent_agent1_agent2_transaction2`, `consent_agent1_agent2_transaction3`...

4.2.2 Syntaxe de la logique DLP

La logique DLP consiste en l'application d'un certain nombre de modalités, déontiques et temporelles, sur le langage $\mathcal{L}_{DLP}$. Il peut être intéressant de définir isolément un fragment déontique et un fragment temporel de la logique avant de spécifier la syntaxe complète de DLP, de manière à pouvoir par la suite raisonner de manière simple sur des propriétés essentiellement non temporelles ou non déontiques du langage.

4.2.2.1 Fragment déontique

La syntaxe du fragment déontique de la logique DLP est caractérisée par la grammaire (4.24). Elle consiste en l'application d'une classe de modalités d'obligation Ob_a^ν, qui correspondent au sens de l'obligation en logique déontique standard.

L'exposant ν désigne l'autorité normative qui a édicté la norme, il permet de rattacher les normes à leur source pour permettre au raisonneur de mettre en place, s'il le juge nécessaire, un traitement différencié par autorité normative. L'indice a, quant à lui, constitue une différenciation des modalités par agent : a est l'agent qui considère être tenu par cette obligation. Dans le cas le plus simple, qui est également celui qui va nous concerner ici sauf

indication contraire, a désigne l'agent local, le raisonneur. On pourra donc la plupart du temps ignorer l'indice des opérateurs déontiques sans perte significative d'information.

$$\begin{aligned} \text{DLP-DEON} = \; & \text{PREDICATE} \\ & \mid \text{DLP-DEON “}\vee\text{” DLP-DEON} \\ & \mid \text{“}\neg\text{” DLP-DEON} \\ & \mid \text{“}Ob_a^\nu\text{” DLP-DEON ;} \end{aligned} \tag{4.24}$$

Nous définissons de manière classique, en tant qu'abréviations, la classe de modalités de permission Per_a^ν, chacune d'entre elles étant la modalité duale de la modalité d'obligation correspondante (4.25). Nous utiliserons également les modalités d'interdiction For_a^ν (4.26). Des modalités abrégées d'option et de gratuité pourraient être définies de la même manière qu'en SDL, mais pour une meilleure lisibilité des formules nous n'en ferons pas usage dans nos travaux.

$$Per_a^\nu\ \varphi \stackrel{def}{=} \neg Ob_a^\nu\ \neg\varphi \tag{4.25}$$

$$For_a^\nu\ \varphi \stackrel{def}{=} Ob_a^\nu\ \neg\varphi \tag{4.26}$$

4.2.2.2 Fragment temporel

Les modalités introduites dans le fragment temporel de DLP sont empruntées à la logique LTL. On utilise le *until* strict $\mathcal{U}$, qui permet un raisonnement riche sur le futur, ainsi qu'un opérateur X^{-1}, permettant de raisonner sur la date immédiatement antérieure au présent, et un opérateur H permettant de travailler sur l'ensemble du passé. En effet, nous verrons par la suite que par la structure des réglementations en matière de protection des données personnelles, notre besoin d'expressivité est moins grand dans le passé que dans le futur.

Lorsqu'il est question de gestion (et de protection) des données personnelles, des actions situées dans le présent ou dans un futur proche peuvent être conditionnées par des événements ayant eu lieu dans un passé arbitrairement lointain (fonction des diverses durées et dates exprimées dans les réglementations). C'est pourquoi nous choisirons un flot de temps isomorphe à $(\mathbb{Z}, <)$ plutôt qu'à $(\mathbb{N}, <)$ afin de nous assurer la latitude temporelle nécessaire.

$$\begin{aligned} \text{DLP-TEMP} = \; & \text{PREDICATE} \\ & \mid \text{DLP-TEMP “}\vee\text{” DLP-TEMP} \\ & \mid \text{“}\neg\text{” DLP-TEMP} \\ & \mid \text{DLP-TEMP “}\mathcal{U}\text{” DLP-TEMP} \\ & \mid \text{“}H\text{” DLP-TEMP} \\ & \mid \text{“}X^{-1}\text{” DLP-TEMP;} \end{aligned} \tag{4.27}$$

Abréviations temporelles

Dans l'optique d'une meilleure lisibilité, il sera utile d'utiliser un certain nombre des abréviations temporelles courantes, à la sémantique parfois plus intuitive que celle de combinaisons des opérateurs fondamentaux $\mathcal{U}$, H et X^{-1}. Ainsi, nous nous autorisons l'usage des opérateurs suivants :

$$\varphi\, \mathcal{U}^{-}\, \psi \stackrel{def}{=} \psi \vee (\varphi \wedge \varphi\, \mathcal{U}\, \psi) \tag{4.28}$$

$$F\varphi \stackrel{def}{=} \top\, \mathcal{U}\, \varphi \tag{4.29}$$

$$G\varphi \stackrel{def}{=} \neg F \neg \varphi \tag{4.30}$$

$$P\varphi \stackrel{def}{=} \neg H \neg \varphi \tag{4.31}$$

Pour mémoire, l'abréviation « *until* faible » $\mathcal{U}^{-}$ (4.28) est similaire à sa contrepartie stricte, à ceci près qu'il est satisfait si la formule cible est déjà vraie à l'instant présent, ou dans le cas contraire que la formule à maintenir est déjà vraie à l'instant présent. La figure 4.3 présente trois modèles LTL. Dans le premier et le troisième de ces exemples, l'opérateur affaibli et sa version stricte ne sont pas interchangeables.

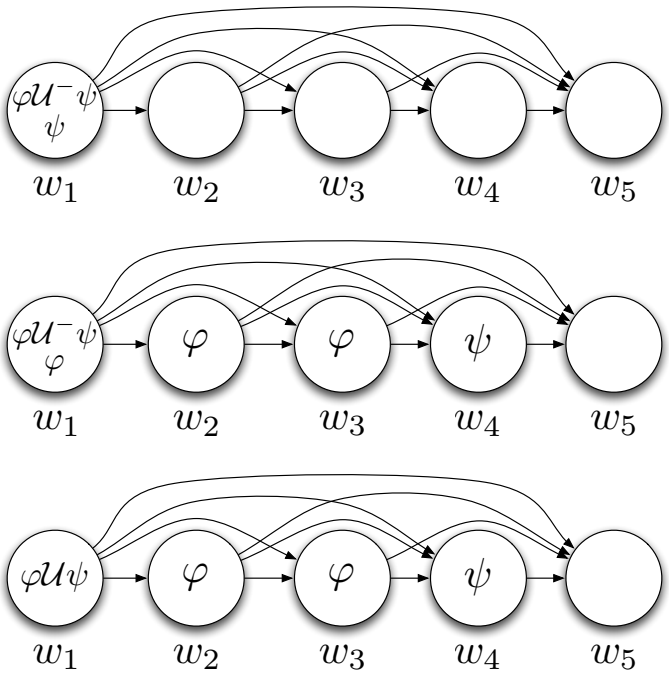

FIG. 4.3 – Spécificité de la modalité $\mathcal{U}^{-}$

La modalité H et les abréviations G, F et P ont leur sens usuel[5]. On s'autorisera également l'utilisation de leurs variantes affaiblies G^{-}, F^{-}, H^{-} et P^{-} incluant le présent.

Opérateurs *neXt* indexés

On utilise également un opérateur *neXt* unaire X permettant de désigner l'instant suivant, contrepartie de l'opérateur X^{-1} désignant l'instant précédent (4.32). Par combinaison de ces deux modalités, on définit un ensemble d'opérateurs *neXt* indexés par des entiers relatifs, permettant d'atteindre tout instant dont on connaît la « distance » au présent (4.33). Par exemple,

[5] À savoir, $G\ \varphi$ signifie que φ sera vraie à tout instant du futur, $F\ \varphi$ que φ sera vraie à un instant du futur, $H\ \varphi$ que φ a été vraie à tout instant du passé et $P\ \varphi$ que φ a été vraie à un instant du passé.

$X^5\varphi$ signifiera que φ sera vraie dans cinq pas de temps, et $X^{-2}\psi$ que ψ a été vraie il y a deux pas de temps.

$$X\varphi \stackrel{def}{=} \bot\, \mathcal{U}\, \varphi \tag{4.32}$$

$$\begin{cases} X^0\varphi \stackrel{def}{=} \varphi \\ \forall n \in \mathbb{Z}_+^*, X^n\varphi \stackrel{def}{=} XX^{n-1}\varphi \\ \forall n \in \mathbb{Z}_-^*, X^n\varphi \stackrel{def}{=} X^{-1}X^{n+1}\varphi \end{cases} \tag{4.33}$$

4.2.2.3 Syntaxe complète de DLP

Nous pouvons maintenant donner la grammaire du langage DLP final, dans laquelle les opérateurs déontiques et temporels sont librement mêlés.

$$\begin{aligned} \text{DLP} = \;& \text{PREDICATE} \\ & \mid \text{DLP “}\vee\text{” DLP} \\ & \mid \text{“}\neg\text{” DLP} \\ & \mid \text{“}Ob_a^\nu\text{” DLP} \\ & \mid \text{DLP “}\mathcal{U}\text{” DLP} \\ & \mid \text{“}H\text{” DLP} \\ & \mid \text{“}X^{-1}\text{” DLP ;} \end{aligned} \tag{4.34}$$

4.2.3 Axiomatique

L'axiomatique que nous proposons pour la logique DLP est fondée sur les axiomatiques standard des diverses modalités utilisées.

4.2.3.1 Règles d'inférence

Les règles d'inférence de la logique DLP sont celles de la logique propositionnelle (*modus ponens* et instanciation universelle), augmentées des règles de nécessitation spécifiques aux modalités introduites. On utilisera l'opérateur d'inférence logique $\vdash_{\text{DLP}}$, qui pourra être noté $\vdash$ en l'absence d'ambiguïté.

Les règles de nécessitation déontique (une règle pour chaque modalité déontique) permet de dériver l'obligation de tout théorème :

$$\frac{\vdash \varphi}{\vdash Ob_a^\nu\, \varphi} \tag{Ob_a^ν-RN}$$

Les règles de nécessitation temporelle permettent de dériver la permanence de tout théorème dans le passé et dans le futur :

$$\frac{\vdash \varphi}{\vdash G\varphi} \tag{G-RN}$$

$$\frac{\vdash \varphi}{\vdash H\varphi} \tag{H-RN}$$

4.2.3.2 Axiomatisation des opérateurs individuels

Les modalités Ob_a^ν sont entendues comme des obligation SDL, elles sont donc assorties d'une axiomatique de type KD.

$$Ob_a^\nu\ (\varphi \to \psi) \to (Ob_a^\nu\ \varphi \to Ob_a^\nu\ \psi) \qquad (Ob_a^\nu\text{-K})$$
$$Ob_a^\nu\ \varphi \to Per_a^\nu\ \varphi \qquad (Ob_a^\nu\text{-D})$$

La modalité $\mathcal{U}$ est axiomatisée comme précisé dans le chapitre 2, à l'exclusion de l'axiome de conversion (2.27) entre $\mathcal{U}$ et $\mathcal{S}$. On omet également l'axiome ($\mathcal{S}$-L), qui impose un point de départ au flot de temps, car la logique SDL est prévue pour travailler sur un flot de temps isomorphe à $(\mathbb{Z}, <)$ plutôt qu'à $(\mathbb{N}, <)$.

La modalité X^{-1} a la même axiomatique que celle présentée pour X dans le chapitre 2, à savoir $KDAlt$:

$$X^{-1}(\varphi \to \psi) \to (X^{-1}\varphi \to X^{-1}\psi) \qquad (\text{X}^{-1}\text{-K})$$
$$X^{-1}\varphi \to \neg X^{-1}\neg\varphi \qquad (\text{X}^{-1}\text{-D})$$
$$\neg X^{-1}\neg\varphi \to X^{-1}\varphi \qquad (\text{X}^{-1}\text{-Alt})$$

L'axiomatique des modalités G et F sur le futur est contenue dans l'axiomatique de $\mathcal{U}$ dont elles sont des abréviations, mais celle de H (et par extension de son dual P) doit être précisée. Elle correspond à l'axiomatique classique de l'opérateur en logique temporelle linéaire.

$$H(\varphi \to \psi) \to (H\varphi \to H\psi) \qquad (\text{H-K})$$
$$H\varphi \to P\varphi \qquad (\text{H-D})$$
$$H\varphi \to HH\varphi \qquad (\text{H-4})$$
$$(P\varphi \wedge P\psi) \to P(\varphi \wedge P\psi) \vee P(\varphi \wedge \psi) \vee P(P\varphi \wedge \psi) \qquad (\text{H-.3})$$

4.2.3.3 Axiomes de conversion entre modalités temporelles

La logique DLP permettant un raisonnement conjoint sur passé et futur, il est nécessaire d'utiliser les axiomes de conversion correspondants, empruntés à LTL, et liant les modalité H et X^{-1} à $\mathcal{U}$.

$$\varphi \leftrightarrow XX^{-1}\varphi \qquad (\text{XX}^{-1})$$
$$\varphi \leftrightarrow X^{-1}X\varphi \qquad (\text{X}^{-1}\text{X})$$
$$\varphi \to GP\varphi \qquad (\text{GP})$$
$$\varphi \to HF\varphi \qquad (\text{HF})$$

Il est également nécessaire d'introduire un certain nombre d'axiomes pour assurer les bonnes relations entre les modalités X^{-1} et H[6] :

[6] En réalité, l'axiome (HX^{-1}) est un théorème, il peut être dérivé de (X^{-1}P) et de (X^{-1}-D). Nous le laissons néanmoins présent pour des raisons de symétrie et de lisibilité de l'axiomatique.

$$X^{-1}\varphi \rightarrow P\varphi \qquad (\mathrm{X}^{-1}\mathrm{P})$$
$$H\varphi \rightarrow X^{-1}\varphi \qquad (\mathrm{HX}^{-1})$$
$$(X^{-1}\varphi \wedge X^{-1}H\varphi) \rightarrow H\varphi \qquad (\text{H-FP})$$
$$(X^{-1}\varphi \wedge H(\varphi \rightarrow X^{-1}\varphi)) \rightarrow H\varphi \qquad (\text{H-LFP})$$

4.2.3.4 Pertinence des axiomes de conversion entre modalités déontiques et temporelles

Il convient par ailleurs de considérer les relations existant entre modalités temporelles et modalités d'obligation. Les candidats principaux sont les équivalences suivantes :

- $Ob_a^\nu\ X\varphi \leftrightarrow XOb_a^\nu\ \varphi$
- $Ob_a^\nu\ F\varphi \leftrightarrow FOb_a^\nu\ \varphi$
- $Ob_a^\nu\ G\varphi \leftrightarrow GOb_a^\nu\ \varphi$

Ces axiomes de conversion sont connus pour avoir des conséquences dramatiques sur la complexité calculatoire du problème de satisfaisabilité. Cependant, ces considérations ne doivent pas nous arrêter si nous nous attachons à concevoir une logique dont le but est de représenter au mieux la réalité des réglementations. L'acceptation de tels axiomes de conversion aurait pour conséquence une certaine pérennité des obligations, dès lors qu'elle sont exprimées. Par exemple, si à une date donnée une autorité impose à un agent que φ soit vraie le lendemain, ou à un instant indéterminé du futur, ou définitivement, alors cette obligation ne pourra plus être invalidée, elle sera reportée jusqu'à la date à laquelle elle doit être respectée. Cette idée est en contradiction évidente avec la nécessité que nous avons identifiée de représenter des normes dynamiques, pouvant varier au cours du temps.

En conséquence, nous choisissons de ne pas adopter de tels axiomes, mais de nous engager à fournir des opérateurs déontiques temporels (définis comme des abréviations) qui prendront soin par eux-même de propager obligations et interdictions, et qui pourront être utilisés dans les cas où cela s'avérera nécessaire.

4.2.3.5 Axiomes de persistance des prédicats d'état

Dans le langage de base $\mathcal{L}_{DLP}$, nous avons introduit un certain nombre de prédicats que nous avons appelés « prédicats d'état ». En effet, ils servent à mémoriser des informations relatives à des traitements identifiés de manière unique. Afin que la valeur de vérité de ces prédicats ne soit pas limitée à un instant du temps mais puisse perdurer à partir du moment où ils sont connus par l'agent (en accord avec la sémantique de « mémorisation »), il faut inclure dans la logique DLP des axiomes de persistance de ces prédicats d'état :

$$\mathit{actionType}(\mathrm{ID}, \mathrm{ActionType}) \rightarrow G\mathit{actionType}(\mathrm{ID}, \mathrm{ActionType}) \qquad (\text{G-at})$$
$$\mathit{datatype}(\mathrm{ID}, \mathrm{DataID}, \mathrm{DataType}) \rightarrow G\mathit{datatype}(\mathrm{ID}, \mathrm{DataID}, \mathrm{DataType}) \qquad (\text{G-dt})$$
$$\mathit{forwardList}(\mathrm{ID}, \mathrm{DataID}, \mathrm{Agent}) \rightarrow G\mathit{forwardList}(\mathrm{ID}, \mathrm{DataID}, \mathrm{Agent}) \qquad (\text{G-fl})$$
$$\mathit{duration}(\mathrm{ID}, \mathrm{DataID}, \mathrm{Duration}) \rightarrow G\mathit{duration}(\mathrm{ID}, \mathrm{DataID}, \mathrm{Duration}) \qquad (\text{G-dur})$$
$$\mathit{responsible}(\mathrm{Agent}, \mathrm{ID}) \rightarrow G\mathit{responsible}(\mathrm{Agent}, \mathrm{ID}) \qquad (\text{G-res})$$
$$\mathit{contact}(\mathrm{Agent}, \mathrm{ID}) \rightarrow G\mathit{contact}(\mathrm{Agent}, \mathrm{ID}) \qquad (\text{G-ct})$$
$$\mathit{owner}(\mathrm{ID}, \mathrm{DataID}, \mathrm{Agent}) \rightarrow G\mathit{owner}(\mathrm{ID}, \mathrm{DataID}, \mathrm{Agent}) \qquad (\text{G-own})$$

L'introduction de ces axiomes dans le langage interdit toute modification ultérieure de ces valeurs. Ceci peut être justifié par l'identification unique des données et des traitements : si un traitement ayant existé dans le passé est remis en service par un nouvel agent, plutôt que de modifier l'instance mémorisée du prédicat *responsible* pour ce traitement, un nouveau traitement similaire sera créé avec un nouvel identifiant unique. La même justification s'applique pour les données personnelles, d'autant plus que l'on dispose du prédicat performatif *forward* pour mettre en relation une donnée personnelle avec une autre.

4.2.3.6 Axiomes de cohérence des prédicats performatifs

La logique DLP ne comporte pas d'axiomes supplémentaires concernant l'agencement des différents prédicats performatifs. Au premier abord, il pourrait pourtant paraître sensé de considérer, par exemple, un axiome du type de la formule (4.35), obligeant qu'une demande de suppression de données soit immédiatement suivie d'effet, ou encore de la forme de (4.36), l'imposant de manière mécanique dans la logique.

$$\mathit{forgetRequest}(\text{Agent1}, \text{Agent2}, \text{ID}, \text{DataID}) \rightarrow Ob_a^{\nu}\ X\mathit{forget}(\text{Agent2}, \text{ID}, \text{DataID}) \tag{4.35}$$

$$\mathit{forgetRequest}(\text{Agent1}, \text{Agent2}, \text{ID}, \text{DataID}) \rightarrow X\mathit{forget}(\text{Agent2}, \text{ID}, \text{DataID}) \tag{4.36}$$

Cependant, cela introduirait dans le langage des présupposés sur ce que doit être une politique de traitement des données personnelles (ou sur la nécessité de son respect par les agents), ce qui n'est pas l'objectif de DLP. La logique DLP doit permettre de raisonner sur des normes et des situations relevant de modèles réglementaires et sociaux très différents, et non prendre parti pour certaines règles. En revanche, un agent PAw peut très bien introduire lui-même ce genre de normes dans sa base de formules DLP, par exemple par le biais des préférences de son utilisateur ou via une autorité normative fictive.

4.2.3.7 Axiomes de déduction sur les types de données

Nous avons vu que le prédicat d'état *dataType* enregistrait une information sémantique sur une donnée personnelle particulière, en utilisant notamment les structures arborescentes des schémas de données P3P [Wor06, 5.5]. À cause même de cette structure, il peut être utile à l'agent PAw de disposer de moyens de généralisation des types de données. Par exemple, le type de donnée `user.home-info.telecom` est un fils du type `user.home-info`. En conséquence, la validité de la formule suivante fait tout-à-fait sens :

$$\begin{aligned} &\mathit{datatype}(\text{ProcessID}, \text{DataID}, \texttt{user.home-info.telecom}) \\ &\quad \rightarrow \mathit{datatype}(\text{ProcessID}, \text{DataID}, \texttt{user.home-info}) \end{aligned} \tag{4.37}$$

De la même manière, on peut concevoir, pour chaque schéma de données accepté par l'agent (ceux-ci pouvant éventuellement être fournis par des tierces parties, en complément des schémas de base de P3P), un ensemble (fini) de telles implications logiques (sous forme de schémas axiomatiques) représentant dans le langage DLP la structure arborescente du schéma de données.

4.2.3.8 Axiomes de cohérence entre prédicats informatifs et prédicats d'état

De la même manière, nous n'introduisons pas d'axiomes imposant que l'information d'un agent par un autre implique que les deux interlocuteurs enregistrent les données correspondantes

de manière identique dans des prédicats d'état. En effet, il est nécessaire de laisser la possibilité à un agent utilisateur de manipuler artificiellement les valeurs des prédicats d'état, par exemple pour se ménager des marges de sécurité sur les durées de conservation des données. De manière similaire, il faut pouvoir permettre à un agent de service de mettre en œuvre le même genre de marges de sécurité, ou au contraire de mentir délibérément à l'utilisateur sur ses intentions réelles. Si cette liberté ainsi laissée peut apparaître comme contraire aux principes qui nous guident (à savoir le développement d'agent capables de respecter strictement les normes en matière de vie privée), il faut bien comprendre également que la représentation des stratégies de violation est essentielle à leur appréhension et à leur anticipation par l'agent PAw.

Ces considérations posées, il est probable qu'un agent PAw bienveillant, ne pouvant considérer par exemple les axiomes (4.38) et (4.39) (forçant les prédicats d'état *actionType* et *duration* à correspondre à d'éventuelles déclarations passées) comme valides, intègre les formules (4.40) et (4.41) (où `self` désigne l'agent lui-même) dans sa base de normes[7]. Dans ces normes, l'agent s'oblige lui-même à informer son interlocuteur des valeurs qui sont à sa connaissance vraies concernant ces mêmes prédicats d'état.

$$\begin{aligned} &P^{-}\,\mathit{informActionType}(\text{ServiceAgent}, \text{UserAgent}, \text{ID}, \text{ActionType}) \\ &\quad \rightarrow \mathit{actionType}(\text{ID}, \text{ActionType}) \end{aligned} \tag{4.38}$$

$$\begin{aligned} &P^{-}\,\mathit{informDuration}(\text{ServiceAgent}, \text{UserAgent}, \text{ID}, \text{DataID}, \text{Duration}) \\ &\quad \rightarrow \mathit{duration}(\text{ID}, \text{DataID}, \text{Duration}) \end{aligned} \tag{4.39}$$

$$\left.\begin{aligned} &\mathit{request}(\texttt{self}, \text{OtherAgent}, \text{ID}, \\ &\qquad \text{Datatype}, \text{DataID}) \\ &\mathit{actionType}(\text{ID}, \text{ActionType}) \end{aligned}\right\} \rightarrow \begin{aligned} &Ob^{\texttt{self}}_{\texttt{self}}\ \mathit{informActionType}(\texttt{self}, \\ &\qquad \text{OtherAgent}, \text{ID}, \text{ActionType}) \end{aligned} \tag{4.40}$$

$$\left.\begin{aligned} &\mathit{request}(\texttt{self}, \text{OtherAgent}, \text{ID}, \\ &\qquad \text{Datatype}, \text{DataID}) \\ &\mathit{duration}(\text{ID}, \text{DataID}, \text{Duration}) \end{aligned}\right\} \rightarrow \begin{aligned} &Ob^{\texttt{self}}_{\texttt{self}}\ \mathit{informDuration}(\texttt{self}, \\ &\qquad \text{UserAgent}, \text{ID}, \text{DataID}, \text{Duration}) \end{aligned} \tag{4.41}$$

4.2.3.9 Différenciation et inconsistance

Comme en logique déontique standard, le schéma axiomatique (Ob^{ν}_{a}-D) menace d'inconsistance logique les systèmes contenant des obligations incompatibles. Ainsi, un ensemble de normes comme celui présenté en (4.42) est inconsistant.

$$\begin{cases} Ob^{\nu}_{a}\ \mathit{forget}(\texttt{agent1}, \texttt{trait1}, \texttt{data1}) \\ Ob^{\nu}_{a}\ \neg\mathit{forget}(\texttt{agent1}, \texttt{trait1}, \texttt{data1}) \end{cases} \tag{4.42}$$

Cette situation est cohérente avec le point de vue que nous avons choisi sur les incohérences de normes internes à une même autorité normatives : l'agent PAw n'a pas à raisonner dessus et par conséquent les éliminera avant de les introduire dans sa base DLP, sous peine de la voir devenir inconsistante.

Si l'on s'intéresse au cas des incohérences entre autorités distinctes, on observe que l'utilisation d'une modalité d'obligation unique pour chaque autorité normative et pour chaque agent permet (outre la différenciation des traitements) de limiter le risque d'inconsistance logique du système en cas d'incohérences dans les normes. En effet, l'ensemble de normes (4.43) ne provoque

[7] Nous considérons ici ces normes comme des obligations immédiates. Nous verrons par la suite comment affiner leur portée temporelle.

pas l'inconsistance en logique DLP, alors qu'avec une seule modalité (comme en SDL), l'ensemble de normes (4.44) empêche tout raisonnement sain, à cause du principe *ex contradictione sequitur quodlibet*.

$$\begin{cases} Ob_a^{\nu_1}\ \mathit{forget}(\texttt{agent1}, \texttt{trait1}, \texttt{data1}) \\ Ob_a^{\nu_2}\ \neg\mathit{forget}(\texttt{agent1}, \texttt{trait1}, \texttt{data1}) \end{cases} \tag{4.43}$$

$$\begin{cases} Ob\ \mathit{forget}(\texttt{agent1}, \texttt{trait1}, \texttt{data1}) \\ Ob\ \neg\mathit{forget}(\texttt{agent1}, \texttt{trait1}, \texttt{data1}) \end{cases} \tag{4.44}$$

Cette différenciation des opérateurs d'obligation et leur axiomatisation de type KD permet donc à DLP de rester compatible avec notre besoin de représenter les conflits entre autorités normatives, sans avoir à gérer l'incohérence d'une autorité unique.

4.2.4 Structures sémantiques

Tous les opérateurs de la logique DLP sont des modalités normales, chacun d'entre eux est donc candidat à une interprétation sur une structure de Kripke. Nous allons donc définir les *modèles DLP* qui serviront à l'interprétation formelle des formules du langage. Pour cela, nous introduisons plusieurs structures destinées à entrer en jeu dans la construction du modèle.

4.2.4.1 Système d'entités DLP

Étant donné que la logique DLP utilise une modalité déontique pour chaque agent et chaque autorité du système, la conception d'un modèle DLP nécessite la représentation de ces entités. Il est nécessaire à la correction de la définition de DLP en tant que logique que les agents et autorités considérés soient en nombre fini, afin que le nombre des modalités déontiques soit lui-même bien déterminé.

Définition 4.3 (Système d'entités DLP)**.** Un **système d'entités DLP** Σ est un couple $(\mathcal{A}, \mathcal{N})$ où :

- $\mathcal{A} = \{a_i\}_{0 \leq i \leq m}$ est un ensemble dont les éléments sont appelés les **agents** du système ;
- $\mathcal{N} = \{\nu_i\}_{0 \leq i \leq n}$ est un ensemble dont les éléments sont appelés les **autorités normatives** du système.

Dans la suite, les agents pourront être désignés par $a, a', a'' \ldots$ ou $a_i (0 \leq i \leq m)$ et les autorités normatives par $\nu, \nu', \nu'' \ldots$ ou $\nu_i (0 \leq i \leq n)$.

4.2.4.2 Domaine d'interprétation, histoires, dates et instants

La logique DLP étant fondée sur un produit entre une logique multi-modale déontique et une logique temporelle linéaire, elle sera interprétée sur une structure bi-dimensionnelle. La première dimension est l'axe du temps (que nous représenterons horizontalement) et la deuxième détermine la collection des exécutions possibles du système (organisées verticalement dans les représentations graphiques que nous pourrons en faire).

Cette grille bi-dimensionnelle, dont les nœuds seront les mondes de notre modèle, est appelée le *domaine d'interprétation* du système est est définie de la manière suivante :

Définition 4.4 (Domaine d'interprétation DLP)**.** Un **domaine d'interprétation DLP** $\mathcal{I}$ est un ensemble $\mathcal{H} \times \mathcal{T}$ où :

- $\mathcal{H} = \{h_i\}_{i \in \mathbb{N}}$ est un ensemble dont les éléments sont appelés **histoires** ;

– $\mathcal{T} = \{t_j\}_{j \in \mathbb{Z}}$ est un ensemble dont les éléments sont appelés **dates**.
Un couple $(h_i, t_j) \in \mathcal{I}$ est appelé un **instant** du domaine $\mathcal{I}$.

Les instants sont donc des points d'un espace à deux dimensions, identifié par deux coordonnées : une *histoire* (l'exécution du système considéré) et une *date*. Comme nous l'avons déjà dit, le flot de temps sur lequel nous travaillons est infini dans le passé comme dans le futur. Ainsi, s'il existe une date t_0, elle peut désigner un point de référence arbitraire mais en aucun cas une date d'origine absolue. Dans la suite, les histoires pourront être désignés par $h, h', h'' \ldots$ ou $h_i (i \in \mathbb{N})$, les dates par $t, t', t'' \ldots$ ou $t_j (j \in \mathbb{Z})$ et les instants par $i, i', i'' \ldots$ ou par des couples (h_i, t_j). La figure 4.4 représente une portion d'un domaine d'interprétation DLP. Les disques en représentent les instants.

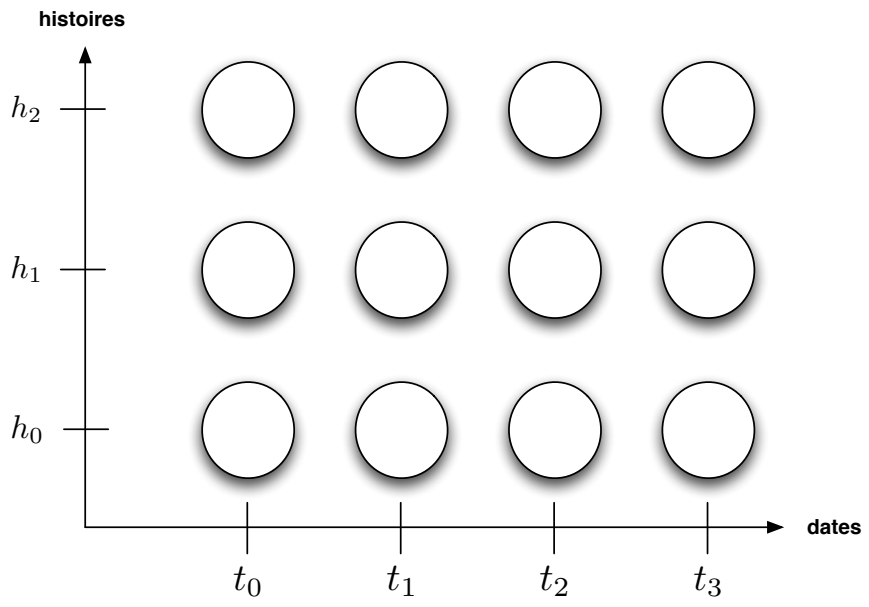

FIG. 4.4 – Exemple de domaine d'interprétation DLP

4.2.5 Sémantique de la logique DLP

Nous allons maintenant définir formellement la structure des modèles de la logique DLP, puis expliciter cette définition.

4.2.5.1 Modèles DLP

Définition 4.5 (modèle DLP)**.** Un modèle DLP $\mathcal{M}$ est un tuple $(\Sigma, \mathcal{I}, \rightsquigarrow, \mathcal{O}, v)$ où :
- Σ est un ensemble d'entités DLP ;
- $\mathcal{I}$ est un domaine d'interprétation DLP ;
- $\mathcal{O} = \{\mathbb{O}_a^\nu\}_{a \in \mathcal{A}, \nu \in \mathcal{N}}$ est un ensemble dont les éléments sont appelés **relations d'accessibilité déontique**. Les éléments de $\mathcal{O}$ sont des relations binaires sérielles entre instants : $\forall a \in \mathcal{A}, \forall \nu \in \mathcal{N}, \mathbb{O}_a^\nu \subseteq \mathcal{I} \times \mathcal{I}$) ;
- $\rightsquigarrow$ est une relation binaire sérielle à gauche, sérielle à droite, linéaire à gauche et linéaire à droite entre dates ($\rightsquigarrow \subset \mathcal{T} \times \mathcal{T}$) appelée la **relation de succession temporelle** ;

– v : $atom(\mathcal{L}_{DLP}) \longrightarrow \wp(\mathcal{I})$ est une application appelée la **fonction de valuation** du modèle $\mathcal{M}$.

4.2.5.2 Relations d'accessibilité déontique

L'utilisation d'un système d'entités Σ permet la caractérisation de la classe des modalités Ob_a^ν, à laquelle correspond naturellement une classe de relations binaires $\mathbb{O}_a^\nu$. Chacune de ces relations est sérielle (4.45), pour assurer la complétude de l'axiomatique des opérateurs déontiques par rapport à leur sémantique (le théorème de complétude de Sahlqvist associant la sérialité de la relation aux axiomes de type D).

$$\forall a \in \mathcal{A}, \forall \nu \in \mathcal{N}, \forall i \in \mathcal{I}, \exists i' \in \mathcal{I}, i\mathbb{O}_a^\nu i'' \tag{4.45}$$

4.2.5.3 Relations d'accessibilité temporelle

L'interprétation des opérateurs temporels est fondée par la relation $\rightsquigarrow$, qui associe toute date t_i à son successeur dans le temps t_{i+1}. Les caractéristiques de sérialité et de linéarité à gauche comme à droite assurent le bon ordonnancement des dates sur un axe linéaire infini, à savoir qu'une date a un et un seul successeur et un et un seul prédécesseur :

$$\forall t \in \mathcal{T},\ \exists t' \in \mathcal{T}, t' \rightsquigarrow t \tag{4.46}$$
$$\forall t \in \mathcal{T},\ \exists t' \in \mathcal{T}, t \rightsquigarrow t' \tag{4.47}$$
$$\forall t, t', t'' \in \mathcal{T},\ \text{si } t' \rightsquigarrow t \text{ et } t'' \rightsquigarrow t \text{ alors } t' = t'' \tag{4.48}$$
$$\forall t, t', t'' \in \mathcal{T},\ \text{si } t \rightsquigarrow t' \text{ et } t \rightsquigarrow t'' \text{ alors } t' = t'' \tag{4.49}$$

Pour plus de commodité, on définit la relation d'ordre stricte $\prec$ entre dates comme la fermeture transitive de la relation $\rightsquigarrow$, et $\succ$ comme la relation converse, permettant ainsi la comparaison de deux dates sur l'axe du temps :

$$\forall t, t' \in \mathcal{T},\ t \prec t' \stackrel{def}{=} \begin{cases} t \rightsquigarrow t' \\ \quad \text{ou} \\ \exists t'' \in \mathcal{T}, t \rightsquigarrow t'' \text{ et } t'' \prec t' \end{cases} \tag{4.50}$$
$$\forall t, t' \in \mathcal{T},\ t \succ t' \stackrel{def}{=} t' \prec t \tag{4.51}$$

Ces relations opérant sur les dates et non sur les instants, il est utile de définir des relations temporelles pour raisonner au sein d'une même histoire. Nous introduisons donc la relation $\hookrightarrow$ (respectivement, $\hookleftarrow$) qui lie, au sein d'une même histoire, un instant à son successeur (respectivement à son prédécesseur).

$$\forall h \in \mathcal{H}, \forall t, t' \in \mathcal{T},\ (h, t) \hookrightarrow (h, t') \text{ si et seulement si } t \rightsquigarrow t' \tag{4.52}$$
$$\forall h \in \mathcal{H}, \forall t, t' \in \mathcal{T},\ (h, t) \hookleftarrow (h, t') \text{ si et seulement si } t' \rightsquigarrow t \tag{4.53}$$
$$\tag{4.54}$$

De la même manière que précédemment, nous introduisons les abréviations $<$ et $>$ (et par extension, (respectivement, $\leq$ et $\geq$) comme les fermetures transitives (respectivement transitives et réflexives) des relations $\hookrightarrow$ et $\hookleftarrow$. Ainsi, pour résumer :

- $t \rightsquigarrow t'$ signifie que la date t est directement suivie par la date t' ;
- $t \prec t'$ signifie que la date t est strictement antérieure à la date t' ;
- $i \hookrightarrow i'$ signifie que l'instant i est directement suivi par l'instant i', au sein de la même histoire ;
- $i < i'$ signifie que l'instant i est strictement antérieur à l'instant i', au sein de la même histoire.

4.2.5.4 Interprétation des formules DLP

L'interprétation des formules du langage s'appuie sur la fonction de valuation v introduite dans la notion de modèle DLP. Cette fonction détermine, pour chaque atome propositionnel de $\mathcal{L}_{DLP}$, l'ensemble des mondes (instants) du modèle DLP dans lesquels cet atome est défini comme étant vrai. À partir de la, la valeur de vérité des formules est définie par induction sur leur structure, à la manière classique de l'interprétation des logiques modales normales sur les modèles de Kripke. Dans le cas d'une formule déontique par exemple, l'obligation d'une formule est vraie dans un monde si et seulement si la formule est vraie dans tous les mondes déontiquement accessibles. De la même manière, l'interprétation des formules temporelles correspond à leur interprétation classique en LTL.

On utilise un opérateur de modélisation $\models_{\text{DLP}}$, noté $\models$ en l'absence d'ambiguïté. Le tableau 4.2 donne la signification des différents types d'expressions impliquant cet opérateur, tandis que les formules (4.55) à (4.62) définissent formellement l'interprétation des formules DLP sur les modèles associés. Dans ces formules, p est un atome propositionnel de $\mathcal{L}_{DLP}$ ($p \in atom(\mathcal{L}_{DLP})$) et φ et ψ sont des formules DLP bien formées.

Formule sémantique	**Lecture**
$\mathcal{M}, h, t \models_{\text{DLP}} \varphi$	φ est vraie à la date t de l'histoire h du modèle $\mathcal{M}$
$\mathcal{M}, (h, t) \models_{\text{DLP}} \varphi$	
$\mathcal{M}, i \models_{\text{DLP}} \varphi$	φ est vraie à l'instant i du modèle $\mathcal{M}$
$\mathcal{M}, h \models_{\text{DLP}} \varphi$	φ est vraie à tout instant de l'histoire h du modèle $\mathcal{M}$
$\mathcal{M}, t \models_{\text{DLP}} \varphi$	φ est vraie à la date t pour toute histoire du modèle $\mathcal{M}$
$\mathcal{M} \models_{\text{DLP}} \varphi$	φ est vraie à toute date de toute histoire du modèle $\mathcal{M}$
	φ est $\mathcal{M}$-valide
$\models_{\text{DLP}} \varphi$	φ est vraie à toute date de toute histoire de tout modèle DLP
	φ est DLP-valide

TAB. 4.2 – Lecture de l'opérateur de modélisation de la logique DLP

$$\mathcal{M}, i \models \top \tag{4.55}$$

$$\mathcal{M}, i \models p \text{ si et seulement si } i \in v(p) \tag{4.56}$$

$$\mathcal{M}, i \models \neg\varphi \text{ si et seulement si } \mathcal{M}, i \not\models \varphi \tag{4.57}$$

$$\mathcal{M}, i \models \varphi \vee \psi \text{ si et seulement si } (\mathcal{M}, i \models \varphi \text{ ou } \mathcal{M}, i \models \psi) \tag{4.58}$$

$$\mathcal{M}, i \models Ob_a^{\nu}\ \varphi \text{ si et seulement si } (\forall i' \in \mathcal{I},\ \text{si } i\mathbb{O}_a^{\nu} i' \text{ alors } \mathcal{M}, i' \models \varphi) \tag{4.59}$$

$$\mathcal{M}, h, t \models \varphi\, \mathcal{U}\, \psi \text{ si et seulement si } \exists t' \in \mathcal{T}, \begin{cases} t \prec t', \\ \mathcal{M}, h, t' \models \psi \\ \forall t'' \in \mathcal{T}, \text{ si } t \prec t'' \prec t' \text{ alors } \mathcal{M}, h, t'' \models \varphi \end{cases} \tag{4.60}$$

$$\mathcal{M}, h, t \models H\varphi \text{ si et seulement si } \forall t' \in \mathcal{T}, \text{ si } t' \prec t \text{ alors } \mathcal{M}, h, t' \models \varphi \tag{4.61}$$

$$\mathcal{M}, h, t \models X^{-1}\varphi \text{ si et seulement si } \forall t' \in \mathcal{T}, \text{ si } t' \rightsquigarrow t \text{ alors } \mathcal{M}, h, t' \models \varphi \tag{4.62}$$

4.2.5.5 Représentation graphique

La figure 4.5 est une représentation d'un modèle DLP avec un unique opérateur d'obligation Ob. La relation $\hookrightarrow$ y est représentée par les flèches en trait plein et la relation déontique $\mathbb{O}$ par des flèches en pointillés. Afin de ne pas alourdir le schéma, seules quelques formules et quelques flèches sont représentées pour la relation d'accessibilité déontique. En réalité, de chaque monde du diagramme part au moins une flèche d'accessibilité déontique (expression de la sérialité de la relation). On peut noter en particulier que la relation d'accessibilité déontique s'entend entre instants et non entre histoires. En conséquence, si l'on peut considérer que les dates sont, pour chaque histoire, les mondes d'un modèle LTL, les histoires ne peuvent être considérées comme les mondes d'un quelquonque modèle déontique.

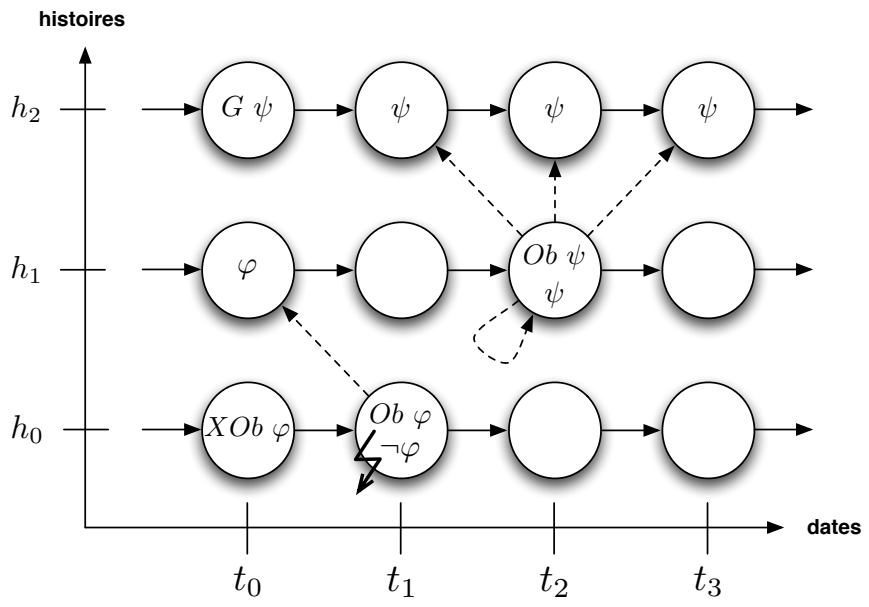

FIG. 4.5 – Exemple de modèle DLP (avec une seule modalité d'obligation)

Dans cet exemple, ψ est obligatoire à l'instant (h_1, t_2), et cette obligation est respectée. Tous les mondes (ou instants) déontiquement accessibles, c'est-à-dire reliés à (h_1, t_2) par des flèches en pointillés, respectent cette obligation. À l'instant (h_0, t_1), φ est obligatoire mais cette obligation est violée (ce que symbolise la flèche brisée). Il existe par contre un autre monde, dans une autre histoire, où cette obligation est respectée. Il est possible que pour l'histoire h_0, aucun monde ne satisfasse jamais φ. Ce n'est pas en contradiction avec l'axiome (Ob_a^ν-D) ni avec la

sérialité de la relation $\mathbb{O}$. À l'instant $(h2, t_0)$, $G\ \psi$ est vraie, ce qui signifie que ψ est vraie pour tous les instants suivants de la même histoire (sans préjuger de sa valeur de vérité dans d'autres histoires). À l'instant (h_0, t_0), $XOb\ \varphi$ est vraie, mais sans que $Ob\ X\varphi$ soit vraie (à cause de notre rejet de l'axiome de conversion correspondant). Ainsi, le monde (h_0, t_1) n'est pas nécessairement incompatible avec les normes du monde (h_0, t_0).

4.3 Utilisation de la logique DLP

Nous avons formellement défini une logique pour la représentation des réglementations en matière de protection des données personnelles, sur la base des logiques déontique et temporelle. Disposant de la mécanique de raisonnement associée à ce formalisme, il nous faut maintenant montrer comment l'utiliser pour permettre à l'agent PAw de construire un modèle de comportement cohérent.

4.3.1 P3P et DLP

Comme nous l'avons mentionné, le standard *Platform for Privacy Preferences* (P3P, [Wor06]) est un moyen efficace, standardisé et de plus en plus répandu pour communiquer sur les politiques de protection des données personnelles d'un site web. Il convient donc d'observer comment un langage de ce type peut être mis en regard de notre proposition de formalisation logique. Nous allons montrer qu'ils poursuivent des objectifs concourants mais distincts et qu'ils ont vocation à collaborer dans le cadre d'une application distribuée sur internet, comme au sein même des activités cognitives de l'agent PAw. Nous nous appliquerons donc à assurer la meilleure intéropérabilité possible entre ces deux langages.

4.3.1.1 Pertinence d'une comparaison entre DLP et P3P

Il est délicat de comparer les deux outils que sont DLP et P3P. Certes ils opèrent sur le même domaine sémantique de la protection des données personnelles, mais ils ne partagent pas les mêmes objectifs. P3P est un langage purement déclaratif, destiné à décrire une politique de gestion des données opérée par un site internet. La dimension normative d'une politique P3P est restreinte à l'entité qui l'émet, aucun autre agent du système n'a à se sentir contraint par cette politique. C'est donc une situation très différente d'une autorité normative émettant des réglementations devant être suivies. Le même type de constat peut s'appliquer à d'autres langages de politiques comme SPARCLE [KS02] ou *XACML Privacy Policies* [Org05], des langages plus orientés vers les politiques de sécurité d'entreprises. P3P présente par ailleurs l'inconvénient d'être limité à une utilisation par des sites web[8].

DLP, de son côté, est un langage plus déductif que déclaratif, visant à raisonner sur la pertinence d'un comportement ou d'une politique de comportement vis-à-vis d'un contexte normatif donné. DLP raisonne sur des concepts essentiellement normatifs, c'est-à-dire que le langage permet de représenter non seulement une politique de sécurité, mais également ce que devrait être une politique de sécurité. Il apparaît donc que DLP a vocation à extraire des informations depuis des politiques P3P afin d'alimenter une base de formules sur lesquelles raisonner.

En réalité, il est plus pertinent de chercher à comparer DLP avec un langage d'expression de préférences comme APPEL [Wor02], langage du W3C associé à P3P. APPEL permet à un agent

[8] En P3P 1.0, toute politique était forcément rattachée à une URI. Cela interdisait donc l'utilisation du langage dans le cadre de SOAP ou de certains services web. Depuis la version 1.1, le W3C s'applique à attacher les politiques à des attributs génériques XML, élargissant ainsi le champ d'application de P3P.

d'évaluer une politique P3P au regard d'un certain nombre de règles édictées par l'utilisateur. L'avantage d'un agent PAw à utiliser un moteur DLP se résume alors en quelques points :

- Possibilité de prendre en compte un grand nombre d'autorités normatives, en sus des préférences de l'utilisateur ;
- Possibilité de construire un modèle de comportement pour l'agent, dépassant la simple validation d'une politique ;
- Raisonnement sur des concepts temporels complexes, plutôt que sur une correspondance syntaxique.

Cependant, pour des raisons évidentes de compatibilité avec les standards, l'utilisation d'APPEL pour la spécification des préférences de l'utilisateur (constituant une des multiples autorités normatives pour l'agent) semble être un choix tout-à-fait naturel et justifié, quitte à l'étendre avec des concepts plus riches.

En résumé, le moteur DLP de l'agent PAw présente des fonctionnalités de raisonnement qui sont tout simplement inexistantes dans les outils de politiques strictement déclaratifs. Les outils DLP auront donc pour vocation d'extraire les informations existantes dans d'autres langages pour alimenter ses mécanismes d'inférences. Dans le cadre de nos travaux, nous nous limiterons à l'interaction entre DLP et P3P.

4.3.1.2 Exploitation d'une politique P3P par le moteur DLP

L'objectif purement déclaratif d'une politique P3P permet de considérer son existence comme un acte de langage passif (ou par défaut) d'un site web. En conséquence, ces politiques peuvent être exploitées pour en déduire des instances de prédicats appartenant au langage $\mathcal{L}_{DLP}$ (la notion de politique est ainsi distinguée de la notion de norme), qui permettront un raisonnement ultérieur de l'agent PAw. Le lecteur intéressé pourra trouver dans l'annexe B les principes d'extraction de formules $\mathcal{L}_{DLP}$ à partir de politiques P3P.

4.3.2 Représentation des « normes datées »

Quelles que soient les notions sur lesquelles elle porte, l'utilisation d'une simple modalité déontique ne suffit pas à refléter la réalité (et notamment la situation dans l'écoulement du temps) des devoirs qui incombent à un agent. Nous en sommes déjà convaincus, puisque nous avons intégré une logique temporelle à DLP. Néanmoins, même ainsi, il nous faut nous pencher de plus près sur la relation entre temps et devoir, de manière à pouvoir construire des normes porteuses d'un sens concret et applicable.

La notion de norme datée n'est pas nouvelle à proprement parler. Parmi les travaux s'y rapportant on peut notamment citer ceux de Frank Dignum *et al.* [DBDM04], Robert Demolombe *et al.* [Dem05] ou Julien Brunel *et al.* [BBF06]. Nous apportons ici un nouvel éclairage sur la question et adaptons les propositions existantes au formalisme de la logique DLP. Notre travail sur les normes datées a été présenté à l'occasion de deux communications scientifiques [PD08a, PD08b].

4.3.2.1 De la nécessité des normes datées

Si l'on utilise une simple obligation $Ob_a^\nu\ \varphi$ au sein de notre flot temporel, cette obligation n'est valable que pour l'instant présent. C'est ce que nous appellerons une obligation immédiate : la seule manière pour un agent de la respecter est d'avoir déjà fait en sorte que φ soit vraie au moment même où la norme devient vraie. Pour que cette situation soit acceptable, il faudrait

que l'agent ait été mis au courant à l'avance que cette obligation immédiate allait survenir, afin qu'il soit en mesure d'élaborer un plan permettant de la respecter.

On peut donc imaginer que l'agent ait eu connaissance d'une norme $FOb_a^\nu\ \varphi$, lui signifiant que cette obligation deviendrait vraie dans le futur. Seulement, tant que ce moment n'est pas arrivé (et l'agent n'a aucun moyen ici de savoir quand il arrivera), aucune obligation ne s'applique à lui, aucune « pression normative » ne l'engage à élaborer un plan.

On pourrait alors suggérer une norme $Ob_a^\nu\ F\varphi$, l'obligeant à faire en sorte que φ devienne vraie dans le futur. On dispose ici de cette pression normative, et un agent de bonne volonté pourrait effectivement décider de mettre en œuvre un plan pour s'assurer de φ. Cette formulation pose néanmoins problème, car le flot de temps sur lequel nous travaillons est infini dans le futur. En conséquence, à aucun moment un observateur extérieur ne pourra décider que l'agent, n'ayant pas encore assuré la valeur de vérité de φ, a violé la norme initiale. L'agent peut en effet toujours prétendre (de bonne ou de mauvaise foi) qu'il a l'intention d'assurer φ dans un futur plus lointain, et ainsi de respecter la norme. C'est pour cette raison qu'il est usuellement considéré qu'une obligation temporelle sans échéance est vide de sens : il est impossible de caractériser une violation associée à une telle norme.

Forts de cette réflexion, nous pourrions utiliser les opérateurs *neXt* indexés pour préciser une telle échéance. La norme $X^n Ob_a^\nu\ \varphi$, par exemple, spécifie une date exacte à laquelle l'obligation sera vraie. Ainsi, une violation pourra être définie à cette date si φ n'est alors pas vraie. Cependant, on a par la même occasion perdu la notion de pression normative que nous avions précédemment : pour que cette dernière soit présente il faut que la modalité externe de la formule soit une modalité déontique.

Ces quelques exemples illustrent la subtilité du problème et mettent en avant le besoin d'un opérateur spécifique pour décrire les obligations avec échéances. Un tel opérateur doit avoir un certain nombre de propriétés pour correspondre à sa signification intuitive, pour permettre la caractérisation d'une violation et appliquer une « pression normative » sur l'agent, même en l'absence d'une obligation immédiate.

4.3.2.2 Propositions-dates

Afin de caractériser les échéances de manière saine, nous proposons de travailler sur des atomes propositionnels capables de représenter les dates des modèles DLP. Les dates étant des notions uniquement sémantiques, elles ne transparaissent pas naturellement dans le langage DLP, et ce que nous appelerons les *propositions-dates* permet de construire un nouveau lien très expressif entre langage et méta-langage.

Définition 4.6 (Proposition-date)**.** En logique DLP, une proposition-date δ est un atome propositionnel qui est vrai à une et une seule date du flot de temps, et ce pour toute histoire du modèle. On note $date(\delta)$ le fait que l'atome δ soit une proposition-date.

$$date(\delta) \stackrel{def}{=} \begin{cases} \delta \wedge G\neg\delta \wedge H\neg\delta \\ \vee F(\delta \wedge G\neg\delta \wedge H\neg\delta) \\ \vee P(\delta \wedge G\neg\delta \wedge H\neg\delta) \end{cases} \tag{4.63}$$

On pourra remarquer que l'introduction des propositions-dates dans la logique DLP est une extension du langage propositionnel de base $\mathcal{L}_{DLP}$. Nous nous dispenserons toutefois de fournir une nouvelle grammaire formelle, l'impact sur la structure du langage étant minime.

Si l'on a accepté qu'il existe une date de référence t_0 dans le flot de temps, on peut lui faire correspondre une proposition-date de référence δ_0. De la même manière, on fait correspondre

les propositions-dates indicées par des entiers relatifs aux dates indicées correspondantes. Le fait que les δ_i soient des propositions-dates est donc ajouté à l'axiomatique de DLP (4.64). La spécification sémantique correspondante est exprimée par la formule (4.65). On supposera naturellement, le flot de temps étant discret, que les dates indicées sont ordonnées par leurs indices. Néanmoins, cette considération ne transparaît pas dans les formules que nous proposons, c'est uniquement une suggestion pratique de notation.

$$\forall i \in \mathbb{Z}, \vdash date(\delta_i) \tag{4.64}$$

$$\forall i, j \in \mathbb{Z}, \begin{cases} \mathcal{M}, t_i \models \delta_i \\ \text{si } i \neq j \text{ alors } \mathcal{M}, t_i \not\models \delta_j \end{cases} \tag{4.65}$$

L'utilisation de propositions pour marquer des jalons dans le flot de temps est une méthode déjà utilisée (par exemple par Demolombe *et al.* [Dem05]) et que nous adoptons pour son caractère intuitif. Néanmoins, nous verrons plus tard qu'il existe une méthode alternative mais pas forcément incompatible (notamment utilisée par Brunel *et al.* [BBF06]), qui consiste à préferer une représentation explicite des durées pour représenter les caractéristiques temporelles des normes.

4.3.2.3 Obligations avec échéances

Notre objectif ici est la définition d'un opérateur d'obligations avec échéances $Ob_a^\nu(\varphi, \delta)$ dont la signification serait l'obligation que φ devienne vraie entre l'instant présent (inclus) et l'instant (exclus) où δ sera vraie. Par exemple, si δ est vraie à l'instant présent, l'obligation est forcément violée, alors que si $\varphi \wedge \neg\delta$ est vraie à l'instant présent, elle est d'ores et déjà respectée.

Critères pour la correction de l'opérateur

À partir (notamment) de nos constatations intuitives, nous proposons huit critères pour s'assurer qu'un opérateur candidat dispose des propriétés requises.

1. **La date d'échéance doit être une proposition-date** : pour qu'une obligation assortie d'une échéance ait un sens pour l'agent, il faut que cette échéance soit définie de manière claire et distincte. La définition que nous avons donnée des propositions-dates fait en sorte qu'une telle échéance existe vraiment dans le flot de temps et qu'elle ne survienne qu'une seule fois.
2. **L'échéance doit être située strictement dans le futur** : c'est une évidence pour un interprète humain, une échéance dans le passé (ou même dans le présent) ne laisse aucune marge de manœuvre à un agent, son respect est indépendant de ses actions et de ses capacités.
3. **Les obligations sur $\top$ doivent être des tautologies**[9] : nous introduisons ce principe par mesure de cohérence avec l'obligation déontique immédiate, pour laquelle $Ob_a^\nu\ \top$ est un théorème. De la même manière, $Ob_a^\nu(\top, \delta)$ devrait être un théorème.
4. **Les obligations sur $\bot$ doivent être des antilogies** (*i.e.* être toujours fausses) : de la même manière que précédemment et pour les mêmes raisons, $\neg\ Ob_a^\nu(\bot, \delta)$ devrait être un théorème.

[9] Il faut être attentif à ne pas confondre la valeur de vérité (ou la dérivabilité) des obligations avec le fait qu'elles soient respectées ou non. Dans le cas d'une modalité normale, dotée de la règle de nécessitation, les deux notions correspondent pour les obligations sur des théorèmes. Cependant, l'obligation avec échéance n'est a priori pas une modalité normale.

5. **Les obligations non respectées doivent être abandonnées à l'échéance** : le choix de ce principe est davantage d'ordre philosophique que purement logique, il est par conséquent sujet à débat. Nous estimons que si l'agent n'a pas pu ou voulu se conformer dans les temps à l'obligation, alors, le mal étant fait, cette obligation ne doit plus pouvoir être dérivée. Ceci n'exclut pas l'existence éventuelle d'obligations *contrary-to-duties* qui pourraient survenir suite à la violation d'une obligation avec échéance. Ce critère permet à l'agent de ne pas avoir à tenir compte indéfiniment d'obligations datées qu'il ne pourra jamais respecter et qui sont peut-être caduques pour des raisons liées à leur signification dans le monde réel. Il est de plus une conséquence logique de notre deuxième exigence, suivant laquelle une échéance doit être située dans le futur.
6. **Les violations doivent être définies comme étant ponctuelles** : dans certains formalismes logiques, on considère que l'agent est dans un « état » de violation dès lors qu'une obligation avec échéance n'a pas été respectée. Étant donné l'expressivité qui nous est donnée par la logique DLP, et notamment la possibilité de raisonner sur le passé, il nous paraît suffisant de pouvoir considérer que l'agent a violé telle obligation à une date donnée du passé. Cela revient donc à considérer la violation comme un événement plutôt que comme un état, ce qui est à la fois un débat sémantique et une conséquence de l'expressivité du formalisme choisi.
7. **Le principe de propagation doit être respecté** : ce principe dit que l'obligation avec échéance doit être maintenue d'une date à la suivante, tant que l'obligation n'a pas été respectée ou que l'échéance n'est pas atteinte (4.66)[10].

$$Ob_a^\nu(\varphi, \delta) \wedge \neg(\varphi \vee \delta) \wedge \neg X\delta \rightarrow XOb_a^\nu(\varphi, \delta) \tag{4.66}$$

8. **Le principe de monotonie doit être respecté** : ce principe dit que s'il existe une obligation avec une échéance, alors on peut dériver la même obligation pour une échéance plus lointaine (4.67). Par exemple, si un agent doit détruire une information avant le premier juillet courant, alors il est normal de considérer qu'il doit également la détruire avant le premier août.

$$Ob_a^\nu(\varphi, \delta) \wedge F(\delta \wedge F^- \delta') \rightarrow Ob_a^\nu(\varphi, \delta') \tag{4.67}$$

Les six premiers critères ont été débattus par Dignum *et al.*, avec cependant des conclusions différentes. En effet, le formalisme utilisé étant assez différent du langage DLP, les violations y sont notamment considérées comme permanentes (à cause de l'impossibilité d'un raisonnement sur le passé). Sur certains aspects comme la nature des échéances, les auteurs engagent un débat sans prendre délibérément position. Notamment, ils acceptent l'éventualité d'utiliser $\perp$ comme échéance pour signifier une obligation sur le futur sans échéance particulière. Cela reviendrait à notre suggestion « naïve » d'utiliser $Ob_a^\nu\ F\varphi$, ce que nous avons rejeté dans le cadre d'une obligation avec échéance, mais qui reste utilisable dans le langage DLP. Concernant les obligations sur $\top$, les auteurs (utilisant un opérateur de type STIT impliquant la notion de responsabilité des agents) récusent la possibilité pour un agent de satisfaire à une obligation sur une tautologie. Dignum *et al.* nous rejoignent cependant sur l'inconsistance nécessaire des obligations portant sur des contradictions.

Les deux critères de monotonie et de propagation ont été introduits par Brunel *et al.* et nous les avons adoptés en les adaptant à notre notion stricte d'échéance. Nous nous sommes dispensés

[10] Comme notre notion d'échéance est stricte, il ne faut pas non plus que l'échéance survienne à l'instant suivant.

d'ajouter certains critères évidents. Notamment, la violation de l'obligation devra être définie de manière à correspondre au fait que φ ne survienne pas dans l'intervalle de temps considéré.

Nous nous proposons maintenant d'évaluer trois opérateurs déjà existants pour l'obligation avec échéances au regard de nos huit critères[11]. Dans le cadre de ces évaluations, un symbole d'ensemble vide ($\emptyset$) ou une absence de symbole représentera un obstacle majeur au respect du critère correspondant ; un carré plein ($\blacksquare$) représentera un respect natif et pleinement satisfaisant du critère, un carré vide ($\square$) représentera un respect partiel ou nécessitant une modification mineure.

L'opérateur de Dignum *et al.*

Le premier opérateur auquel nous nous intéressons, que nous noterons Ob^{α}, est celui défini par Dignum *et al.* [DBDM04]. La formule (4.68) est une transcription de cet opérateur dans un formalisme compatible avec celui de DLP. On peut observer notamment qu'en raison du formalisme originel, qui ne repose pas sur une logique déontique pure, la notion d'obligation avec échéance est définie conjointement avec celle de sa violation par l'agent (notée `viol`)[12].

$$Ob^{a}(\varphi,\delta) \stackrel{def}{=} (\neg\texttt{viol}\ \mathcal{U}^{-}\ \delta) \wedge \left(\begin{array}{c} \left[\begin{array}{c} ((\neg\varphi \wedge \neg\delta)\ \mathcal{U}^{-}\ \varphi) \\ \wedge(\neg\varphi\ \mathcal{U}^{-}\ (\varphi \wedge G^{-}\neg\texttt{viol})) \end{array} \right] \\ \vee \\ \left[\begin{array}{c} ((\neg\varphi \wedge \neg\delta)\ \mathcal{U}^{-}\ \delta) \\ \wedge(\neg\delta\ \mathcal{U}^{-}\ (\delta \wedge G^{-}\texttt{viol})) \end{array} \right] \end{array} \right) \tag{4.68}$$

Cette définition relativement complexe se lirait de la manière suivante : l'obligation de φ avec une échéance δ est définie d'une part par l'absence de violation avant la survenue de l'échéance, et d'autre part par l'une des deux situations suivantes :

- Soit φ survient avant l'échéance et l'on n'aura jamais de violation après la survenue de φ ;
- Soit l'échéance survient avant que φ n'ait été satisfaite, et l'agent est en situation de violation à partir de ce moment et pour l'ensemble du futur.

Le fait que cette définition ne repose pas sur un opérateur déontique est un inconvénient notoire dans le cadre de notre travail, car cela signifie que dès lors qu'une succession d'événements semble s'être déroulée comme si elle avait été contrainte par une obligation avec échéance, alors cette obligation avec échéance peut être déduite[13].

[11] Nos évaluations portent en fait sur des traductions de ces opérateurs dans notre formalisme ; en conséquence, certaines caractéristiques détaillées ici peuvent ne pas correspondre strictement à leurs contrepartie dans les formalismes d'origine. Cette évaluation a donc une portée limitée à la logique DLP : nous ne comparons pas la qualité des propositions en elles-mêmes, mais le fait que leur transcription en logique DLP soit adaptée ou non à nos besoins.

[12] Le caractère peu expressif de la violation, cette proposition binaire qui veut qu'un agent ou non soit en situation de violation, tient principalement au fait qu'une caractérisation plus précise n'était pas nécessaire pour le propos particulier de la communication à laquelle nous faisons référence [DBDM04]. Il serait cependant tout à fait possible, dans ce même contexte, de définir des violations plus spécifiques et contextualisées.

[13] Par exemple, supposons qu'un agent effectue un traitement `traitement` (action représentée par le prédicat *perform*(`agent`, `traitement`), qui sera notre formule φ), portant sur des données personnelles, avant une date arbitraire δ, et plaçons-nous à un instant dans le passé de ce traitement. Rien n'indique que l'agent est en violation à quelque moment que ce soit, par conséquent $\neg\texttt{viol}\ \mathcal{U}^{-}\ \delta$ est vérifiée. On peut également vérifier que ni *perform*(`agent`, `traitement`) ni δ ne surviennent avant le traitement, donc $(\neg\varphi \wedge \neg\delta)\ \mathcal{U}^{-}\ \varphi)$ est vraie aussi. Enfin, comme rien ne permet d'affirmer que l'agent sera jamais en situation de violation, on peut aisément accepter la valeur de vérité de $(\neg\varphi\ \mathcal{U}^{-}\ (\varphi \wedge G^{-}\neg\texttt{viol}))$. Avec tous ces éléments, il est possible, par définition de l'opérateur, de dériver $Ob^{\alpha}(\varphi,\delta)$ à tout instant antérieur au traitement, ce qui est absolument contre-intuitif : rien ne paraissait obliger l'agent à effectuer ce traitement.

Faisant abstraction de cet inconvénient fondamental, voyons comment l'opérateur Ob^{α} répond à nos critères :

1. $\emptyset$ – Les échéances peuvent ne pas respecter les contraintes d'une proposition-date. Notamment, on peut définir une obligation avec une échéance de $\top$, qui se réduit à une obligation immédiate ;
2. $\square$ – L'échéance ne peut être située dans le passé, mais elle peut toutefois être située à l'instant présent ;
3. $\square$ – Le caractère tautologique d'une obligation sur $\top$ dépend du fait que l'agent soit ou non en situation de violation. Si ce n'est pas le cas, alors le critère est respecté ;
4. $\emptyset$ – Si l'échéance existe dans le futur, alors les obligations sur $\perp$ ne sont pas inconsistantes mais forcent l'état de violation de l'agent après l'échéance[14] ;
5. $\blacksquare$ – Il est effectivement impossible de conserver une obligation après son échéance, qu'elle ait été respectée ou non ;
6. $\emptyset$ – Les violations sont définies comme persistantes et non ponctuelles (ce qui est pleinement justifiée dans le formalisme d'origine, qui ne permet pas de raisonner sur le passé) ;
7. $\blacksquare$ – Le principe de propagation est respecté, en conséquence même du problème de l'existence de dérivations abusives de l'opérateur ;
8. $\emptyset$ – L'opérateur n'est pas monotone, il existe des cas où une obligation avec une échéance plus tardive ne peut être dérivée[15].

L'opérateur de Demolombe *et al.*

Le deuxième opérateur à considérer, noté Ob^{β} dans notre formalisme, est proposé par Demolombe *et al.* [Dem05]. Il est originellement présenté dans une logique déontique et dynamique développée par Krister Segerberg [Seg03]. La formule (4.69) en représente une traduction dans un formalisme compatible avec DLP.

$$Ob^{\beta}(\varphi, \delta) \stackrel{def}{=} Ob(F^{-}(\varphi \wedge F\delta))\ \mathcal{U}^{-}\ (\varphi \vee \delta) \tag{4.69}$$

Cette définition se lirait ainsi : l'obligation de φ avec une échéance δ est définie par le fait que soit maintenue, jusqu'à ce que l'obligation soit respectée ou que l'échéance ne survienne, l'obligation d'assurer dans le futur une situation telle que φ soit vraie et que δ appartienne encore au futur. Voyons comment l'opérateur se comporte vis-à-vis des huit critères énoncés :

1. $\emptyset$ – Aucune restriction n'est posée sur la nature de l'échéance, qui peut survenir zéro ou plusieurs fois dans le flot de temps[16].
2. $\emptyset$ – Du fait du point précédent, l'échéance peut survenir (entre autres) dans le passé.
3. $\blacksquare$ – Les obligations sur $\top$ sont tautologiquement vraies.
4. $\square$ – Les obligations sur $\perp$ ne sont inconsistantes que lorsque l'échéance n'est pas l'instant présent.

[14] Le non-respect de ce critère (et du précédent) est dû à la traduction de l'opérateur dans notre formalisme et notamment à la perte de l'opérateur STIT E_a, interdisant que les agents puissent être responsables de l'avénement de tautologies ou de contradictions.

[15] C'est en particulier le cas lorsque l'obligation d'origine est respectée et qu'une autre obligation sur la même formule est violée après la première échéance. La monotonie est également mise en défaut si l'obligation est respectée seulement après l'échéance.

[16] Dans la formulation que nous avons donnée de cette opérateur, le fait que l'échéance ne survienne jamais dans le futur implique que l'obligation en elle-même est équivalente à $\perp$, sauf si φ est déjà vraie à l'instant présent.

5. ■ – Les obligations sont bien abandonnées après l'échéance, grâce à la sémantique de l'opérateur $\mathcal{U}^-$;
6. □ – Les violations initialement définies par les auteurs sont permanentes (4.70). Il est néanmoins trivial de les transformer en violations ponctuelles (4.71).

$$viol^{\beta}(\varphi, \delta) \stackrel{def}{=} P(\delta \wedge P(Ob^{\beta}(\varphi, \delta) \wedge \neg\varphi \; \mathcal{U}^- \; \delta)) \tag{4.70}$$

$$viol^{\beta'}(\varphi, \delta) \stackrel{def}{=} \delta \wedge P(Ob^{\beta}(\varphi, \delta) \wedge \neg\varphi \; \mathcal{U}^- \; \delta) \tag{4.71}$$

7. ■ – Le principe de propagation est respecté, par construction sur la base de l'opérateur $\mathcal{U}^-$.
8. □ – L'opérateur ne respecte le principe de monotonie que lorsque l'obligation est effectivement respectée. Si l'obligation est violée, il est impossible de dériver des obligation avec des échéances plus lointaines, à cause de la nature de la borne de l'opérateur $\mathcal{U}^-$. Nous nommerons cette propriété la *semi-monotonie*. Cette propriété peut au final s'avérer plus intéressante que la monotonie stricte, car elle permet d'éviter qu'un nombre infini de violations ne soient générées pour une seule obligation initiale violée, tout en conservant la possibilité de dériver des obligations avec des échéances plus lointaines lorsque celles-ci sont effectivement respectées.

L'opérateur de Brunel *et al.*

Le dernier opérateur évalué, Ob^{γ}, est une proposition de Brunel *et al.* [BBF06]. Il est d'ores et déjà exprimé dans un formalisme proche du nôtre, mais raisonne sur des durées plutôt que sur des échéances. Nous respectons ce choix en nous appuyant sur les opérateurs X^i et les propositions-dates δ_i déjà définies, en supposant que l'obligation avec échéance est actée à la date de référence δ_0[17]. L'obligation avec échéances est ici fondée sur un opérateur intermédiaire $Ob^{\gamma'}$ (4.72), destiné à être utilisé au moment où l'obligation avec échéance est actée[18]. Cet opérateur est monotone, mais il ne permet pas la propagation des obligations.

$$Ob^{\gamma'}(\varphi, \delta_k) \stackrel{def}{=} X^k \delta_k \wedge Ob\left(\bigvee_{i=0}^{k-1} X^i \varphi\right) \tag{4.72}$$

Cet opérateur intermédiaire (appelons-le une obligation primaire avec échéance) définit l'échéance en tant qu'elle se trouve à k pas de temps du présent, ainsi qu'une obligation que φ soit vraie à un instant entre le présent et cette échéance. Afin d'assurer la propagation des obligations, l'opérateur final Ob^{γ} est défini de la manière suivante :

$$Ob^{\gamma}(\varphi, \delta_k) \stackrel{def}{=} \begin{cases} X^k \delta_k \\ \exists k' \in \mathbb{N}, X^{-k'} Ob^{\gamma'}(\varphi, \delta_k) \wedge \bigwedge_{i=1}^{k'} X^{-i} \neg\varphi \\ \nexists k'' < k, X^{k''} \delta_{k''} \wedge X^{-k'} Ob^{\gamma'}(\varphi, \delta_{k''}) \end{cases} \tag{4.73}$$

[17] Dans la formulation originale, les auteurs s'appuient sur des opérateurs temporels $\mathcal{U}$, G et F indexés par des durées, qui sont présentés comme des extensions au langage original. Néanmoins, dans le cas de durées finies (ce qui est également le cas dans la communication originale), ces opérateurs peuvent être définis comme des abréviations de modalités déjà existantes, c'est pourquoi nous ne les introduirons pas ici, préférant nous reposer sur un langage déjà suffisamment expressif.

[18] Nous ne suivons pas strictement la démarche de construction de l'opérateur suivie par les auteurs, pour des raisons de lisibilité dues aux notations choisies.

L'obligation Ob^{γ} de φ associée à l'échéance δ_k est donc définie par la conjonction des faits suivants :

- δ_k est située à k pas de temps dans le futur ;
- Il y a eu, il y a k' pas de temps, une obligation primaire de φ avec la même échéance, obligation qui n'a pas encore été satisfaite ;
- Il n'y avait pas, à cette même date passée, d'obligation primaire de φ avec une échéance antérieure à δ_k (mais toujours située dans le futur).

Lors de l'utilisation de cet opérateur, l'obligation primaire $Ob^{\gamma'}$ est posée au moment où la norme apparaît, et c'est l'obligation Ob^{γ} qui est propagée par sa définition même.

Le formalisme utilisé pose un problème technique dans le cadre de la logique DLP : il utilise une quantification existentielle explicite sur les durées (du type $\exists k$) qui ne peut être traduite directement par les modalités temporelles que nous nous sommes autorisées, à cause des conditions qui sont posées par la suite sur les variables quantifiées. L'utilisation de l'opérateur Ob^{γ} nécessiterait donc une extension du langage de DLP.

Cette remarque mise à part, voyons comment Ob^{γ} satisfait aux critères que nous avons définis :

1. ■ – La date d'échéance étant définie explicitement par la durée la séparant du présent, elle existe et est unique. On peut donc considérer qu'elle correspond à une proposition-date.
2. ■ – Par construction, l'échéance est bien située dans le futur.
3. ∅ – À cause de la quantification existentielle niée dans la définition de Ob^{γ}, les obligations sur $\top$ ne peuvent être des théorèmes.
4. ■ – À cause de la construction de $Ob^{\gamma'}$ à partir d'une modalité d'obligation immédiate sur une contradiction, les obligations sur $\perp$ sont bien contradictoires.
5. ■ – Par construction, une obligation (que ce soit $Ob^{\gamma'}$ ou Ob^{γ}) ne peut être maintenue après son échéance.
6. □ – Les auteurs ne donnent pas de définition formelle d'une violation de Ob^{γ} dans leur communication originale. Il est néanmoins possible de définir une violation ponctuelle, située à l'échéance, sur la base de leur formalisme.
7. ■ – Le principe de propagation est respecté par construction, Ob^{γ} pouvant être dérivé depuis l'apparition de l'obligation primaire jusqu'à l'échéance ou à la satisfaction de l'obligation.
8. ∅ – L'opérateur final interdit, par construction, toute monotonie (tant dans le cas d'un respect que d'une violation de l'obligation avec échéance).

Synthèse de l'évaluation

Le tableau 4.3 résume le respect de nos huit critères par les trois opérateurs évalués. On constate que les seuls critères partagés par les trois opérateurs sont le fait que l'obligation doive être abandonnée après l'échéance (alors que c'est un choix qui pourrait être débattu dans certains cas de figure) et la propagation (qui semble par contre une propriété fondamentale de l'obligation avec échéance).

Forts de cette évaluation, il nous faut maintenant construire sur cette base un opérateur, adapté à la logique DLP, qui respecte au mieux les huit critères sur lesquels nous avons choisi de travailler.

Critère	Ob^{α}	Ob^{β}	Ob^{γ}
1			■
2	□		■
3	□	■	
4		□	■
5	■	■	■
6		□	□
7	■	■	■
8		□	

TAB. 4.3 – Comparaison d'opérateurs existants pour l'obligation avec échéances

Conception d'un opérateur adapté

Notre évaluation nous amène à penser que l'opérateur Ob^{γ} serait le plus adapté à nos exigences. Néanmoins, nous avons également vu qu'il posait, au même titre que Ob^{α}, des problèmes conceptuels qui le rendaient incompatibles avec une utilisation en l'état dans la logique DLP[19].

En réalité, le seul opérateur dont la transcription en DLP soit satisfaisante est Ob^{β}. Même s'il est loin de remplir tous nos critères, le travail qui nous en sépare est conceptuellement beaucoup plus abordable que pour les deux autres opérateurs. Il est possible en effet d'imposer l'utilisation d'une proposition-date comme échéance, de la fixer dans le futur strict (assurant par là l'inconsistance des obligations sur $\perp$ dans tous les cas de figure) et d'utiliser des violations ponctuelles comme nous avons déjà vu que c'était possible. En conséquence, notre opérateur d'obligation avec échéance $Ob^{\nu}_{a}(\varphi, \delta)$ est défini par la formule (4.74).

$$Ob^{\nu}_{a}(\varphi, \delta) \stackrel{def}{=} \begin{cases} date(\delta) \\ F\delta \\ Ob^{\nu}_{a}(F^{-}(\varphi \wedge F\delta))\ \mathcal{U}^{-}\ (\varphi \vee \delta) \end{cases} \tag{4.74}$$

Concernant la définition d'une violation associée à cet opérateur, nous avons vu qu'il était possible de définir une violation ponctuelle, située dans le temps à l'instant de l'échéance. Pour faire correspondre correctement la violation avec la norme datée qui l'a engendrée, nous paramétrons la violation par la formule sur laquelle porte l'obligation ainsi que par l'échéance (même si elle correspond forcément à l'instant de la violation). Paramétrer la violation par la date à laquelle l'obligation a été mise en place est peu porteur de sens, à cause même du principe de propagation qui nous ferait définir une multitude de violations simultanées. En conséquence, nous définissons la violation $violOb^{\nu}_{a}(\varphi, \delta)$ comme suit :

$$violOb^{\nu}_{a}(\varphi, \delta) \stackrel{def}{=} \begin{cases} date(\delta) \\ \delta \\ P(Ob^{\nu}_{a}(\varphi, \delta) \wedge \neg\varphi\ \mathcal{U}^{-}\delta) \end{cases} \tag{4.75}$$

Une violation de l'obligation de φ à une échéance de δ est donc définie seulement à l'instant δ, si par le passé il y a eu une obligation de φ à une échéance de δ et que φ n'a pas été vraie entre ce moment et δ.

[19] Pour mémoire, Ob^{γ} travaille sur une quantification explicite des durées qui n'existe pas en DLP, et Ob^{α}, parce que non fondée sur une primitive déontique, permet des dérivations abusives.

On peut maintenant caractériser formellement la manière dont l'opérateur $Ob_a^\nu(\varphi, \delta)$ vérifie nos huit critères (à l'exception de la monotonie à laquelle nous préférerons la semi-monotonie pour les motifs déjà exprimés).

Théorème 4.1 (Obligations avec échéances)**.** L'opérateur d'obligation avec échéance $Ob_a^\nu(\varphi, \delta)$ et sa violation $violOb_a^\nu(\varphi, \delta)$ respectent les propriétés suivantes :

1. Les échéances sont des proposition-dates ;
$$Ob_a^\nu(\varphi, \delta) \rightarrow date(\delta)$$
2. Les échéances sont toujours situées dans un futur strict ;
$$Ob_a^\nu(\varphi, \delta) \rightarrow F\delta$$
3. Les obligations sur des tautologies sont toujours dérivables pour des échéances bien définies ;
$$date(\delta) \wedge F\delta \rightarrow Ob_a^\nu(\top, \delta)$$
4. Les obligations sur des contradictions sont des antilogies ;
$$\vdash \neg Ob_a^\nu(\bot, \delta)$$
5. Les obligations sont abandonnées après l'échéance ;
$$\delta \rightarrow G^-\neg Ob_a^\nu(\varphi, \delta)$$
6. Les violations sont ponctuelles ;
$$violOb_a^\nu(\varphi, \delta) \rightarrow \begin{cases} G\neg violOb_a^\nu(\varphi, \delta) \\ H\neg violOb_a^\nu(\varphi, \delta) \end{cases}$$
7. Le principe de propagation est respecté ;
$$Ob_a^\nu(\varphi, \delta) \wedge \neg(\varphi \vee \delta) \wedge \neg X\delta \rightarrow XOb_a^\nu(\varphi, \delta)$$
8. Le principe de semi-monotonie est respecté.
$$Ob_a^\nu(\varphi, \delta) \wedge date(\delta') \wedge F(\delta \wedge F^-\delta') \wedge \neg(\neg\varphi\ \mathcal{U}^-\ \delta) \rightarrow Ob_a^\nu(\varphi, \delta')$$

Démonstration. On pourra observer que la plupart des démonstrations ne sont valides qu'à cause de la nature des propositions-dates que nous avons définies, qui ne sont vraies qu'à un seul instant du flot de temps.

1. Trivial, par définition de l'opérateur.
2. Trivial, par définition de l'opérateur.
3. Si l'on dispose bien de $date(\delta) \wedge F\delta$ (c'est-à-dire si l'échéance est correctement définie), alors $Ob_a^\nu(\top, \delta)$ se réduit à $Ob(F^-(\top \wedge F\delta))\ \mathcal{U}^-\ (\top)$, soit $\top\ \mathcal{U}^-\ \top$, soit $\top$.
4. Par définition, $Ob_a^\nu(\bot, \delta)$ implique $Ob(F^-(\bot))\ \mathcal{U}^-\ (\delta)$. $F^-(\bot)$ implique $F(\bot)$, qui implique $\bot$ à cause de l'axiome (H-D). Par l'axiome (Ob_a^ν-D), l'ensemble se réduit à $\bot\ \mathcal{U}^-\ \delta$, donc, puisque $Ob_a^\nu(\bot, \delta) \rightarrow F\delta$, on obtient $Ob_a^\nu(\bot, \delta) \rightarrow \bot$.
5. $Ob_a^\nu(\varphi, \delta) \rightarrow F\delta$, donc $F^-Ob_a^\nu(\varphi, \delta) \rightarrow F\delta$. À cause de la dualité entre F^- et G^-, $(\neg G^-\neg Ob_a^\nu(\varphi, \delta)) \rightarrow F\delta$. δ étant une proposition-date, $F\delta \rightarrow \neg\delta$ et donc $(\neg G^-\neg Ob_a^\nu(\varphi, \delta)) \rightarrow \neg\delta$. On obtient ensuite l'implication voulue par contraposition.
6. $violOb_a^\nu(\varphi, \delta) \rightarrow (\delta \wedge date(\delta))$, donc la violation n'est dérivable que lorsque la proposition-date associée l'est, donc une seule fois dans le flot de temps (la démonstration formelle utilise également une contraposition).
7. Si l'on accepte l'antécédent de l'implication logique à démontrer :
   - $X\,date(\delta)$ est dérivable, via (4.64) et la règle de nécessitation appliquable à X ;
   - $\neg X\delta \wedge F\delta$ permet de dériver $XF\delta$;

– En conséquence de $\neg(\varphi \vee \delta)$ ($\neg\delta$ pouvant d'ailleurs être dérivé de $Ob_a^\nu(\varphi, \delta)$), la borne du $\mathcal{U}^-$ n'est toujours pas atteinte, donc l'ensemble du $\mathcal{U}^-$ sera vraie à l'instant suivant à cause de la sémantique de l'opérateur.

En conséquence, l'ensemble de l'obligation par échéance est bien propagée à l'instant suivant : $XOb_a^\nu(\varphi, \delta)$.

8. Si l'on accepte l'antécédent de l'implication logique à démontrer :
   – $date(\delta')$ est assuré par l'antécédent ;
   – $F(\delta \wedge F^-\delta') \rightarrow F\delta'$, trivialement (si δ' est dans le futur au sens large de δ qui lui-même est dans un futur strict, alors δ' est dans un futur strict) ;
   – $\neg(\neg\varphi\ \mathcal{U}^-\ \delta)$ implique que φ surviendra avant δ, et donc avant δ'. En conséquence, $Ob_a^\nu(F^-(\varphi \wedge F\delta))\ \mathcal{U}^-\ (\varphi \vee \delta)$ implique $Ob_a^\nu(F^-(\varphi \wedge F\delta))\ \mathcal{U}^-\ (\varphi \vee \delta')$. D'autre part, $(\varphi \wedge F\delta)$ implique $(\varphi \wedge F\delta')$, donc par fermeture des modalités normales sur l'implication logique on obtient bien $Ob_a^\nu(F^-(\varphi \wedge F\delta'))\ \mathcal{U}^-\ (\varphi \vee \delta')$.

   En conséquence, l'antécédent permet bien de déduire $Ob_a^\nu(\varphi, \delta')$.

□

Nous considérons donc maintenant que nous disposons d'un opérateur d'obligations avec échéances $Ob_a^\nu(\varphi, \delta)$ qui constitue une transcription efficace en logique DLP de la notion normative intuitive.

4.3.2.4 Interdictions maintenues sur une période

Un autre opérateur déontique et temporel qui nous sera utile est le maintien d'une interdiction sur une période donnée. On peut rencontrer cette notion dans des retranscriptions de normes du type de celle-ci (complètement fictive, mais se rapportant tout de même au traitement des données personnelles) :

> Un utilisateur doit être informé au moins une semaine à l'avance de tout traitement utilisant son numéro de sécurité sociale.

Si l'on veut utiliser cette norme pour contraindre le fonctionnement de l'agent de service, ce dernier doit en effet pouvoir s'interdire de procéder à un tel traitement pendant une certaine période (dépendant de la date à laquelle il a éventuellement informé l'agent utilisateur).

Plutôt que de définir une interdiction maintenue sur une période, on aurait pu concevoir un opérateur d'obligation maintenue sur une période. Néanmoins, à cause du sens des prédicats liés à la protection des données personnelles, un tel opérateur aurait moins de sens. En effet, les prédicats performatifs de $\mathcal{L}_{DLP}$, sur lesquels sont susceptibles de porter les normes, représentent des actions ponctuelles. Il est donc plus intuitif de considérer la proposition suivant laquelle une telle action est menée ou non à un instant d'un intervalle de temps particulier, plutôt que celle suivant laquelle cette action est menée en continu pendant ce même intervalle de temps. En tout état de cause, les deux notions sont équivalentes à une négation logique près. Le choix que nous faisons a donc une portée limitée à la lisibilité des formules.

Critères pour la correction de l'opérateur

La notion d'interdiction maintenue sur une période est curieusement beaucoup plus simple à formaliser que l'obligation avec échéance. En effet, elle consiste en une conjonction d'obligations immédiates portant sur des instants définis à l'avance : la propagation y est facile à mettre en œuvre et la « pression normative » est inévitablement présente. Le concept a d'ailleurs moins

motivé la communauté scientifique, cet opérateur semblant suffisamment immédiat pour que nous n'ayons pas de propositions variées à comparer.

Nous établissons néanmoins, par souci de symétrie, les critères à respecter pour un opérateur d'interdiction maintenue. Par cohérence avec l'opérateur $Ob_a^\nu(\varphi, \delta)$, nous travaillons sur un opérateur $For_a^\nu(\varphi, \delta)$ travaillant également sur des propositions-dates. La sémantique de l'opérateur serait la suivante : il est interdit, à partir du présent compris et jusqu'à δ non comprise, que φ devienne vraie. Nous ne nous apesantirons pas sur la justification des critères évidents au regard des caractéristiques défendues pour le premier opérateur.

1. **La limite de l'interdiction doit être une proposition-date**.
2. **La limite de l'interdiction doit être située strictement dans le futur**.
3. **Les interdictions sur $\perp$ doivent être des tautologies**.
4. **Les interdictions sur $\top$ doivent être des antilogies**.
5. **Les interdictions violées doivent cependant être maintenues jusqu'à la date limite** : ce point est sujet à débat. Nous choisissons de prendre cette position qui peut sembler incohérente au regard des critères de l'obligation avec échéance car le problème nous semble différent dans sa nature. En effet, dans le cas de l'obligation avec échéance, qu'il y ait eu violation ou pas, rien n'indiquait au départ l'existence d'une quelconque obligation à une date postérieure à l'échéance. Dans le cas de l'interdiction maintenue, en revanche, l'existence de l'interdiction est annoncée dès lors que la norme est actée. C'est pourquoi nous choisissons d'ignorer l'existence de violations dans le maintien des interdictions. Bien évidemment, la date limite échue, aucune norme ne persiste pour autant[20].
6. **Les violations ne sont définies qu'aux instants où la norme est violée** : en effet, ces violations sont potentiellement multiples si φ survient plusieurs fois (ou est maintenue) pendant sa période d'interdiction. Ceci doit pouvoir être détecté. Il n'est donc pas possible d'imposer (comme on serait tenté de le faire par symétrie avec $Ob_a^\nu(\varphi, \delta)$ que les violations ne surviennent qu'une et une seule fois.
7. **Le principe de propagation doit être respecté**. Dans le cadre de l'interdiction maintenue, nous le définissons comme le fait que si une interdiction de φ jusqu'à δ existe et que l'instant suivant n'est pas δ, alors l'interdiction est propagée à l'instant suivant[21] :

$$For_a^\nu(\varphi, \delta) \wedge \neg X\delta \rightarrow XFor_a^\nu(\varphi, \delta) \tag{4.76}$$

8. **Le principe de monotonie doit être respecté**. Ce principe nous dit ici que si une interdiction de φ existe avec une limite δ, alors il existe une interdiction de φ avec une limite antérieure[22] :

$$For_a^\nu(\varphi, \delta) \wedge date(\delta') \wedge F(\delta' \wedge F^-\delta) \rightarrow For_a^\nu(\varphi, \delta') \tag{4.77}$$

[20] Cela signifie que cet opérateur est une interdiction maintenue sur une durée entre deux dates prédéterminées : il ne saurait représenter (sans l'aide d'une modalité temporelle supplémentaire) une norme du type « il est obligatoire que pendant une période de tant (la date de début de la période restant indéterminée), φ reste fausse ».

[21] La trivialité de ce principe comparé à son dual pour l'obligation avec échéance exprime toute la différence d'intérêt qu'il existe entre les deux opérateurs.

[22] À condition cependant qu'il existe au moins un instant entre le présent et δ. DLP ne travaillant pas sur un flot temporel dense, cela a son importance.

Conception d'un opérateur adapté

Nous proposons l'opérateur défini par la formule (4.78) pour représenter le maintien d'une interdiction. La formule (4.79) définit sa violation $violFor_a^\nu(\varphi, \delta)$, qui prend sa valeur de vérité à chaque fois que φ survient dans l'intervalle de son interdiction[23].

$$For_a^\nu(\varphi, \delta) \stackrel{def}{=} \begin{cases} date(\delta) \\ F\delta \\ (For_a^\nu \varphi)\ \mathcal{U}^-\ \delta \end{cases} \tag{4.78}$$

$$violFor_a^\nu(\varphi, \delta) \stackrel{def}{=} \begin{cases} date(\delta) \\ F\delta \\ \varphi \\ P^- For_a^\nu(\varphi, \delta) \end{cases} \tag{4.79}$$

Théorème 4.2 (Interdiction maintenue)**.** L'opérateur d'interdiction maintenue $For_a^\nu(\varphi, \delta)$ et sa violation associée $violFor_a^\nu(\varphi, \delta)$ vérifient les propriétés suivantes :

1. La limite de l'interdiction est une proposition-date ;
$$For_a^\nu(\varphi, \delta) \rightarrow date(\delta)$$
2. La limite de l'interdiction est située strictement dans le futur ;
$$For_a^\nu(\varphi, \delta) \rightarrow F\delta$$
3. Les interdictions sur des contradictions sont toujours dérivables pour des échéances bien définies ;
$$date(\delta) \wedge F\delta \rightarrow For_a^\nu(\bot, \delta)$$
4. Les interdictions sur des tautologies sont des antilogies ;
$$\vdash \neg For_a^\nu(\top, \delta)$$
5. Les interdictions, mêmes violées, sont maintenues jusqu'à la date limite ;
$$P^- For_a^\nu(\varphi, \delta) \wedge F\delta \rightarrow For_a^\nu(\varphi, \delta)$$
6. Les violations ne sont définies qu'aux instants où la norme est violée ;
$$violFor_a^\nu(\varphi, \delta) \rightarrow \varphi$$
7. Le principe de propagation est respecté ;
$$For_a^\nu(\varphi, \delta) \wedge \neg X\delta \rightarrow XFor_a^\nu(\varphi, \delta)$$
8. Le principe de monotonie est respecté.
$$For_a^\nu(\varphi, \delta) \wedge date(\delta') \wedge F(\delta' \wedge F^-\delta) \rightarrow For_a^\nu(\varphi, \delta')$$

Démonstration. Les propositions à démontrer dans le cas de l'opérateur d'interdiction maintenue sont principalement triviales.

1. Trivial, par définition de l'opérateur.
2. Trivial, par définition de l'opérateur.
3. $For_a^\nu \bot$ est un théorème, donc puisque l'on a $F\delta$, alors $(For_a^\nu \bot\ \mathcal{U}^-\ \delta)$ est dérivable. $For_a^\nu(\bot, \delta)$ est donc dérivable sous les conditions fixées.

[23] Cette définition de la violation ne semble pas pleinement satisfaisante. En effet, à cause de la monotonie de l'opérateur d'obligation maintenue, plusieurs violations peuvent éventuellement être dérivées (pour des dates limites δ différentes) pour la même occurence de φ. Ce problème pourrait être contourné en se reposant uniquement sur la violation de l'interdiction immédiate sur laquelle repose l'interdiction maintenue, mais on perdrait alors la caractérisation de la période d'interdiction. Quoi qu'il en soit, un agent a évidemment toute latitude pour raisonner sur le modèle de violation qui lui convient.

4. $\neg For_a^\nu \top$ est un théorème, donc $(For_a^\nu \top\ \mathcal{U}^-\ \delta)$ n'est dérivable que si δ est dérivable. Or $F\delta$ implique $\neg\delta$ pour une proposition-date, donc on a bien la contradiction souhaitée.
5. Trivial de par la sémantique de $\mathcal{U}^-$. Cette propriété découle d'ailleurs du principe de propagation.
6. Trivial, par définition de la violation.
7. De $(For_a^\nu \varphi)\ \mathcal{U}^-\ \delta$, $F\delta$ et $\neg X\delta$ on en déduit $X((For_a^\nu \varphi)\ \mathcal{U}^-\ \delta)$ par la sémantique de $\mathcal{U}^-$. De $F\delta$ et $\neg X\delta$ on en déduit $XF\delta$, et on obtient $X\,date(\delta)$ car $date(\delta)$ est un théorème.
8. $date(\delta')$ et $F\delta'$ proviennent directement des hypothèses. À partir de $(For_a^\nu \varphi)\ \mathcal{U}^-\ \delta$ on obtient aisément $(For_a^\nu \varphi)\ \mathcal{U}^-\ \delta'$ de par la sémantique de $\mathcal{U}^-$.

□

On considère donc maintenant que l'on dispose d'un opérateur $For_a^\nu(\varphi, \delta)$ permettant de décrire l'interdiction pour φ de devenir vraie entre l'instant présent (compris) et l'instant caractérisé par δ (non compris).

4.3.3 Exemples de normes

Nous nous proposons maintenant d'expliciter, en fonction des formalismes que nous avons introduits, la traduction en logique DLP de quelques normes fictives traitant de la protection des données personnelles. Nous nous placerons dans le cas d'un agent PAw (représenté par `self`) appartenant à un employé de la société « groupe W » travaillant en Syldavie. Nous utiliserons les autorités normatives fictives suivantes :

- la législation syldave : `syldavianLaw` ;
- le réglement de la société « groupe W » : `wGroupReg` ;
- le directeur du groupe pour la branche « Syldavie » : `wSylChair`
- le contrat entre M. Pignon et le groupe W : `wPignonContract` ;
- les préférences de l'utilisateur : `user`.

Nous proposons une norme fictive pour chacun des six axes réglementaires et discutons de sa traduction en logique DLP. Pour chaque norme, nous donnerons une ou plusieurs formules DLP. La traduction effective de la norme sera la conjonction de ces formules.

Les normes étant à l'origine exprimées en français, elles sont sujettes à interprétation avant d'être traduites en logique DLP. Cette nécessaire interprétation est sujette à discussion, et nous ne nous prétendons pas compétents pour expliciter tout élément de règlementation d'une manière inattaquable. Nous souhaitons seulement mettre en lumière la nécessité d'une intervention humaine lorsque les normes préexistantes n'ont pas été formulées de manière à être aisément traduites dans un langage formel. Cette étape d'interprétation peut être évacuée lorsque les normes ne proviennent pas d'un texte en langue naturelle mais d'un éditeur de préférences, ce dernier permettant de contraindre la forme des phrases et autorisant un traitement purement automatique.

4.3.3.1 Robustesse des formules au dynamisme du contexte normatif

Comme nous l'avons déjà vu, le contexte normatif dans lequel évolue l'agent PAw est dynamique. De nouvelles normes ou autorités normatives sont susceptibles d'apparaître, d'autres peuvent disparaître. À chacune de ces modifications du contexte normatif (qui sont cependant supposée survenir assez rarement dans le cas où l'agent PAw ne se place pas de lui-même dans un nouveau contexte), la base de normes de l'agent correspondant à ce contexte sera réinitialisée, et

l'ensemble des formules dérivées sera recalculé. En conséquence, les formules DLP représentant les normes doivent prendre en compte le fait que leur existence dans le système logique peut être postérieure à certains faits sur lesquels elles portent. Si l'on prend l'exemple d'une norme d'obligation qui a pour condition d'application le fait qu'un traitement ait été opéré par l'agent dans le passé, la traduction de cette norme doit permettre de dériver l'obligation correspondante même si la base de norme a été réinitialisée entre le moment auquel le traitement a eu lieu et l'instant présent (quitte à rendre partiellement redondante la traduction de la norme[24]). Nous verrons dans les exemples comment prendre cela en compte.

Le dynamisme du contexte normatif entraîne donc une certaine volatilité de la base de normes. En conséquence, la caractérisation des violations effectuées par un agent à des dates déterminées du flot temporel est également volatile. Afin de permettre à l'agent PAw de raisonner tout de même sur ses violations passées (même si les normes qui les ont engendrées ne sont plus en vigueur), nous supposons l'existence d'un procédé de *stigmatisation* des violations : lorsqu'une violation est dérivée, elle est extraite du système logique DLP pour être introduite dans la base de croyances de l'agent avec toutes les informations nécessaires s'y rattachant[25].

L'agent PAw peut ainsi raisonner efficacement dans un contexte normatif dynamique tout en restant conscient de ses violations passées.

4.3.3.2 Exemple sur l'axe « Information »

> *Extrait du règlement intérieur du groupe W :* « Les clients doivent être informés au moins une semaine à l'avance de la nature de tout traitement utilisant leurs données personnelles. »

Cette norme est formulée de manière simple, sous une forme qui peut apparaître couramment dans des règlements ou des contrats. Elle présente néanmoins des difficultés d'interprétation non triviales pour un agent artificiel. Cette obligation concerne en effet directement les capacités de planification de l'agent : il doit la considérer seulement s'il a prévu de mettre en œuvre un traitement utilisant des données personnelles.

La norme vue comme une interdiction

Une première interprétation de cette norme consiste à la voir comme une interdiction de procéder au traitement si la condition n'a pas été respectée. Trois cas sont alors à considérer, suivant les actions déjà menées par l'agent :

1. Le client n'a pas encore été informé : la norme exprime alors une interdiction de procéder au traitement, interdiction dont on sait qu'elle peut être maintenue au moins une semaine, indépendamment des actions futures de l'agent. Il paraît donc naturel ici d'utiliser l'opérateur d'interdiction maintenue. Si nous considérons qu'il faut également exprimer la relation entre le client et le traitement (via le prédicat d'état *owner* qui l'identifie comme propriétaire d'une donnée utilisée dans un traitement particulier), nous représenterons

[24] Il est possible de s'affranchir de cette redondance en introduisant dans les conditions d'application des normes une formule temporelle détectant une éventuelle réinitialisation de la base dans le passé (via l'introduction d'une nouvelle proposition dédiée). Pour ne pas alourdir inutilement des notations déjà complexes, nous nous accomoderons de cette redondance purement formelle.

[25] Il faut toutefois noter que si le procédé de stigmatisation permet un raisonnement sur les violations passées dans la couche cognitive (doxastive) de l'agent PAw, il ne permet pas toujours de fonder des normes sur ces violations. L'utilisation de normes *contrary-to-duties*, par exemple, est donc limité par le dynamisme du contexte normatif.

donc cette situation de la manière suivante en SDL :

$$\left.\begin{array}{l} actionType(\text{ProcessID}, \text{ActionType}) \\ H\neg informActionType(\texttt{self}, \text{Client}, \\ \quad \text{ProcessID}, \text{ActionType}) \\ owner(\text{ProcessID}, \text{DataID}, \text{Client}) \\ date(\delta) \wedge X^{7*24}\delta \end{array}\right\} \rightarrow For_{\texttt{self}}^{\texttt{wGroupReg}}(perform(\texttt{self}, \text{ProcessID}), \delta) \tag{4.80}$$

2. Le client a été informé il y a moins d'une semaine : la norme interdit alors que le traitement ait lieu avant un certain délai, dépendant de la date à laquelle le client a été informé. Si l'on considère que l'agent a toujours été conscient de cette norme, alors on sait qu'à cause du premier cas de figure, une obligation maintenue a été générée et qu'il existe une obligation immédiate de procéder au traitement (et ce jusqu'à une semaine après l'information du client). Cependant, si la norme est récente (*i.e.* si la base de normes a été réinitialisée il y a moins d'une semaine), aucune obligation maintenue n'a été générée. Il paraît donc prudent ici de générer automatiquement une interdiction immédiate, même si potentiellement redondante avec la précédente formule :

$$\left.\begin{array}{l} actionType(\text{ProcessID}, \text{ActionType}) \\ PinformActionType(\texttt{self}, \text{Client}, \\ \quad \text{ProcessID}, \text{ActionType}) \\ X^{-7*24}H^{-}\neg informActionType(\texttt{self}, \text{Client}, \\ \quad \text{ProcessID}, \text{ActionType}) \\ owner(\text{ProcessID}, \text{DataID}, \text{Client}) \end{array}\right\} \rightarrow For_{\texttt{self}}^{\texttt{wGroupReg}} perform(\texttt{self}, \text{ProcessID}) \tag{4.81}$$

3. Le client a été informé il y a une semaine ou plus : la norme n'interdit plus la mise en œuvre du traitement. Il paraîtrait donc naturel dans ce cas de générer une permission immédiate de procéder au traitement. Néanmoins, dans certains cas de figure cela pourrait amener à l'inconsistance du système. En effet, si le traitement en question utilise les données de deux clients et que seul le premier a été informé dans les temps, alors la norme engendrera à la fois une interdiction et une permission sur la même formule. Afin que l'agent puisse raisonner convenablement dans les cas plus complexes, nous nous contenterons donc de ne générer aucune notion déontique dans ce cas d'espèce, qui ne sera donc pas représenté par une formule DLP. Ainsi, dans le cas simple où un seul client, correctement informé, est concerné, la permission (et même la gratuité) de l'action pourra être dérivée au besoin par une négation par l'échec. Dans le cas complexe que nous avons évoqué, l'agent ne verra qu'une interdiction et aucune permission : le système logique restera cohérent et lui interdira de procéder au traitement[26].

La norme vue comme une obligation

La manière complémentaire de considérer cette norme est de la voir d'un point de vue plus téléologique : si l'agent a déjà pour but de procéder au traitement dans un futur identifié, alors la norme lui dicte une obligation d'informer le client. Il peut donc être intéressant de générer des formules d'obligation dans les cas où celles-ci font sens, et ce même si elles sont redondantes avec

[26] Cette prise de position est tout-à-fait cohérente avec la vision philosophique que nous avons des modalités déontiques dans le cadre de l'agent PAw : elles servent à *contraindre* son comportement. Ainsi, une permission (au sens du dual logique de l'obligation) ne lui donne pas un droit à l'action, elle dénote simplement l'absence d'interdiction.

les interdictions conditionnelles déjà établies. En effet, suivant les cas il peut être plus pratique pour l'agent de raisonner à partir d'obligations ou d'interdictions. Encore une fois, plusieurs cas sont à considérer :

1. Le client n'a pas été informé et le traitement est prévu à une semaine ou plus dans le futur. Dans ce cas de figure, il faut signifier à l'agent qu'il a l'obligation d'informer le client avant une certaine date limite δ (située une semaine avant le traitement). La formule suivante permet de caractériser cette date limite et de générer une obligation avec échéance[27] :

$$\left.\begin{array}{l} actionType(\text{ProcessID}, \text{ActionType}) \\ date(\delta) \\ F(\delta \wedge X^{7*24-1} perform(\texttt{self}, \text{ProcessID})) \\ H\neg informActionType(\texttt{self}, \text{Client}, \\ \quad \text{ProcessID}, \text{ActionType}) \\ owner(\text{ProcessID}, \text{DataID}, \text{Client}) \end{array}\right\} \rightarrow Ob_{\texttt{self}}^{\texttt{wGroupReg}}(informActionType(\texttt{self}, \text{Client}, \text{ProcessID}, \text{ActionType}), \delta) \tag{4.82}$$

2. Le client n'a pas été informé et le traitement est prévu à moins d'une semaine dans le futur. Dans ce cas de figure, aucune obligation ne permettra à l'agent de respecter la norme. Le caractère contrevenant de cette intention est déjà caractérisé par la première des interdictions que nous avons générées, nous n'ajoutons donc pas ici de nouvelle formule à la traduction de la norme.
3. Le client a été informé et le traitement est prévu une semaine ou plus après la date de cette information. Ici, l'agent a déjà tout fait pour se conformer aux prescriptions de la norme. Il n'y a pas d'obligation supplémentaire à exprimer, seule une permission pourrait décrire la situation et nous avons déjà vu qu'il est préférable de s'en abstenir. Nous n'ajoutons donc pas de nouvelles formule.
4. Le client a été informé et le traitement est prévu moins d'une semaine après la date de cette information. Encore une fois, aucune obligation ne peut caractériser un comportement de l'agent qui serait compatible avec la norme. Sa violation est caractérisée soit par l'interdiction maintenue qui a été générée à l'instant précédent l'information du client, soit (dans le cas limite d'une réinitialisation récente de la base de normes) par l'interdiction immédiate qui surviendra au moment du traitement. Encore une fois, nous n'ajoutons donc pas de nouvelle formule à la traduction.

4.3.3.3 Exemple sur l'axe « Consentement »

Extrait du contrat entre M. Pignon et le groupe W : « L'accord exprès et préalable de M. Pignon est nécessaire à toute utilisation qui pourrait être faite de son numéro de compte bancaire. »

Ici encore, la norme peut être considérée avec deux optiques différentes : ou bien c'est une restriction sur la mise en œuvre du traitement, ou bien c'est une obligation inhérente à la planification du traitement. Dans le premier cas, on interdit les traitements auxquels M. Pignon

[27] Le -1 peut sembler étrange dans l'exposant de l'opérateur X. Il est dû au fait que dans l'opérateur d'obligation avec échéance, δ est une limite stricte alors que la norme précise que l'intervalle de temps entre information et traitement doit être d'une semaine « au moins », ce qui suggère une inclusion des bornes. Le lecteur pourra vérifier par lui-même d'une part que ce -1 permet de satisfaire à ces exigences, et d'autre part que les autres formules précédemment proposées correspondent bien à cette sémantique temporelle précise de la norme exprimée.

n'a pas déjà consenti (traitements dans lesquels son numéro de compte bancaire est impliqué, bien évidemment[28]) :

$$\left.\begin{array}{l} owner(\text{ProcessID}, \text{Account}, \texttt{pignon}) \\ dataType(\text{ProcessID}, \text{Account}, \\ \quad \texttt{Ecom.Payment.Bank.PayerAccountNumber}) \\ H\neg consent(\texttt{pignon}, \texttt{self}, \text{ProcessID})) \end{array}\right\} \rightarrow For_{\texttt{self}}^{\texttt{wPignonContract}} \\ \qquad perform(\texttt{self}, \text{ProcessID}) \tag{4.83}$$

Dans le second cas, on connaît déjà la date future à laquelle le traitement est prévu. On peut donc générer une obligation avec échéance sur le consentement de M. Pignon. Il est intéressant de noter que l'obligation engage l'agent PAw (`self`), mais que le consentement est une action de M. Pignon. L'agent PAw devra donc, dans sa couche cognitive de planification, tout mettre en œuvre pour que M. Pignon donne son consentement dans les temps (faute de quoi c'est bien `self`, et pas M. Pignon, qui se trouvera en violation de la norme).

$$date(\delta) \wedge F \left\{\begin{array}{l} \delta \\ perform(\texttt{self}, \text{ProcessID}) \\ owner(\text{ProcessID}, \text{Account}, \texttt{pignon}) \\ dataType(\text{ProcessID}, \text{Account}, \\ \quad \texttt{Ecom.Payment.Bank.PayerAccountNumber}) \end{array}\right. \tag{4.84}$$

$$\rightarrow Ob_{\texttt{self}}^{\texttt{wPignonContract}}(consent(\texttt{pignon}, \texttt{self}\text{ProcessID}), \delta)$$

4.3.3.4 Exemple sur l'axe « Mise à jour »

> *Extrait du règlement intérieur du groupe W :* « Il est obligatoire de fournir aux clients un moyen de communication permettant de faire valoir leur droit de rectification des données personnelles les concernant. »

Cette norme traite à la fois de l'axe « Information » et de l'axe « Modification ». Elle illustre bien le type de traitement superficiel des normes en matière de contrôle des données par leurs propriétaires qui a notamment été valorisé en France depuis 1978. La formulation de la norme, exempte de toute notion temporelle, en devient ambiguë : quelle est l'élément déclencheur de l'obligation, sa condition d'application ? En fait, à quel moment le client doit-il être informé ? Dans la traduction de cette norme, nous choisissons de lever cette ambiguïté afin de permettre à un agent PAw de mettre en place un plan d'action de manière déterministe. Nous décidons d'interpréter cette norme comme une obligation d'informer le client avant de lui réclamer une donnée personnelle[29].

Encore une fois, cette norme peut être vue soit comme une interdiction de réclamer une donnée au client sans l'avoir informé de l'agent à contacter en cas de requête de mise à jour (4.85), soit comme une obligation d'informer le client avant un traitement déjà programmé, si

[28] Le type de donnée utilisée ici (comme dans certains autres exemples) est décrit en utilisant la transcription en schéma de données P3P du langage ECML, proposée par le W3C [Wor99].

[29] On pourrait également fonder la condition d'application sur la transmission effective de l'information par le client, mais l'esprit des réglementations en Europe et en France va dans le sens d'une information de l'utilisateur préalable à toute collecte.

cela n'a pas encore été fait (4.86).

$$H^{-}\neg informContact(\texttt{self}, \text{Client}, \text{ProcessID}, \text{Contact}) \rightarrow For^{\texttt{wGroupReg}}_{\texttt{self}} request(\texttt{self}, \text{Client}, \text{ProcessID}, \text{DatatType}, \text{DataID}) \tag{4.85}$$

$$\left.\begin{array}{l} date(\delta) \\ F(\delta \wedge request(\texttt{self}, \text{Client}, \text{ProcessID}, \\ \quad \text{DatatType}, \text{DataID})) \\ H^{-}\neg informContact(\texttt{self}, \text{Client}, \\ \quad \text{ProcessID}, \text{Contact}) \end{array}\right\} \rightarrow Ob^{\texttt{wGroupReg}}_{\texttt{self}}(informContact(\texttt{self}, \text{Client}, \text{ProcessID}, \text{Contact}), \delta) \tag{4.86}$$

4.3.3.5 Exemple sur l'axe « Justification »

Ordre direct du directeur de la branche « Syldavie » : « Vous n'êtes pas autorisé à utiliser votre e-mail professionnel pour des jeux en ligne. »

La norme exprimée ici peut se voir soit comme une interdiction de transmettre son e-mail à un tiers mettant en œuvre un service, soit comme l'interdiction de mettre en place un tel service. Ici il nous semble que le chef de service interdit à ses subordonnés d'utiliser de tels services existant sur Internet avec leurs données professionnelles, c'est donc sur le prédicat performatif *tell* que portera l'interdiction, ici indépendante de toute notion temporelle.

$$\left.\begin{array}{l} dataType(\text{GameID}, \text{Email}, \\ \quad \texttt{user.business-info.online.email}) \\ \\ \left(\begin{array}{c} actionType(\text{GameID}, \texttt{gaming}) \\ \vee \\ actionType(\text{GameID}, \texttt{gambling}) \end{array}\right) \end{array}\right\} \rightarrow For^{\texttt{wSylChair}}_{\texttt{self}} tell(\texttt{self}, \text{ThirdParty}, \text{GameID}, \text{Email}) \tag{4.87}$$

4.3.3.6 Exemple sur l'axe « Transmission »

Extrait du règlement intérieur du groupe W : « Il est interdit de vendre les adresses de courrier électronique personnelles de nos clients à la société *MarxBros Inc.* »

Dans ce cas également on se trouve devant une norme indépendante de toute notion temporelle. Nous ferons donc en sorte de pouvoir déduire des interdictions immédiates, comme dans le cas de l'exemple précédent. Pour rappel, le concept de transmission des informations à la société tierce se modélise en DLP par une mise en relation de deux données utilisées dans deux traitements différents via le prédicat performatif *forward*.

Un souci d'interprétation demeure, celui de la notion de « client ». En effet, le langage DLP ne permet pas d'associer une formule à cette sémantique. D'une manière générale, l'expression de normes très dépendantes de notions liées aux spécificité d'un scénario nécessite de se reposer sur des ontologies *ad hoc* et d'étendre le langage de prédicats pour les représenter. Nous supposerons donc ici que le schéma propositionnel Client représente l'ensemble des propositions décrivant les agents identifiés comme des clients, et ce par des moyens externes à DLP.

$$\left.\begin{array}{l} responsible(\texttt{self}, \text{Process1}) \\ responsible(\texttt{marxBros}, \text{Process2}) \\ dataType(\text{Process1}, \text{Email1}, \\ \quad \texttt{user.home-info.online.email}) \\ owner(\text{Process1}, \text{Email1}, \text{Client}) \end{array}\right\} \rightarrow For^{\texttt{wGroupReg}}_{\texttt{self}} forward(\text{Process1}, \text{Email1}, \text{Process2}, \text{Email2}) \tag{4.88}$$

4.3.3.7 Exemple sur l'axe « Rétention »

Extrait du code du commerce syldave : « Il est interdit de conserver le numéro de carte de crédit d'un tiers plus d'une semaine après la transaction pour laquelle elle a été utilisée. »

Cette norme est relativement simple à traiter. La mise en œuvre d'un traitement déclenche simplement une obligation avec échéance de détruire (d'oublier) une donnée personnelle[30]. On remarquera que comme la norme parle de « tiers », il est nécessaire de s'assurer que ce n'est pas l'agent lui-même qui est le propriétaire de l'information à supprimer. En effet, dans ce cas, l'obligation de suppression ne s'appliquerait pas.

$$\left.\begin{array}{l} \mathit{perform}(\texttt{self}, \text{ProcessID}) \\ \mathit{date}(\delta) \\ X^{7*24+1}\delta \\ \mathit{owner}(\text{ProcessID}, \text{CreditCard}, \text{Tiers}) \\ \neg\mathit{owner}(\text{ProcessID}, \text{CreditCard}, \texttt{self}) \\ \mathit{dataType}(\text{ProcessID}, \text{CreditCard}, \\ \quad \texttt{Ecom.Payment.Card.Number}) \end{array}\right\} \rightarrow Ob^{\texttt{syldavianLaw}}_{\texttt{self}}(\mathit{forget}(\texttt{self}, \text{ProcessID}, \text{CreditCard}), \delta) \tag{4.89}$$

Il nous faut également considérer le cas de figure dans lequel la base de norme a été réinitialisée récemment, mais qu'un traitement utilisant le numéro de carte de crédit d'un client a été réalisé il y a moins d'une semaine. Dans ce cas, nous produisons une nouvelle obligation avec échéance, sur la base d'une date située une semaine après ledit traitement :

$$\left.\begin{array}{l} \mathit{date}(\delta) \\ P(\mathit{perform}(\texttt{self}, \text{ProcessID}) \wedge X^{7*24+1}\delta) \\ F\delta \\ \mathit{owner}(\text{ProcessID}, \text{CreditCard}, \text{Tiers}) \\ \neg\mathit{owner}(\text{ProcessID}, \text{CreditCard}, \texttt{self}) \\ \mathit{dataType}(\text{ProcessID}, \text{CreditCard}, \\ \quad \texttt{Ecom.Payment.Card.Number}) \end{array}\right\} \rightarrow Ob^{\texttt{syldavianLaw}}_{\texttt{self}}(\mathit{forget}(\texttt{self}, \text{ProcessID}, \text{CreditCard}), \delta) \tag{4.90}$$

Enfin, si le traitement a eu lieu il y a plus d'une semaine, toute violation ayant pu survenir, avant ou après une éventuelle réinitialisation de la base de normes, a déjà été stigmatisée dans la base de croyances de l'agent PAw.

4.3.4 Synthèse

Nous avons montré comment le formalisme de la logique DLP pouvait être utilisé pour représenter des normes traitant de la protection des données personnelles. Nous avons mis en avant la grande expressivité du langage, permettant la définition d'opérateurs déontiques datés autorisant la représentation correcte de concepts complexes. Nous avons vu comment les politiques P3P pouvaient être utilisées pour extraire des informations sur les traitements, et ainsi servir à la construction de formules DLP, tout en pointant les objectifs différents poursuivis par ce standard W3C. Nous avons enfin montré que les normes exprimées en langage naturel devaient être interprétées par un expert humain afin de limiter leur ambiguïté, avant de les traduire de manière satisfaisante en normes DLP.

[30] Encore une fois, le +1 dans l'exposant de l'opérateur X est dû au fait que l'échéance de l'opérateur daté est stricte, alors que la norme semble autoriser une conservation des données pendant exactement une semaine.

4.4 Conflits d'obligations

Comme nous l'avons déjà suggéré, les ensembles de normes DLP provenant d'autorités normatives distinctes peuvent comporter des incohérences, empêchant ainsi un agent cognitif de construire un modèle comportemental adapté de manière déterministe. Nous allons donc maintenant voir comment traiter ce que nous appellerons des *conflits d'obligation* afin d'arriver à un ensemble de normes cohérent et directement utilisable par l'agent PAw. Nos travaux sur la définition, la détection et l'arbitrage des conflits d'obligations ont été partiellement présentés à l'occasion d'une communication scientifique [PD08c].

4.4.1 Aperçu d'approches existantes

Plusieurs travaux traitent de la gestion des conflits normatifs. On peut citer les approches de Laurence Cholvy et Frédéric Cuppens sur la résolution des incohérences dans une politique de sécurité hétérogène par la fusion de rôles [CC97], avec une approche fondée sur l'origine des normes [CC98] ou encore par l'algorithme dit de SOL-résolution [Cho99].

Martin J. Kollingbaum, Wamberto Weber Vasconcelos, Andrés García-Camino et Timothy J. Norman ont proposé en 2007 une méthode de résolution de conflits normatifs utilisant l'unification logique et la résolution de contraintes [KVGC07]. Comme dans nombre de travaux, les aspects normatifs ne sont pas traités en se basant sur l'inférence déontique modale, et seuls les conflits binaires (mettant en jeu deux normes uniquement) sont traités. Néanmoins, le formalisme proposé, en logique du premier ordre, est suffisamment expressif pour permettre d'exprimer des politiques de résolution complexes et qui pourront être proches de celles que nous proposerons.

Récemment, Nir Oren, Michael Luck, Simon Miles et Timothy J. Norman ont présenté une évaluation de plusieurs heuristiques visant à arbitrer des conflits binaires dans des ensembles d'obligations atomiques, représentés comme des problèmes d'argumentation [OLMN08]. Ces travaux comparent trois méthodes, respectivement fondées sur l'abandon aléatoire de normes, la recherche d'extensions préférées et la recherche d'un ensemble non-conflictuel maximal. Les conclusions montrent que cette dernière méthode est celle qui permet de désactiver le moins de normes, et donc potentiellement qui permet à un agent d'éviter le plus grand nombre de violations.

4.4.2 Insuffisance de la notion existante de conflit normatif

Les solutions déjà proposées pour résoudre les conflits normatifs dans un ensemble de normes ne conviennent pas à notre problème, à cause de l'utilisation que nous faisons des normes, incompatible avec la notion de conflit telle qu'elle est couramment utilisée.

Dans tous les travaux existants, le conflit normatif est défini sur la base de l'inconsistance d'un système de logique déontique fondé sur une axiomatique KD. Ceci veut notamment dire que les ensembles de normes suivants sont des conflits normatifs :

- Une obligation et une interdiction sur la même formule : $\{Ob\ \varphi, Ob\ \neg\varphi\} \vdash \bot$;
- Une interdiction et une permission sur la même formule : $\{For\varphi, Per\ \varphi\} \vdash \bot$;
- Une obligation et sa négation (ce qui revient au même que le cas précédent) : $\{Ob\ \varphi, \neg Ob\ \varphi\} \vdash \bot$.

Le premier exemple est absolument indiscutable. Un agent obligé à la fois à une chose est son contraire est dans une situation inextricable, il n'a aucun moyen de dériver un plan d'action cohérent en se fondant sur l'ensemble de normes qui lui est fourni. Il est dans une situation de

dilemme, et c'est ce terme que nous utiliserons pour désigner ce type de problème[31].

Les deux derniers exemples sont plus discutables dans le cas d'une application de la logique déontique à un agent PAw. Supposons qu'une autorité ν_1 autorise l'agent à utiliser une information donnée alors qu'une autorité ν_2 le lui interdise. L'agent ne se trouve pas alors dans une situation désespérée, même d'un point de vue strictement logique. Tout d'abord, nous avons vu que comme les modalités d'obligation sont différenciées par autorité normative, nous pouvons représenter cette situation sans déclencher l'incohérence du système, puisque les deux opérateurs $Ob_a^{\nu_1}$ et $Ob_a^{\nu_2}$ sont indépendants (aucun axiome ne les lie l'un à l'autre). D'un point de vue pratique ensuite, il reste un moyen à l'agent PAw de respecter l'ensemble des normes formulées : il lui suffit de ne pas utiliser l'information en question. Il respecte bien ainsi l'obligation et ne viole pas la permission. En effet, une permission ne peut être violée car elle ne contraint pas le fonctionnement de l'agent.

Étant placés dans une situation où la notion de permission n'est pas un droit garanti à l'agent mais uniquement une absence de contrainte, il paraît naturel de considérer que les deux derniers cas de conflits normatifs que nous avons évoqués ne devraient pas être considérés comme un problème grave par l'agent PAw : les contraintes sur son comportement restent suffisamment lâches pour lui permettre d'établir un plan d'action cohérent.

Cependant, il faut bien noter que l'on se trouve ici dans un cas où les normes émanent de deux autorités différentes. Si l'on considère deux normes $For_a^\nu\ \varphi$ et $Per_a^\nu\ \varphi$ émanant d'une même autorité, l'axiome (Ob_a^ν-D) amène à une inconsistance du système logique. C'est une conséquence du fait que l'autorité normative ν, supposée représenter un idéal de comportement cohérent, a émis des normes qui ne le sont pas. En ce cas, même si l'agent PAw peut éventuellement respecter l'ensemble des normes, nous considérerons que l'autorité ν est devenue incohérente et qu'elle ne doit plus être prise en compte par l'agent (et ce jusqu'à ce qu'elle publie de nouveau un ensemble de normes exempt de tout conflit normatif).

Afin de représenter cette nuance entre les conflits normatifs usuellement utilisés et les limitations que nous avons posées à l'inconsistance logique, nous allons préciser le concept de *conflit normatif* en DLP et introduire celui de *conflit d'obligations*.

4.4.3 Définitions

4.4.3.1 Conflits normatifs

En logique DLP, nous considérons qu'il y a conflit normatif dans les cas restreints où une autorité en particulier devient incohérente. Ces cas engendrent forcément l'inconsistance du système, mais la définition suivante vise à se limiter aux inconsistances dues aux notions normatives. En effet, une formule strictement temporelle peut rendre le système inconsistant mais cela ne doit pas pour autant être considéré comme un conflit normatif.

Définition 4.7 (Conflits normatifs en DLP)**.** Un ensemble de formules DLP contient un conflit normatif si et seulement si :

- Il permet de dériver $\perp$ à l'aide de l'axiomatique de la logique DLP ;
- Dans toute dérivation possible de $\perp$ à partir de cet ensemble, l'un des axiomes (Ob_a^ν-K) ou (Ob_a^ν-D) est impliqué.

[31] La précision lexicale a ici son importance. Certains auteurs réservent le terme de dilemme à l'obligation simultanée d'une formule et de sa négation, comme nous ici, alors que d'autres l'utilisent pour couvrir toute situation donnant lieu à une inconsistance en logique déontique standard.

4.4.3.2 Conflits d'obligations

La définition qu'il nous faut poser pour le conflit d'obligations doit donc respecter les deux principes que nous avons intuitivement dégagés : les dilemmes sont des conflits d'obligations, mais les conflits normatifs reposant nécessairement sur des permissions n'en sont pas. Pour arriver à ce résultat, nous utilisons une logique de transition, que nous nommerons DLP+. Dans la logique DLP, les conflits d'obligations ne devront pas déclencher l'inconsistance du système. Par contre, nous construirons la logique DLP+ de manière à rendre le nouveau système inconsistant en cas de conflit d'obligations. Ainsi, la caractérisation des conflits d'obligations pourra être décrite ainsi : si un ensemble de normes est consistant en DLP mais inconsistant en DLP+, alors il comporte un conflit d'obligations.

Définition 4.8 (Logique DLP+)**.** La logique DLP+ est définie de la même manière que la logique DLP, et augmentée de l'axiome d'absence de conflit d'obligations :

$$\bigwedge_{i=1}^{n} Ob_a^{\nu_i}\ \varphi_i \rightarrow Per_a^{\nu_{i+1}} \bigwedge_{i=1}^{n} \varphi_i \qquad \text{(DLP+-AC)}$$

Ce schéma axiomatique, qui est une adaptation de celui présenté en 2005 par Philippe Balbiani [Bal05] permet de couvrir les conflits mettant en cause deux autorités, mais également davantage. Pour $n = 2$ et $n = 3$, respectivement, on obtient par exemple les schémas axiomatiques suivants (s'entendant pour toutes autorités normatives ν_1, ν_2, ν_3 et ν_4) :

$$Ob_a^{\nu_1}\ \varphi_1 \wedge Ob_a^{\nu_2}\ \varphi_2 \rightarrow Per_a^{\nu 3}(\varphi_1 \wedge \varphi_2) \tag{4.91}$$

$$Ob_a^{\nu_1}\ \varphi_1 \wedge Ob_a^{\nu_2}\ \varphi_2 \wedge Ob_a^{\nu_3}\ \varphi_3 \rightarrow Per_a^{\nu 4}(\varphi_1 \wedge \varphi_2 \wedge \varphi_3) \tag{4.92}$$

Il est donc possible de définir des conflits plus complexes que les contraintes binaires examinées par Oren *et al.* [OLMN08], par exemple. À l'aide de cette logique intermédiaire DLP+, nous pouvons maintenant caractériser la présence de conflits d'obligations en DLP.

Définition 4.9 (Conflits d'obligations en DLP)**.** Un ensemble de formules DLP exempt de conflits normatifs comporte des conflits d'obligation si et seulement si ce même ensemble de formule est inconsistant au sens de la logique DLP+.

On remarquera que l'on ne s'intéresse pas à la notion de conflit d'obligations lorsque l'on se trouve en présence d'un conflit normatif. En effet, ceux-ci rendent le système inconsistant dans son ensemble et doivent être traités avant tout raisonnement plus poussé sur les incohérences entre autorités normatives.

On arrive donc bien à une caractérisation des notions intuitivements proposées : un ensemble de normes comporte un conflit d'obligations si l'une des autorités normatives présentes dans le système interdit ce que les autres obligent.

4.4.4 Détection des conflits d'obligations

Nous avons donné une caractérisation purement formelle, axiomatique des conflits d'obligations. Si nous supposons disposer d'une procédure permettant de décider la satisfaisabilité d'une formule du langage ($\perp$ en l'occurence), alors nous savons théoriquement détecter les conflits d'obligations, ainsi que l'incohérence d'une autorité normative. Cependant, cette hypothèse implique que la procédure en question prenne en compte tous les axiomes et règles d'inférence des logiques modales entrant en jeu dans DLP, ce qui n'est pas chose facile. En effet, la décidabilité

n'est pas acquise (même si elle reste fortement plausible pour la formule $\perp$ en particulier), et la complexité du problème de satisfaisabilité est souvent au minimum PSPACE-difficile. Les logiques déontiques et temporelles dotées d'axiomes de conversion peuvent atteindre des complexités NEXPTIME-complètes.

Par conséquent, lorsque nous mettrons en œuvre une procédure de décision, celle-ci sera sans doute une réduction de cette procédure de décision théorique. Ses capacités d'inférence modale (et donc de détection des conflits d'obligations) seront donc limités. Il nous faudra donc alors caractériser et évaluer cette limitation.

4.4.5 Arbitrage des conflits d'obligations

Nous disposons maintenant des moyens théoriques pour détecter si un ensemble de normes Δ contient des conflits d'obligations. L'objectif de l'agent PAw à ce stade est alors d'*arbitrer* ces conflits, c'est-à-dire de décider lesquelles de ces normes doivent être conservées, lesquelles doivent être ignorées et lesquelles, éventuellement, doivent être modifiées, afin d'obtenir un nouvel ensemble de normes Δ' exempt de conflits. À partir d'un tel ensemble Δ', l'agent PAw pourrait en effet adapter son comportement de manière cohérente pour respecter les normes, en ayant la garantie de pouvoir n'en violer aucune[32].

Nous allons maintenant voir comment il est possible de construire un ensemble de normes exempt de conflits, en proposant une procédure d'arbitrage fondée sur une relation entre les différentes autorités normatives du contexte de l'agent PAw.

4.4.5.1 Principe des arbitrages de conflits

S'il existe un conflit d'obligations dans l'ensemble Δ, c'est qu'un ou plusieurs sous-ensembles de Δ permettent de dériver $\perp$ à l'aide des règles d'inférence et des axiomes de la logique DLP+. Nous avons ici intérêt à détecter les *sous-ensembles conflictuels minimaux*, qui sont tels que si une seule des normes les composant en était retirée, le conflit disparaîtrait.

L'idée générale de l'arbitrage est donc la suivante : pour chaque sous-ensemble conflictuel minimal identifié dans Δ, l'agent PAw choisit de désactiver une norme. Lorsqu'une norme a été désactivée dans chacun de ces sous-ensembles, alors l'agent dispose d'un ensemble Δ' exempt de conflits d'obligations[33].

Le point de choix majeur est donc ici la procédure de décision, que nous appelerons *politique d'arbitrage*, qui permettra à l'agent d'identifier la norme à désactiver dans chacun des sous-ensembles conflictuels minimaux.

Kollingbaum *et al.* ont mis en lumière trois politiques pouvant être utilisées pour l'arbitrage des conflits normatifs, en se basant sur les pratiques juridiques les plus courantes [KVGC07] :

- *legis posterior* : cette politique stipule que la loi (la norme) la plus récente est prioritaire (et doit donc être conservée) ;
- *legis specialis* : cette politique stipule que la loi la plus spécialisée (celle considérée comme pouvant constituer une exception à une règle générale) est prioritaire ;

[32] Ou tout du moins la garantie que si une norme de Δ' est violée, ce ne sera pas une conséquence des autres normes de l'ensemble. En effet, des circonstances externes pourraient contraindre l'agent PAw à violer certaines normes. L'arbitrage des conflits et l'élaboration d'un ensemble de normes cohérent ne le prémunira pas contre une telle éventualité.

[33] Nous ne donnons ici que l'idée générale de l'algorithme d'arbitrage. Une mise en œuvre complète doit prendre en compte le fait que les sous-ensembles conflictuels minimaux ne sont pas forcément indépendants les uns des autres, et donc procéder à une réévaluation de ces sous-ensembles après chaque désactivation de norme.

– *legis superior* : cette politique stipule que la loi émanant de la plus haute autorité (la norme édictée par l'autorité normative considérée comme la plus importante) est prioritaire.

Le formalisme de la logique DLP ne nous autorise pas à mettre en place une politique de type *legis posterior*, qui supposerait de toute manière que la dernière autorité normative connue aurait priorité sur les autres. Dans un environnement ouvert, où les croyances d'un agent PAw sont limitées et incomplètes, cette politique ne semble de toute manière pas judicieuse. La politique de *legis specialis* est celle qui est tout naturellement mise en place par défaut dans les logiques défaisables, qui représentent bien la notion d'exception ou de cas particulier [Nut01]. Cependant, il paraît clair qu'un conflit entre plusieurs normes émanant de plusieurs autorités indépendantes ne peut être considéré comme la coexistence d'une règle générale et d'une exception. En effet, la règle d'exception, pour être considérée comme telle, devrait être formulée par la même autorité qui a édicté la règle générale (ou bien par une autorité jouissant auprès de cette dernière d'un statut particulier et reconnu). La politique de *legis superior* semble plus adaptée à la résolution des conflits d'obligations entre autorités distinctes, car elle permet à l'agent PAw de raisonner sur les relations explicites qu'il entretient déjà avec les autorités. Nous pensons qu'un agent humain, placé dans la situation d'un conflit d'obligations, aura sans doute tendance à se fonder sur l'importance de l'autorité de la norme pour en déduire sa priorité, privilégiant par exemple la loi de son pays d'origine à la loi d'un pays étranger, ou une clause contractuelle à un texte réglementaire, et ce selon sa sensibilité et sa relation aux diverses autorités normatives en jeu[34].

Il nous faut donc définir, à l'aide d'une nouvelle relation binaire que nous appelerons *prévalence normative*, le fait qu'une autorité soit considérée par un agent PAw comme plus importante qu'une autre.

4.4.5.2 Notion de prévalence normative

Nous introduisons donc le concept de prévalence normative, que nous notons à l'aide de l'opérateur $\lhd_a$.

Définition 4.10 (Prévalence normative)**.** On dit qu'une autorité normative ν_1 prévaut, pour l'agent a, sur l'autorité normative ν_2 (et on notera $\nu_1 \lhd_a \nu_2$), lorsque l'agent considère que les normes émanant de l'autorité ν_1 doivent être respectées prioritairement aux normes émanant de l'autorité ν_2.

Si l'on considère chaque norme comme étant rattachée à une autorité normative, on pourra également parler, pour des raisons de lisibilité, de la prévalence des normes en se référant à la prévalence des autorités correspondantes. Il est important de noter que la prévalence normative est subjective. Deux agents humains placés dans la même situation de conflit d'obligations ne l'arbitreront pas forcément de la même manière, chacun ayant un point de vue différent sur les autorités du contexte normatif. Pour refléter cette subjectivité, chaque agent PAw dispose de sa propre définition de la relation de prévalence normative, et en conséquence, l'opérateur associé est indexé par l'identifiant de l'agent. L'autonomie de l'agent PAw, que nous avons mise en avant dans notre définition d'un système multi-agent normatif, prend donc corps ici dans la relative liberté dont il dispose dans le cadre de l'arbitrage des conflits. Cette liberté (et cette autonomie)

[34] Notons que d'autres politiques pourraient être considérées, fondées par exemple sur des préférences entre actions (pour nous des formules de $\mathcal{L}_{DLP}$) ou entre mondes sémantiques, comme le suggère Sven Ove Hansson [Han02]. Cela impliquerait toutefois l'existence d'un point de vue spécifique de l'agent sur la qualité, l'utilité, la désirabilité d'une action ou d'une situation, ce qui reviendrait à considérer une nouvelle autorité normative. Il semble donc que ce type de politique puisse être simulé par une politique de *legis superior* appliqué à de plus nombreuses autorités, bien que cette intuition mérite sans doute une étude plus approfondie.

n'est que relative, car dans le but que les agissements de l'agent PAw reflètent les intentions de son utilisateur humain (condition nécessaire pour engager la responsabilité juridique de ce dernier, comme le souligne par exemple Manuel Martinez Ribas [Mar03, Mar]), cette relation de prévalence normative est définie par l'utilisateur humain et ne peut être modifiée par l'agent lui-même.

À cette étape, l'utilisateur humain peut se placer lui-même (en tant que représenté par l'autorité normative `self`) à n'importe quel niveau dans l'échelle de la prévalence normative. Nous choisissons de poser sur la relation binaire $\lhd_a$ une hypothèse simplificatrice en la traitant comme une relation de préordre total. Ainsi l'on considère que tout couple d'autorités normatives peut être comparé à l'aide de cette relation (4.93), que la relation est réflexive (4.94) et transitive (4.95).

$$\forall \nu_1, \nu_2 \in \mathcal{N},\ \nu_1 \lhd_a \nu_2 \text{ ou } \nu_2 \lhd_a \nu_1 \tag{4.93}$$

$$\forall \nu \in \mathcal{N},\ \nu \lhd_a \nu \tag{4.94}$$

$$\forall \nu_1, \nu_2, \nu_3 \in \mathcal{N},\ \text{si } \nu_1 \lhd_a \nu_2 \text{ et } \nu_2 \lhd_a \nu_3 \text{ alors } \nu_1 \lhd_a \nu_3 \tag{4.95}$$

Notre hypothèse ne va pas jusqu'à considérer la relation comme un ordre (ce qui impliquerait l'antisymétrie), car on estime que l'on peut mettre deux autorités normatives sur un pied d'égalité. Dans le cadre d'une application distribuée mettant en jeu des agents PAw dépendant de deux États distincts par exemple, les agents peuvent choisir de considérer les deux systèmes législatifs associés comme aussi importants l'un que l'autre.

Nous pensons que si l'hypothèse du préordre total doit être attaquée, elle doit l'être en priorité sur la propriété de transitivité, avant de l'être sur son caractère total. En effet, lorsque l'on considère des relations de préférences exprimées par des agents humains, on n'a en général pas de problème à faire exprimer une préférence pour tout couple donné (surtout si l'on autorise la relation de préférence à ne pas être antisymétrique), alors que l'ensemble des couples exprimés ne satisfait pas toujours la propriété de transitivité. C'est un problème comparable à celui que l'on retrouve en théorie de la décision dans le cas des préférences multimodales. Néanmoins, cette hypothèse de transitivité semble pour l'instant nécessaire au fondement d'une politique d'arbitrage cohérente, qui doit pouvoir identifier les autorités qui sont des minima de la relation de prévalence normative.

4.4.5.3 Pseudo-algorithme d'arbitrage

Nous présentons ici un pseudo-algorithme de principe permettant d'arbitrer les conflits présents dans un ensemble de normes Δ afin d'obtenir un ensemble Δ' exempt de conflits (algorithme 1). Cet algorithme utilise une primitive *ssensemble_conflictuel_minimal*, qui extrait un sous-ensemble conflictuel minimal à partir d'un ensemble de normes conflictuel. Si, comme on l'a supposé, on peut s'appuyer sur une procédure de décision pour la satisfaisabilité des formules DLP et DLP+, il est aisé de concevoir une version naïve de cette primitive. On utilisera également une primitive de tri, paramétrée par une relation binaire qui sera ici la prévalence normative.

Le principe de l'algorithme d'arbitrage est le suivant : on initialise Δ' à partir de l'ensemble Δ, et pour chaque sous-ensemble conflictuel minimal encore présent dans Δ', on supprime une norme issue d'une autorité normative de prévalence minimale. On s'assure ainsi que les normes désactivées seront en nombre minimal, et qu'à chaque fois c'est la norme de prévalence minimale qui sera désactivée. Cet algorithme revient donc intuitivement à la recherche d'un ensemble

consistant maximal dans la logique DLP+, assortie d'une politique de priorité utilisant la notion de prévalence.

Algorithme 1 arbitrer(Δ)

Précondition : Δ est un ensemble fini de formules DLP
Postcondition : Δ' est un ensemble de formules DLP exempt de conflits d'obligations
  $\Delta' \leftarrow \Delta$
  tant que $\Delta' \vdash_{DLP+} \bot$ **faire**
    $ListeConflit \leftarrow ssensemble_conflictuel_minimal(\Delta')$
    $ListeConflit \leftarrow trier(ListeConflit, \lhd_a)$
    $\Delta' \leftarrow \Delta' \backslash \{ListeConflit[0]\}$
  fin tant que
  retourner Δ'

Au sein d'un même sous-ensemble conflictuel minimal, il peut y avoir plusieurs normes issues d'une même autorité. En outre, plusieurs autorités peuvent être de prévalence normative comparable (on peut avoir, pour ν_1 et ν_2 distincts, $\nu_1 \lhd_a \nu_2$ et $\nu_2 \lhd_a \nu_1$). Pour ces raisons, le choix de la norme à désactiver n'est pas forcément déterministe : il peut y avoir plusieurs normes de prévalence minimale dans le sous-ensemble conflictuel minimal considéré. On considère alors que le choix de la norme à désactiver est sans importance. Dans le pseudo-algorithme tel qu'il est présenté, ce choix est déterminé par les détails de conception des primitives *ssensemble_conflictuel_minimal* et *trier*.

La terminaison de l'algorithme est triviale à démontrer : l'ensemble de départ est fini, et à chaque passage dans la boucle un élément en est retiré. L'ensemble vide étant exempt de conflit, le nombre de passages dans la boucle est donc borné par le cardinal de Δ.

4.4.5.4 Spécificité de la procédure d'arbitrage proposée

Le pseudo-algorithme que nous proposons se distingue de celui proposé par Kollingbaum *et al.* [KVGC07] en ceci qu'au lieu d'ajouter une à une des formules dans un ensemble initialement vide, on les retire d'un ensemble initialement plein. On perd ainsi un invariant (la cohérence de l'ensemble en cours de construction), mais cela permet de prendre en compte l'inférence modale dans la détection des ensembles conflictuels. En effet, Kollingbaum *et al.* ne considèrent que des conflits binaires : à chaque inclusion, la formule candidate est comparée deux à deux à toutes les formules déjà présentes pour détecter ces incompatibilités et les arbitrer. Notre pseudo-algorithme, quoique purement théorique pour l'instant, permet de prendre en compte des conflits mettant en jeu un nombre plus élevé de normes, pour lesquels les incompatibilités binaires ne peuvent être détectées qu'après un travail de dérivation. Il serait sans doute possible de mettre en œuvre notre pseudo-algorithme d'arbitrage à l'aide de celui de Kollingbaum *et al.*, après une phase de génération de nouvelles formules (impliquées dans des conflits binaires) à partir des normes initiales (impliquées dans des ensembles conflictuels). Nous nous situons donc à un niveau d'abstraction supérieur, nous autorisant ainsi à raisonner directement sur les normes originales et non sur leurs conséquences, de manière à pouvoir éventuellement prodiguer à l'utilisateur des explications plus facilement compréhensibles sur les arbitrages. Nous nous plaçons donc dans un modèle compatible avec l'idée défendue par Jan Broersen, Mehdi Dastani, Joris Hulstijn, Zisheng Huang et Leendert van der Torre suivant laquelle la prise en compte des conséquences des obligations exprimées est essentielle à la résolution d'un conflit normatif [BDH+01].

4.4.5.5 Exemple

Nous proposons un exemple simple mettant en œuvre la procédure d'arbitrage des conflits d'obligations. Dans cet exemple, on considérera des normes édictées sous la forme d'une seule modalité déontique portant sur une formule du fragment temporel de DLP[35].

Situation

Plaçons-nous dans le cas où un agent PAw `a` considère un traitement `process` auquel son utilisateur (l'autorité `self`) lui demande de procéder immédiatement. Ce traitement utilise une donnée `data`, de type `datatype`. Cette activité sera représenté dans notre exemple par la variable φ, définie comme suit :

$$\varphi \stackrel{def}{=} \begin{cases} \mathit{perform}(\mathtt{a}, \mathtt{process}) \\ \mathit{datatype}(\mathtt{process}, \mathtt{data}, \mathtt{datatype}) \end{cases} \tag{4.96}$$

La requête de l'utilisateur s'exprime donc comme l'obligation $Ob_{\mathtt{a}}^{\mathtt{self}}\ \varphi$. Nous représenterons ensuite par ψ l'éventuelle action de suppression de la donnée `data` par l'agent `a` dans le futur :

$$\psi \stackrel{def}{=} F\ \mathit{forget}(\mathtt{a}, \mathtt{process}, \mathtt{data}) \tag{4.97}$$

Nous supposons dans notre exemple que l'utilisateur interdit à l'agent, pour des raisons qui lui sont propres, de supprimer cette donnée dans le futur. Cette norme se notera donc $For_{\mathtt{a}}^{\mathtt{self}}\psi$.

Nous supposons maintenant que l'agent `a` est placé dans un contexte normatif comprenant une autorité `syldavianLaw`, représentant la législation syldave, à laquelle l'agent est soumis. Nous supposons que les schémas normatifs de cette autorité permettent dans la situation actuelle de dériver la norme $Ob_{\mathtt{a}}^{\mathtt{syldavianLaw}}(\varphi \rightarrow \psi)$, imposant à l'agent `a` de supprimer l'information mise en cause dès lors que le traitement a eu lieu. Nous introduisons enfin une dernière norme $Per_{\mathtt{a}}^{\mathtt{syldavianLaw}}\neg\varphi$, exprimant le fait que la loi syldave autorise explicitement l'agent `a` à ne pas procéder au traitement.

Détection du conflit

L'ensemble de normes Δ sur lequel s'appuie l'agent `a` contient donc les formules suivantes :

$$Ob_{\mathtt{a}}^{\mathtt{self}}\ \varphi \tag{4.98}$$

$$For_{\mathtt{a}}^{\mathtt{self}}\psi \tag{4.99}$$

$$Ob_{\mathtt{a}}^{\mathtt{syldavianLaw}}(\varphi \rightarrow \psi) \tag{4.100}$$

$$Per_{\mathtt{a}}^{\mathtt{syldavianLaw}}\neg\varphi \tag{4.101}$$

On peut vérifier très rapidement qu'aucun couple de formules de cet ensemble ne contient de conflit. Par contre, l'ensemble de ces quatre normes, pris globalement, est conflictuel. En effet, si l'on se place dans le cadre axiomatique de DLP+, on peut déduire la formule $Per_{\mathtt{a}}^{\mathtt{self}}(\neg\varphi \vee \psi)$, soit $\neg Ob_{\mathtt{a}}^{\mathtt{self}}(\varphi \wedge \neg\psi)$, à partir de (4.100) et de l'axiome (DLP+-AC). D'autre part, à partir de (4.98) et (4.99) on peut dériver $Ob_{\mathtt{a}}^{\mathtt{self}}(\varphi \wedge \neg\psi)$[36], ce qui constitue une inconsistance avec la

[35] Lorsque nous traiterons la mise en œuvre de l'algorithme d'arbitrage, nous discuterons des restrictions nécessaires – ou tout du moins utiles – à apporter à la syntaxe des normes afin de les traiter efficacement.

[36] $(\Box\varphi \wedge \Box\psi) \rightarrow \Box(\varphi \wedge \psi)$ est en effet un théorème de toute logique modale normale. La démonstration, laissée au lecteur intéressé, part de la tautologie $\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi))$ et utilise des instances particulières de l'axiome (K) (à savoir, $\Box(\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi))) \rightarrow (\Box\varphi \rightarrow \Box(\psi \rightarrow (\varphi \wedge \psi)))$ et $\Box(\psi \rightarrow (\varphi \wedge \psi)) \rightarrow (\Box\psi \rightarrow \Box(\varphi \wedge \psi))$).

formule précédemment déduite, dont elle est la négation. On peut donc considérer qu'il existe un conflit binaire dans l'ensemble de formules formé de la conséquence de Δ en DLP+, ce qui est une autre manière de dire que Δ contient un conflit d'obligations. On pourra aisément vérifier que l'ensemble conflictuel { (4.98), (4.99), (4.100) } ainsi mis en valeur est minimal, l'inconsistance ne pouvant être mise en évidence si l'on retire l'une des trois formules.

On notera par contre que l'ensemble { (4.98), (4.101) } ne présente pas de conflit d'obligations, de par le rôle particulier que nous avons donné à la notion de permission. Les axiomes de DLP+ n'aboutissent pas à une contradiction, alors que la plupart des autres systèmes de résolution de conflits normatifs considèreraient ce couple comme inconsistant (car $\{Ob\ \varphi, Per\ \neg\varphi\}$ est inconsistant en SDL).

Arbitrage du conflit

Pour arbitrer le conflit afin de construire un ensemble de normes Δ' exempt de conflits d'obligations, on peut considérer deux cas de figure[37] :

- Soit la loi syldave a été définie par l'utilisateur comme plus importante que ses propres préférences (`syldavianLaw` $\lhd_a$ `self`) ;
- Soit l'utilisateur a défini ses préférences comme prévalant sur la loi syldave (`self` $\lhd_a$ `syldavianLaw`).

Dans chacune de ces deux hypothèses, la désactivation d'une seule des trois normes du sous-ensemble conflictuel minimal identifié rendra le nouvel ensemble exempt de conflits.

Dans le premier cas, une des deux normes de l'utilisateur, (4.98) ou (4.99), doit être désactivée. Le pseudo-algorithme d'arbitrage que nous proposons ne permet pas de décider laquelle, c'est donc à l'agent de décider si la suppression des données présente pour lui un risque suffisamment important pour motiver un renoncement au traitement. Dans le cas contraire, qui semble le plus probable, l'agent procèdera au traitement mais supprimera les données, conformément à la loi syldave. Les deux ensembles de normes que l'on peut théoriquement obtenir sont donc les suivants :

$$\Delta'_1 = \left\{ \begin{array}{c} For_{\mathtt{a}}^{\mathtt{self}}\psi \\ Ob_{\mathtt{a}}^{\mathtt{syldavianLaw}}(\varphi \rightarrow \psi) \end{array} \right\} \tag{4.102}$$

$$\Delta'_2 = \left\{ \begin{array}{c} Ob_{\mathtt{a}}^{\mathtt{self}}\ \varphi \\ Ob_{\mathtt{a}}^{\mathtt{syldavianLaw}}(\varphi \rightarrow \psi) \end{array} \right\} \tag{4.103}$$

Dans le second cas, c'est la norme correspondant à la loi syldave qui est ignorée. L'agent procédera au traitement et conservera les données. L'ensemble de normes correspondant est le suivant :

$$\Delta'_3 = \left\{ \begin{array}{c} Ob_{\mathtt{a}}^{\mathtt{self}}\ \varphi \\ For_{\mathtt{a}}^{\mathtt{self}}\psi \end{array} \right\} \tag{4.104}$$

4.4.6 Synthèse

Nous avons posé une définition des conflits d'obligations permettant de caractériser, parmi les incohérences du contexte normatif d'un agent PAw, celles qui l'empêchent de construire un modèle de comportement ne violant aucune des normes. Nous avons donné un moyen formel,

[37] À cause de la non-antisymétrie de $\lhd_a$, on peut également avoir le cas où les deux autorités prévalent mutuellement l'une sur l'autre, elle sont alors sur un même pied d'égalité. Dans ce cas, le pseudo-algorithme proposé ne permet pas de prendre une décision de manière déterministe, n'importe laquelle des trois normes pourra être désactivée.

axiomatique, de détecter ces conflits. Nous avons proposé un algorithme de principe permettant de les arbitrer en utilisant une relation de prévalence entre autorités normatives.

Nous avons également mis en lumière la principale limitation de l'aspect purement théorique de cet arbitrage, à savoir son non-déterminisme, qui survient lorsque plusieurs autorités ont des prévalences normatives équivalentes[38] ou bien lorsqu'un sous-ensemble conflictuel minimal comporte plusieurs normes rattachée à l'autorité normative de prévalence minimale. Le déterminisme de la procédure de décision devra donc être apporté à l'étape de sa mise en œuvre technique, par l'établissement de stratégies de choix adéquates.

4.5 Impact sur les croyances

En présentant la logique DLP comme outil de raisonnement sur les normes en matière de protection des données personnelles, notre objectif est de fournir un outil générique, pouvant servir à la conception d'un composant indépendant d'un modèle d'agent ou être intégré dans un modèle existant. Nous proposons ici une brève réflexion sur le lien nécessaire entre une couche de raisonnement DLP et une couche de raisonnement doxastique. Les idées présentées ici ne sont que des pistes pour relier un moteur DLP à un modèle préexistant. La seule intégration complète que nous proposons sera présentées dans le chapitre 6.

4.5.1 Croyances sur des données personnelles

Les modèles classiques de la gestion des croyances par un agent cognitif ne prennent pas en compte les spécificités de la protection des données personnelles. Toutes les croyances de l'agent sont généralement placées sur le même plan, accessibles de la même manière aux outils de raisonnement et de planification de l'agent. L'objectif qui nous intéresse ici est donc de distinguer les données personnelles des autres croyances, afin de permettre ou même d'imposer qu'elles soient traitées conformément aux contraintes exprimées par la base de normes DLP. Cette problématique se pose de manière différente pour les modèles cognitifs fondés sur la logique modale (ou toute autre logique permettant une compatibilité avec la logique modale) et pour les modèles procéduraux.

4.5.1.1 Croyances représentées en logique modale

Les modèles doxastiques les plus courants utilisent une logique modale normale utilisant des modalités de croyances Bel_a de type $KD45$. Lorsque l'agent utilise une formule de type $Bel_a\varphi$ dans son raisonnement, il n'a a priori aucun moyen de savoir si φ contient des données personnelles ou non. Si l'on prévoit d'intégrer complètement le langage DLP au modèle doxastique d'un agent, il pourraît paraître naturel d'introduire un opérateur $Priv$, du type "$Priv$("BELIEF", "ID", "DATAID")" permettant de mettre en relation une croyance avec une donnée personnelle identifiée comme telle en DLP (en utilisant au besoin des types de traitements destinés uniquement à la conservation de données). Cette solution peut ainsi permettre aux raisonnements déjà prévus pour raisonner sur les données personnelles de prendre en compte les informations normatives apportées par les formules DLP disponibles. Néanmoins, elle n'empêche en rien d'autres types de raisonnements, ne se préoccupant pas du caractère privé de certaines informations, de raisonner uniquement sur la modalité doxastique en ignorant les éventuelles restrictions représentées en DLP.

[38] Ce cas de non-déterminisme pourrait évidemment être évité si l'on acceptait l'hypothèse, encore plus restrictive, faisant de $\lhd_a$ une relation d'ordre et non plus un préordre.

Pour contourner cet inconvénient, il pourrait être plus judicieux de remplacer, dans le cas d'une donnée personnelle, la modalité Bel_a par une modalité spécialisée $BelPriv_a$, en la dotant de règles d'inférence subordonnant son utilisation à une mise en relation avec les normes DLP associées à la croyance en question (via un opérateur du type de celui précédemment suggéré, par exemple). Ainsi, un modèle de raisonnement utilisant uniquement sur la modalité Bel_a ne pourrait accéder à des données identifiées comme personnelles, et un modèle de raisonnement exploitant des croyances représentées par $BelPriv_a$ serait formellement contraint par les normes DLP connues de l'agent. Étant donné que l'utilisation faite par les agents des croyances, représentées sous forme purement logique, est très dépendante de la spécialisation de l'agent, de son activité « métier », il est cependant délicat de proposer directement un ensemble de règles génériques applicables à tout type de modèle d'agent fondé sur la logique modale.

4.5.1.2 Croyances des modèles procéduraux

Si l'on considère des modèles mentaux procéduraux, comme ceux de Jason [BH06] ou de JACK [AOS], l'adaptation semble plus simple. À cause de la différence de formalisme évidente, si l'on souhaite mettre en œuvre un moteur DLP purement logique, il devra être un composant annexe au modèle mental principal de l'agent, et l'ensemble des formules DLP déduites pour un contexte normatif donné devront être importées en tant que croyances dans le modèle natif. Les normes y seraient donc représentées par des croyances, au même titre que les observations de l'agent sur les faits du monde ou que les données personnelles. Cependant, dans un modèle comme dans l'autre, il est aisé de différencier les trois concepts. Jason permet en effet d'annoter chaque croyance et d'utiliser les annotation dans les phase d'unification de la mise en œuvre des plans. Il semble donc possible d'indiquer aux plans qu'ils travaillent avec des croyances sur des données personnelles et qu'ils doivent donc prendre en compte un certain nombre de contraintes fournies par DLP. Dans le cas de JACK, normes, croyances classiques et données personnelles peuvent être stockées dans des *beliefsets* de types différents, dotés de méthodes de rappel (*callback*) permettant de mettre en œuvre des procédures de contrôle d'accès. Encore une fois, il paraît donc aisé de vérifier que le plan qui tente d'accéder à une base de croyances est bien connu comme respectant les normes qui pourraient y être associées[39].

4.5.2 Gestion normative des croyances sensibles

En considérant à la fois les croyances normatives de l'agent issues du raisonnement DLP et ses croyances portant sur des données personnelles, il est possible de construire des procédures spécifiques mettant en œuvre des traitements sur les données personnelles tout en respectant le contexte normatif de l'agent.

4.5.2.1 Utilisation de concepts déontiques préexistants dans le modèle

Dans certains modèles d'agents, une notion préexistante peut être utilisée pour représenter l'obligation déontique des normes DLP.

Dans un modèle de type PLEIAD, par exemple [AEG+06], il peut être justifié d'utiliser la modalité de choix $Choice_a$, exprimant la désirabilité d'une formule pour l'agent. En effet, cette modalité, inspirée du modèle des intentions de Cohen et Levesque [CL90], va guider le

[39] Le mécanisme des méthodes de rappel de JACK semble plus puissant à ce niveau que les annotation de Jason, qui pourraient être ignorées par un plan incompatible avec DLP. On se retrouverait alors dans un cas similaire à celui évoqué pour les modèles logiques avec la seule utilisation d'un opérateur de correspondance entre croyances et formules DLP.

comportement de l'agent. Dans cet exemple, des normes DLP de la forme $Ob_a^\nu \varphi$ peuvent être liées au modèle de base à l'aide de la règle d'inférence basique suivante :

$$\frac{Ob_a^\nu \varphi, \neg Bel_a \ \neg\varphi}{Choice_a \ \varphi} \tag{4.105}$$

Ici il est nécessaire, pour que l'obligation soit prise en compte, que l'agent croie qu'elle n'est pas déjà violée. En effet, les agents PLEIAD ne peuvent faire que des choix réalistes. Ainsi, s'il ne pense pas l'objet de la norme impossible, l'agent préférera que la norme soit respectée et en dérivera les intentions correspondantes. Si la norme est d'ores et déjà violée, un agent PLEIAD ne dérivera aucune modalité $Choice_a$, car il ne croit pas possible de respecter la norme. La prise de conscience par l'agent de la violation d'une norme nécessiterait alors l'introduction d'un nouvel opérateur *ad hoc*. Le modèle doit cependant être modifié pour pouvoir raisonner sur le langage $\mathcal{L}_{DLP}$, prenant ainsi en compte les actions spécifiques au traitement des données personnelles. La règle d'inférence (4.105) n'est évidemment fournie qu'à titre d'exemple simpliste, le modèle PLEIAD manquant de l'expressivité temporelle nécessaire à l'expression de tous les concepts de DLP. Si l'on suppose un modèle similaire, mais disposant en outre de la modalité $\mathcal{U}$, il suffit de faire correspondre directement les modalités temporelles de DLP à celles du modèle cognitif. Comme nous sommes capables d'exprimer obligations avec échéances et interdictions maintenues sans étendre les modalités d'origine, il n'est alors nul besoin d'introduire de règle spécifique pour les traduire. Nous ne proposons pas de règles concernant les normes de nature permissive, étant donné que nous ne leur donnons aucun sens particulier dans le contexte philosophique de l'agent PAw.

Dans un modèle disposant d'un moteur de raisonnement sur les obligations, comme l'architecture BOID [BDH+01], il suffirait naïvement de faire correspondre la notion d'obligation DLP avec celle d'obligation BOID. Cela nécessiterait cependant une réduction très sévère du modèle DLP, la logique BOID ne raisonnant pas directement sur le temps. L'architecture BOID est conçue pour résoudre les conflits pouvant survenir entre croyances, obligations, intentions et désirs. Néanmoins, l'utilisation d'un module DLP implique que les obligations sur lesquelles BOID raisonne ont déjà été arbitrées du point de vue des conflits d'obligations DLP. En conséquence, une petite partie des conflits qui auraient été traités par BOID (appelés *conflits internes entre obligations*, ou plus simplement *O conflicts*) sont confiés à DLP. Le moteur BOID conserve par contre la responsabilité de l'arbitrage dans les quatorze autres types de conflits possibles.

4.5.2.2 Mise en œuvre procédurale

Nous avons déjà évoqué le type de pratiques pouvant être mis en œuvre au sein d'un modèle d'agent procédural pour s'assurer que les croyances concernant des données personnelles ne soient traitées que par des fonctions, procédures ou méthodes garantissant le respect des normes issues du moteur DLP. Pour résumer, les principes de la protection spécifiques des données à caractères personnel par un agent PAw tel que nous le concevons doivent être les suivants :

- L'ensemble de formules fournies par la couche DLP est identifié de manière spécifique comme un ensemble de croyances « normatives » ;
- Les croyances portant sur des données personnelles sont isolées et étiquetées comme telles, de manière à pouvoir les relier aux formules DLP les concernant ;
- Les méthodes de rappel attachées aux croyances « personnelles » s'assurent que ces dernières sont utilisées en conformité avec les croyances normatives applicables.

Ce type de méthodes peut également être mis en œuvre dans des agents utilisant délibérément la logique du premier ordre plutôt qu'une réduction d'un modèle en logique modale.

Nous détaillerons la mise en œuvre pratique de ce contrôle d'accès et d'utilisation dans le chapitre 6.

4.6 Positionnement

Nous avons présenté le modèle logique de la couche normative de l'agent PAw, sur la base du langage DLP. Nous avons délibérément fondé ce langage sur un produit de la Logique Déontique Standard et de la Logique Temporelle Linéaire, montrant ainsi que malgré une expressivité moindre que celle des logiques de type STIT (permettant de modéliser les notions de responsabilité et de causalité), des logiques fondées sur des structures temporelles arborescentes ou encore des logiques déontiques défaisables (fournissant nativement des outils d'arbitrage de conflits), ces langages fondamentaux permettent de travailler sur des notions suffisamment évoluées pour décrire des concepts complexes et précis, dans notre cas liés à la protection des données personnelles.

Nous avons montré qu'il était possible, sans extension du langage, de concevoir des opérateurs de normes datées plus adaptés que les propositions déjà existantes, permettant ainsi d'exprimer de manière satisfaisante les notions d'obligation avec échéance et d'interdiction maintenue. Nous avons montré la possible interaction du langage DLP avec des langages de politiques comme P3P et illustré comment des réglementations portant sur les six axes de la protection des données personnelles pouvaient être exprimées dans cette logique.

Nous avons proposé une définition des conflits d'obligations entre autorités normatives distinctes qui se distingue des travaux existants sur la gestion des conflits normatifs de par le rôle très restreint qu'elle confère à la notion de permission, qui n'est pas pour l'agent PAw un droit accordé mais bien une absence d'interdiction. Par le fait, il nous semble avoir entr'ouvert un débat sur les différents rôles que peuvent jouer les notions déontiques, suivant la manière dont elles sont considérées par les agents ou le système normatif. Nous avons mis en avant un modèle théorique d'arbitrage des conflits d'obligations qui utilise les méthodes courantes de fusion de normes avec préférences pour la recherche d'un ensemble non conflictuel maximal, tout en prenant appui sur l'inférence modale pour traiter les conflits d'arité quelconque, et non pas seulement les conflits binaires comme c'est couramment le cas.

Néanmoins, l'ensemble des outils formels que nous avons mis en place ici sont internes à l'agent PAw. Ils ne peuvent donc que lui servir à respecter lui-même le contexte normatif dans lequel il évolue. Si l'agent PAw doit porter la responsabilité des données personnelles qui lui sont confiées, il lui faut s'assurer que les normes dont il est maintenant conscient seront respectées dans l'ensemble de l'application distribuée, et donc même après que ces données aient été transmises à un agent tiers. Le respect des normes DLP, leur intérêt réel pour l'utilisateur, repose donc maintenant sur la capacité de l'agent PAw à assurer la *protection étendue* des données.

Chapitre 5

Assurer la protection étendue au sein d'une transaction entre agents

Notations

Dans ce chapitre, nous serons amenés à traiter de sécurité informatique au sens plus classique du terme (par comparaison avec la protection des données personnelles). Nous supposerons le lecteur familier avec les notions de hachage cryptographique (chiffrement à sens unique produisant un *condensat*), de chiffrement et déchiffrement (symétrique à l'aide d'une clé partagée ou asymétrique à l'aide d'une clé publique et d'une clé privée) et de signature cryptographique (à l'aide d'un couple de clés asymétriques). Nous utiliserons les conventions suivantes pour la notation des primitives cryptographiques (toute abstraction faite des algorithmes utilisés) :

- $h(C)$ est le hachage cryptographique du nombre C ;
- $\{C\}_K$ est le chiffrement (ou le déchiffrement) symétrique du nombre C à l'aide de la clé K ;
- $\{C\}_{Kpub}$ est le chiffrement asymétrique du nombre C (ou la vérification de la signature C) à l'aide de la partie publique du couple de clés K ;
- $\{C\}_{Kpriv}$ est le déchiffrement asymétrique du nombre C (ou la signature du nombre C) à l'aide de la partie privée du couple de clés K.

5.1 Exigences de la protection étendue

La protection étendue des données, telle que nous l'avons définie, nécessite de pouvoir garantir qu'une politique donnée s'applique bien aux informations sensibles, qu'elles soient gérées par l'agent personnel de l'utilisateur ou bien par un agent distant. À chacun des six axes de la protection des données personnelles correspondent des exigences spécifiques. Si l'on veut les satisfaire pleinement, il est nécessaire de caractériser la confiance que l'on peut avoir dans l'exécution d'un programme distant, dans le système d'exploitation distant, et dans le matériel de la plate-forme distante.

5.1.1 La protection étendue rapportée aux six axes réglementaires

Les six axes réglementaires que nous avons définis dans la section 1.3.1 se sont montrés adaptés pour la classification des normes en la matière (et notamment pour la conception du langage DLP). Cependant, nous allons voir que les frontières entre les six axes s'estompent lorsque l'on traite des méthodes de protection étendue.

Nous supposons ici disposer de telles méthodes, assurant de manière abstraite la protection étendue des données personnelles telle que nous l'avons définie. Nous reprenons les six axes, et pour chacun nous recherchons le type de garanties que doivent fournir à l'utilisateur les moyens techniques à mettre en place.

5.1.1.1 Information

Comme nous l'avons déjà mentionné, l'information de l'utilisateur est un concept essentiellement local. Dans une interaction, si l'agent PAw est du côté « utilisateur », il lui est facile de déterminer si, suivant ses critères (c'est-à-dire les normes DLP de son contexte), il a été suffisamment informé ou pas sur un traitement en particulier. S'il est du côté « service », il sait également si son comportement en matière d'information a été conforme à ses normes ou pas. Dans les deux cas, l'agent PAw peut observer l'interaction et donc raisonner dessus.

Il ne semble donc pas y avoir de nécessité majeure à prendre en compte le concept d'information de l'utilisateur dans la protection étendue de l'information, si l'on s'en tient aux informations que nous avons considérées jusqu'à présent[1]. Cependant, il pourraît être intéressant que les méthodes de protection étendue mises en œuvre permettent de s'assurer qu'un traitement ne puissent s'opérer sur des données personnelles sans que son propriétaire en soit averti, si des normes DLP l'interdisent. En effet, dans ce cas, l'utilisateur ne peut plus vérifier le respect des normes. Les méthodes qui seront jugées utilisables pour assurer les propriétés distantes de l'axe « justification » pourront sans doute être appliquées à ce cas d'espèce, qui se résoud à contrôler l'utilisation des données dans un cadre non prévu.

5.1.1.2 Consentement

De la même manière que pour l'information, le consentement est un concept vérifiable localement. L'agent PAw sait si on lui a demandé ou non son consentement pour un traitement (respectivement, s'il a demandé le consentement d'un autre agent) et s'il l'a accordé (resp. reçu), le cas échéant.

De la même manière que pour l'axe d'information, il peut tout de même être intéressant que les méthodes de protection étendue garantissent qu'un traitement ne puisse avoir lieu que si les normes en matière de consentement du propriétaire des données soient respectées.

5.1.1.3 Modification

Comme nous l'avons vu, l'axe de modification comporte une partie locale et une partie distante. La protection étendue relative aux propriétés distantes de l'axe de modification consiste à s'assurer du respect, par l'agent responsable des données, d'éventuelles requêtes de modification ou de suppression (si l'on considère que des normes DLP « par défaut » imposent le respect de ces requêtes, et de manière général un comportement sincère des agents). Les méthodes mises en œuvre doivent permettre de garantir la mise à jour ou la suppression distante de données, de manière à empêcher l'existence de messages de confirmation mensongers dans le système.

[1] Il est éventuellement possible d'intégrer un autre type d'information dans la protection étendue, par exemple en prévoyant d'informer l'utilisateur à chaque fois qu'une de ses informations est effectivement utilisée. Le protocole proposé par Casassa Mont *et al.* [CPB03], que nous serons amenés à étudier en détail, permet ce type d'interaction planifiée.

5.1.1.4 Justification

L'axe que nous avons nommé « justification » est, lui, essentiellement non local. Les normes qui s'y rapportent spécifient le type d'information qu'il est autorisé ou non de collecter ou d'utiliser pour un traitement d'un type donné. Les propriétés définies par ces normes sont donc observable sur la plate-forme du responsable du traitement, et non pas sur la plate-forme du propriétaire des données. La justification est donc un exemple flagrant de la nécessité de disposer de méthodes de protection étendue. De telles méthodes doivent permettre à l'utilisateur d'obtenir une garantie sur la correspondance (sur la plate-forme distante) entre les données utilisées et la nature du traitement. Il doit être possible de fournir à l'utilisateur une garantie du type : « sur telle plate-forme, telle donnée est utilisée par tel traitement qui effectue tel type d'opération, mais pas tel autre ».

Considérons par exemple que le contexte normatif autorise un agent, au service d'une collectivité locale, à utiliser une information de type « régime alimentaire » pour les traitements se rapportant à la restauration scolaire, mais l'interdisant dans le cas du calcul de la taxe d'habitation (type de norme tout à fait conforme au principe général de finalité). Dans ce cas d'espèce, l'agent PAw d'un citoyen doit pouvoir obtenir des garanties fermes certifiant que l'agent de service, bien que disposant de l'information en question, ne peut pas l'utiliser dans le cadre de ses applications fiscales.

5.1.1.5 Conservation

De la même manière que pour l'axe précédent, l'axe de conservation des données concerne des propriétés distantes, nécessitant des méthodes de protection étendue. La conservation ou la destruction de données par l'agent de service n'est pas directement observable par l'agent utilisateur, ce dernier doit donc obtenir (via les méthodes de protection étendue à mettre en place) des garanties sur le respect des normes en la matière. Ainsi, si le contexte normatif dit que la durée de vie autorisée d'une information est écoulée (*i.e.* qu'un agent ou qu'un ensemble d'agent ont l'obligation de l'oublier), l'agent PAw utilisateur doit pouvoir être « convaincu » (la nature et la force de cette conviction restant encore à déterminer) que la donnée a été détruite par tel agent l'ayant en sa garde, ou bien par tous les agents l'ayant en leur garde, ou encore qu'elle a été rendue inutilisable par un quelconque procédé technique.

5.1.1.6 Transmission

Le dernier axe, traitant de la transmission d'une donnée déjà collectée à un agent tiers, concerne des propriétés distantes très proches de celles de l'axe de justification, comme nous l'avons déjà mentionné. Les méthodes de protection étendue s'y rapportant poursuivent donc des objectifs similaires : l'agent utilisateur doit pouvoir obtenir des garanties fortes sur la transmission ou la non-transmission des données par les agents auxquels elles ont été confiées. En fonction du contexte normatif, les méthodes techniques de protection étendues devront certifier une absence de transmission ou une transmission s'effectuant dans des conditions conformes (fonction de la nature des données, des traitements envisagés, des agents destinataires...).

5.1.2 Les différents niveaux de la protection étendue

Maintenant que nous avons une idée du type de garanties que doivent pouvoir fournir les mesures techniques de protection étendue des données personnelles, nous pouvons nous intéresser aux différents niveaux de sécurité envisageables. Le niveau de sécurité de l'application distribuée

en matière de protection des données personnelles se ramène ici à un problème de confiance. En effet, l'agent utilisateur doit être « convaincu » par les garanties fournies, et cette conviction s'appuie sur la confiance qu'il a dans une certaine entité.

Il peut être intéressant ici de s'autoriser un parallèle avec le problème de l'authentification. Lorsqu'un serveur web sécurisé s'authentifie auprès d'un utilisateur, ce dernier est convaincu de l'identité du serveur (garantie par un certificat cryptographique) parce qu'il a confiance dans la fiabilité du protocole d'authentification utilisé d'une part, et dans l'autorité de certification (tiers de confiance) qui garantit le certificat d'autre part. Cette confiance semble judicieusement placée si le protocole est considéré comme fiable, car le tiers de confiance n'a pas d'intérêt a priori à tromper les utilisateurs sur les certificats qu'il garantit[2].

C'est ce même type de relation de confiance que doivent établir les moyens techniques de protection étendue des données personnelles, et cette confiance peut se situer à différents niveaux. Ces niveaux correspondront à des garanties plus ou moins fortes, la confiance de l'agent utilisateur y étant placée dans des entités plus ou moins fiables.

5.1.2.1 Confiance dans l'agent distant

Le premier niveau consiste pour l'agent PAw à mettre sa confiance dans les agents[3] avec lesquels il interagit et à qui il confie des données. On se repose ici sur la notion cognitive de confiance, telle qu'étudiée dans le cadre des systèmes multi-agents par Cristiano Castelfranchi [CF98, Cas04] ou Robert Demolombe [Dem04], par exemple. Dans ces modèles, la confiance est une relation binaire entre agents, asymétrique, graduée, dynamique et contextuelle. L'agent PAw aura originellement envers un agent tiers un niveau de confiance par défaut, qui sera modifié en regard de l'utilité mesurée des interactions menées avec cet agent : si l'interaction est jugée profitable la confiance augmentera, tandis que si l'agent PAw juge que l'agent tiers a failli à ses obligations (ou qu'il a causé une mesure négative de l'utilité de l'interaction, quelle qu'en soit la définition choisie), la confiance chutera. La confiance est un système de régulation par rétroaction en ce sens que c'est sur la base de la confiance qu'il a dans l'agent tiers que l'agent PAw décidera d'engager une nouvelle interaction avec ce dernier.

À ce niveau, la confiance de l'agent PAw réside donc dans l'agent avec lequel il communique. Ce choix est extrêmement dangereux dans le cas de la protection des données personnelles, et ce pour deux raisons principalement :

- Premièrement, comme nous l'avons déjà vu[4], les violations des normes en matière de vie privée sont le plus souvent « silencieuses » , c'est-à-dire qu'elles ne pourront être observées par l'agent PAw. Il est donc particulièrement mal venu de penser fonder la variation d'une relation de confiance sur la mesure de telles violations.
- Deuxièmement, dans bien des cas l'agent distant est précisément celui qui aurait intérêt à violer les normes en matière de protection des données personnelles. Ces données représentant une ressource de valeur, les agents peuvent naturellement considérer comme

[2] L'entité la plus petite dans laquelle il est nécessaire d'avoir confiance est appelée *root of trust* dans la littérature, car c'est d'elle que découle toute confiance qui peut être accordée au système.

[3] Nous considérons ici que les autres entités logicielles impliquées dans l'application répartie sont des agents. En réalité, dans le cadre de ce chapitre, il pourrait très bien s'agir de programmes informatiques ne répondant pas à la définition d'un agent. L'agent PAw n'a a priori pas connaissance de la structure interne de ses interlocuteurs, ni a fortiori de leurs capacités d'autonomie, de raisonnement, de perception de leur environnement... Le seul prérequis de ces entités distante est leur capacité à respecter un certain nombre de protocoles de communication et d'interaction.

[4] Voir la section 2.2.5 sur la subjectivité des normes et l'observabilité des violations, ou encore la réflexion menée dans une précédente communication [CPBD07].

stratégique de les conserver, de les revendre ou de les réutiliser, en dépit d'éventuelles normes contrôlant ces comportements.

On peut donc conclure que ce niveau de sécurité n'est pas satisfaisant, la confiance de l'agent étant placée dans des entités de l'application qui auraient intérêt à trahir cette confiance, et qui de plus disposeraient les moyens de le faire en toute impunité.

Amélioration de la confiance par l'identification d'un composant logiciel particulier

On peut imaginer un sous-niveau de sécurité, pour l'agent PAw ne se repose plus sur une confiance aveugle dans les autres agents pour évaluer les garanties qui lui sont proposées. On peut maintenant considérer que l'agent PAw exige de l'agent distant qu'il identifie de manière précise le composant logiciel qui va traiter les données, afin de pouvoir déterminer si ce composant respecte ou non les normes applicables. Ce cas de figure peut se présenter lorsque les différents agents impliqués dans l'interaction partagent des composants logiciels d'une même application répartie, interagissant entre eux à l'aide d'un protocole de haut niveau. L'application dans son ensemble est alors supposée connue de l'agent PAw, qui a confiance dans le fait qu'elle respecte effectivement un certain ensemble de normes ou de politiques. On peut également imaginer que l'agent distant affirme utiliser un composant logiciel disponible publiquement et dont le fonctionnement est bien connu.

Dans ces cas de figure, l'agent PAw réclamera aux agents une « signature » du composant logiciel utilisé, par exemple sous la forme d'un hachage cryptographique de son code source ou d'un certificat visé par un tiers de confiance.

Le niveau de sécurité est ici déterminé par le fait que l'agent PAw fonde sa confiance d'une part sur sa connaissance du composant logiciel utilisé (sa conviction a priori que celui-ci ne violera pas les normes), et d'autre part sur la méthode utilisée pour son identification. C'est dans ce dernier point que repose la faille principale de ce niveau de sécurité : l'identification du composant utilisé sur la plate-forme distante revient toujours à l'agent tiers, que nous avons pointé du doigt comme n'étant pas forcément digne de confiance. Le niveau de sécurité n'a donc que peu augmenté par rapport à la solution précédente, car il reste fondamentalement lié à la confiance de l'agent Paw dans l'agent distant, comme illustré dans la figure 5.1. Pour l'améliorer, il nous faut rechercher une méthode similaire de vérification du logiciel distant, mais indépendante des éventuels choix stratégiques de l'agent tiers.

5.1.2.2 Confiance dans le système d'exploitation distant

Une solution envisageable à ce problème est de demander au système d'exploitation distant de garantir la nature des logiciels exécutés sur la plate-forme. Le système peut ainsi fournir une signature de l'agent tiers lui-même (sans pour autant avoir à dévoiler son code) ainsi qu'une signature ou un certificat concernant le composant logiciel chargé d'exploiter les données personnelles. De telles signatures peuvent ensuite être comparées à des bases de données publiques assurant que les logiciels produisant telle ou telle signature respectent ou non telle ou telle propriété lors de leur exécution. Il est bien entendu nécessaire que le système d'exploitation distant soit doté de la capacité à produire de telles informations, qui peuvent suivant les cas concerner tout ou partie des logiciels exécutés[5]. Le principe de ce deuxième niveau de confiance est illustré par la figure 5.2.

[5] Il est généralement intéressant de disposer d'une signature complète des processus actifs sur le système, afin de s'assurer de l'absence de logiciels connus pour exploiter des failles de sécurité ou écouter les échanges de messages internes à la plate-forme, ce qui mettrait en danger la confidentialité des informations traitées et donc le respect des normes de protection des données personnelles.

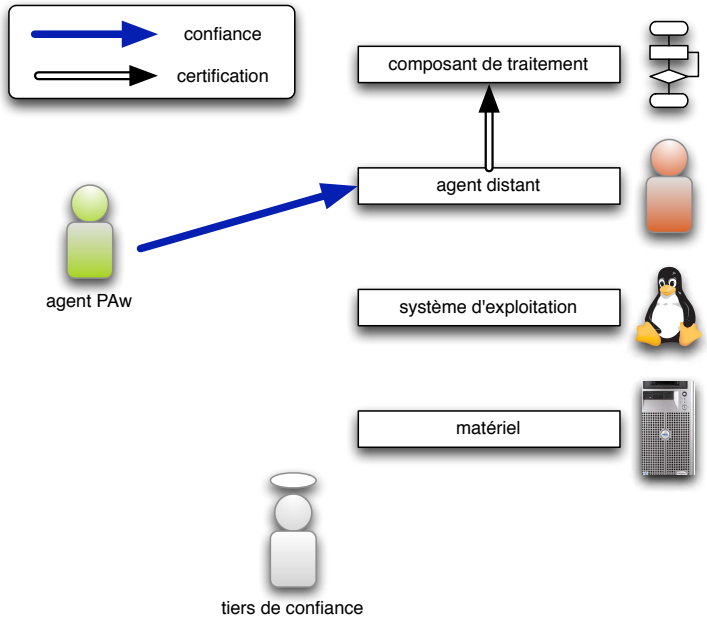

FIG. 5.1 – Premier niveau de confiance de la protection étendue

Dans ce cas de figure, l'agent PAw fonde sa conviction en la valeur des garanties fournies sur la confiance qu'il a dans le système d'exploitation distant. Le niveau de sécurité augmente ici sensiblement, car pour contrefaire ce type de garantie, le propriétaire de l'agent distant doit modifier le système d'exploitation lui-même, afin de le forcer à émettre de fausses déclarations sur la nature des logiciels exécutés. Si l'agent tiers est un autre agent utilisateur, dont le propriétaire est un particulier, il est fort peu vraisemblable que ce dernier ait pu mettre en œuvre les moyens nécessaires à une telle manipulation. En revanche, dans le cas où l'agent tiers est sous le contrôle d'une grande entreprise ou d'une administration publique, il est raisonnable de supposer que ces entités ont les moyens de développer des systèmes d'exploitation personnalisés utilisant une version modifiée (possiblement malveillante ou défaillante) du composant chargé de l'identification des logiciels exécutés.

Ainsi, même si le niveau de sécurité augmente sensiblement à cette étape, il peut rester nécessaire (dans certains cas d'utilisation relativement courants) de rechercher une sécurité supplémentaire, ne nécessitant pas de se reposer sur une confiance dans le système d'exploitation distant.

5.1.2.3 Confiance dans la plate-forme matérielle distante

Il est possible d'augmenter encore la sécurité en demandant au matériel lui-même de garantir le ou les systèmes d'exploitation lancés[6], ainsi que les logiciels exécutés. On peut en effet imaginer que les composants matériels responsables du chargement du BIOS et du *bootstrap* du système d'exploitation s'assurent que cette phase du lancement de la machine se passe de la manière

[6] En effet, l'utilisation d'un système de virtualisation sur la plate-forme distante est également un risque potentiel pour la protection des données personnelles. Dans cette situation, même si le système d'exploitation et les logiciels qu'il contrôle sont bienveillants, le logiciel de virtualisation peut éventuellement observer son fonctionnement interne et en soutirer des informations pour les utiliser ensuite en violation du contexte normatif.

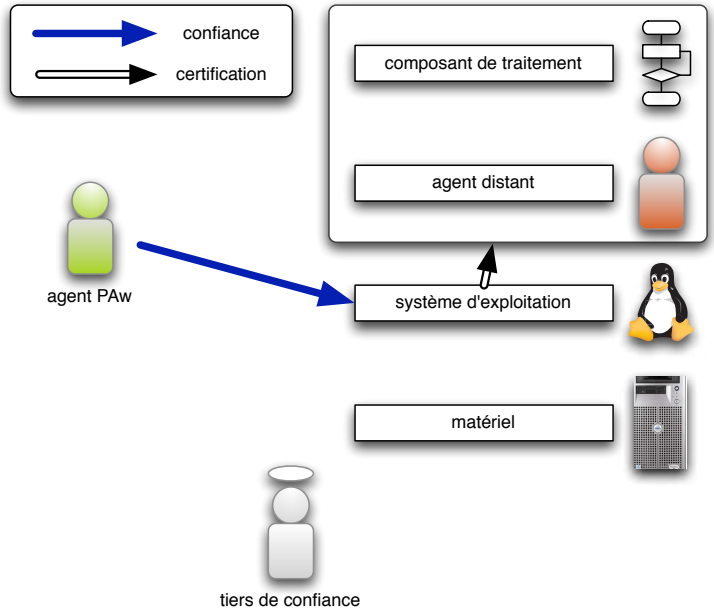

FIG. 5.2 – Deuxième niveau de confiance de la protection étendue

prévue et que le code du système d'exploitation n'a pas été altéré. Un composant matériel dédié pourrait donc certifier l'identité du système d'exploitation, qui pourrait en retour certifier les logiciels en cours d'exécution, et ainsi de suite. Ce principe est illustré par la figure 5.3.

Le risque de faillibilité d'une procédure de ce type est encore réduit par rapport au niveau précédent. Ici, l'agent PAw doit faire confiance au composant matériel spécialisé de la plate-forme distante, qui garantit la nature de tous les composants logiciels (il n'y a donc pas besoin de faire explicitement confiance à ces derniers). Une entité souhaitant circonvenir ce protocole et fournir des garanties contrefaites devrait ainsi utiliser un composant matériel développé spécifiquement pour générer des certificats trompeurs. Les coûts engendrés sont donc encore supérieurs à ceux attachés à la modification du système d'exploitation, et la liste des agents susceptibles de mettre en œuvre ces moyens de contournement se réduit donc d'autant.

Cependant, on peut considérer que les données manipulées par l'agent PAw sont particulièrement sensibles et que les agents à qui elles sont confiées disposent de moyens particulièrement poussés. On peut donc envisager un dernier niveau de sécurité, pour lequel l'agent PAw n'aurait pas besoin d'avoir confiance dans le matériel distant.

5.1.2.4 Confiance dans une autorité indépendante

À tous les niveaux précédents, il était nécessaire pour l'agent PAw de faire confiance à un composant logiciel ou matériel directement impliqué dans le traitement des données à protéger, et donc potentiellement faillible. Ce dernier niveau de sécurité consiste à déporter complètement la confiance de l'agent PAw sur une autorité de certification, un agent tiers indépendant (et sur la fiabilité des méthodes cryptographiques mises en œuvre).

S'il était possible qu'un tiers de confiance, choisi par l'agent PAw pour limiter les risques de collusion avec d'autres agents peu fiables, certifie que le composant matériel spécialisé de la plate-forme distante est bien conforme à ses spécifications de fonctionnement, alors l'agent PAw

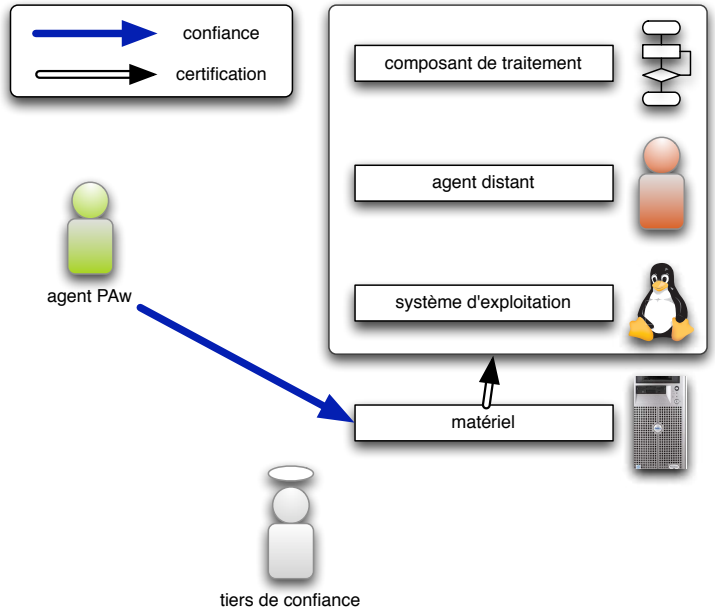

FIG. 5.3 – Troisième niveau de confiance de la protection étendue

pourrait être raisonnablement convaincu du bien-fondé des garanties que ce composant fournit, et ainsi de toute la chaîne de garanties qui en découle sur la nature et le fonctionnement des composants logiciels impliqués dans le traitement distant. Ce principe est illustré par la figure 5.4. Permettre cette certification n'est pas chose aisée, car cela implique que le tiers de confiance ait une connaissance approfondie de la manière dont a été fabriqué le composant matériel spécialisé de chacune des plates-formes pour lesquelles il est susceptible de devoir se prononcer. Cette certification nécessite donc une participation active du fondeur ayant développé le composant.

Ce niveau de confiance particulièrement élevé peut néanmoins théoriquement être atteint. En effet, nous allons voir que les spécifications techniques développées par le *Trusted Computing Group*, ou TCG [Tru][7], initialement développées pour la poursuite d'objectifs tout autres, permettent de mettre en place des architectures fournissant à un agent utilisateur des garanties fortes sur les propriétés distantes de la protection des données personnelles.

5.2 Les technologies fondées sur le *Trusted Computing*

Nous venons de voir que si des garanties fortes doivent être présentées à l'agent PAw concernant la protection étendue de ses données, les solutions techniques de nature purement logicielle sont insuffisantes. La nécessaire implication du niveau matériel (qui permettrait de déporter la confiance de l'agent PAw sur des entités suffisamment fiables, comme un tiers de confiance ou les propriétés connues d'un protocole cryptographique) a déjà été suggérée dans le cadre des propositions du *Trusted Computing Group* [Tru] afin d'assurer d'autres types de propriétés distantes[8].

[7] Le TCG est un consortium regroupant de nombreuses entreprises informatiques, et notamment de fondeurs, anciennement connu sous le nom de TCPA ou *Trusted Computing Platform Alliance*.

[8] Les technologies du TCG ont initialement été développées avec pour objectif principal d'assurer ce que l'on a appelé le « DRM (*Digital Right Management*) matériel », une protection forte des droits d'auteur attachés à des

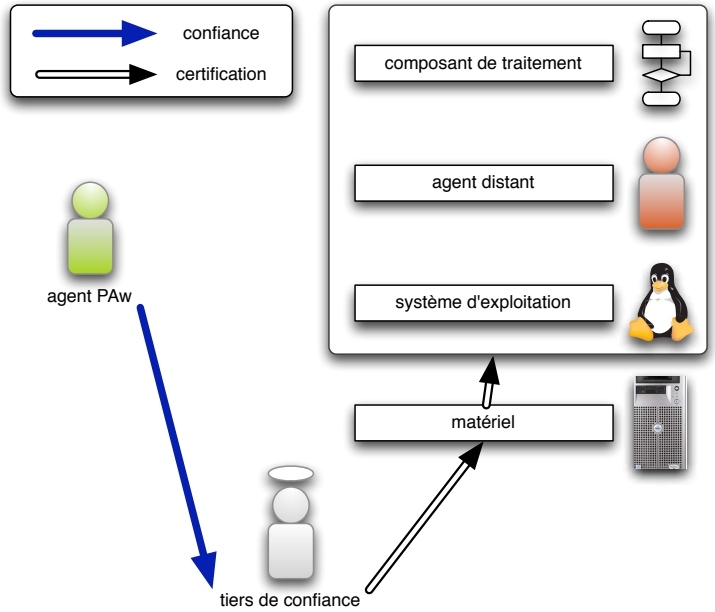

FIG. 5.4 – Quatrième niveau de confiance de la protection étendue

Nous allons brièvement présenter ces technologies de *Trusted Computing* et observer dans quel cadre elles ont déjà été impliquées dans des architectures orientées vers la protection de la vie privée. Il nous faudra caractériser les différents modes d'utilisation de ces technologies et évaluer les risques potentiels (et notamment en matière de vie privée, de manière paradoxale) qu'elles peuvent faire courir à l'utilisateur.

5.2.1 Principes du *Trusted Computing*

Le *Trusted Computing*, ou « informatique de confiance », est un ensemble de concepts qui reposent essentiellement sur le principe des derniers niveaux de confiance que nous avons évoqués. Une « plate-forme de confiance » (qui désigne ici une *Trusted Computing Platform* telle que décrite dans les spécifications du TCG [Tru03a]) est en fait une plate-forme informatique dont la couche matérielle comporte un composant particulier, le *Trusted Platform Module* (ou TPM). Le TPM est doté de capacités cryptographiques particulières qui permettent de mettre en œuvre un certain nombre d'activités de base, utiles à la protection étendue des données. Les architectures d'informatique de confiance peuvent également impliquer des tiers de confiance, des autorités de certification cryptographique dédiées appelés *Trusted Authorities*. Lorsque nous nous référerons à des « tiers de confiance » ou à des « autorités de confiance », nous les supposerons dorénavant compatibles avec les protocoles de l'informatique de confiance.

Les plates-formes de confiance ont été décrites de manière très détaillée par Siani Pearson dans son ouvrage *Trusted Computing Platforms : TCPA technology in context* [Pea02b], qui se réfère toutefois à la première version des spécifications. Nous tâcherons dans notre analyse de prendre en compte les développements importants introduits notamment par la version 1.1 des spécifications générales [Tru03a] et la version 1.2 des spécifications du TPM, parues en 2003

documents (notamment multimédias) mis à disposition sur un réseau.

[Tru03b]. Les propositions de l'informatique de confiance ont été étudiées avec attention par la communauté académique de sécurité informatique, par exemple par Mark Dermot Ryan [Rya06a, Rya06b] ou Ross Anderson [And03], et intégrées à de nombreuses propositions. Nous présentons ici un résumé très synthétique des composants, propriétés et fonctionnalités spécifiques d'une plate-forme de confiance.

5.2.1.1 Le *Trusted Platform Module*

Le *Trusted Platform Module*, ou TPM[9] [Tru03b] est une puce électronique qui peut être montée sur une carte d'extension ou sur la carte mère d'une plate-forme informatique. Le TPM comprend les composants suivants (les éléments logiques seront détaillés par la suite) :

- Un processeur cryptographique disposant d'un générateur de nombres pseudo-aléatoires, d'un moteur de hachage SHA-1 [Nat93], d'un générateur de clés asymétriques RSA [RSA78] et d'un moteur de signature et de chiffrement (symétrique et asymétrique)[10] ;
- Une mémoire persistante permettant le stockage permanent de deux paires de clés RSA ;
- Une mémoire volatile comprenant notamment plusieurs emplacements de stockage de clés ;
- Une interface d'entrée-sortie.

La figure 5.5 représente de manière schématique les éléments fondamentaux du TPM, ainsi que l'icône que nous utiliserons pour le représenter par la suite.

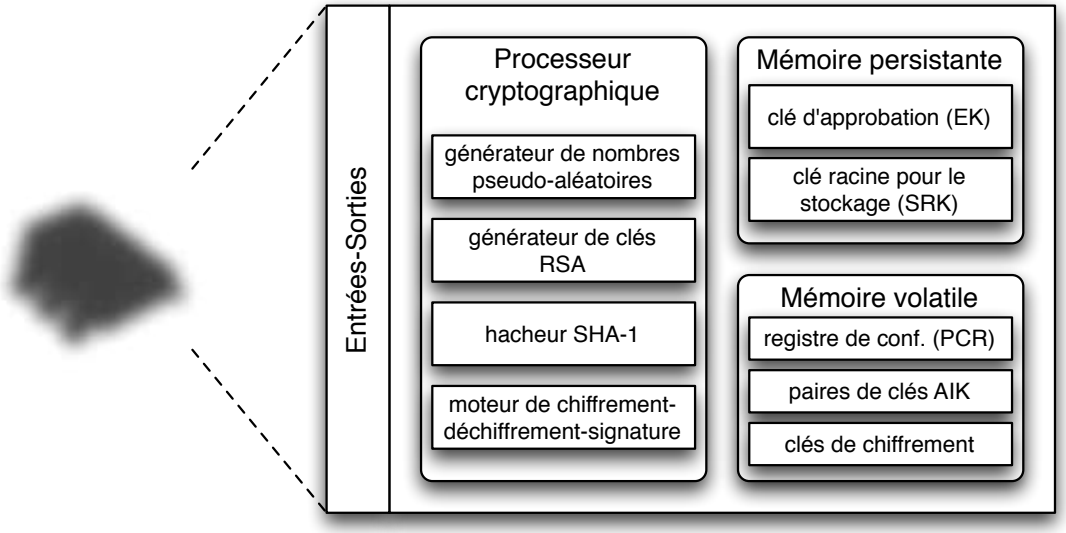

FIG. 5.5 – Schéma des éléments constitutifs d'un *Trusted Platform Module*

Le TPM est conçu de manière à être résistant aux attaques physiques, c'est-à-dire que les informations qu'il renferme (notamment celles de la mémoire persistante) seront détruites en cas de tentative d'intervention physique sur le composant. Les fonctionnalités des plates-formes de confiance reposent sur quelques éléments clé stockés ou créés par le TPM : la **clé d'approbation**,

[9] Le TPM a également été surnommé *Fritz chip* dans ses premières années [And03].

[10] Les algorithmes SHA-1 et RSA sont imposés par les spécifications, mais un TPM peut également proposer d'autres algorithmes de chiffrement.

la **clé racine pour le stockage**, le **registre de configuration de la plate-forme** et les **clés d'identité d'attestation**.

La clé d'approbation

La clé d'approbation (*Endorsement Key* ou *EK*) est une paire de clés RSA (*EKpub*, *EKpriv*) stockée dans la mémoire persistante du TPM. Cette paire de clé est créée en même temps que le TPM. La clé privée est stockée dans le TPM et nulle par ailleurs : elle n'est normalement pas conservée par le fabricant du composant. Avant de commercialiser la puce, le fondeur diffuse la partie publique de la clé. Toute autorité de confiance sera ainsi en mesure d'être convaincue, lors d'un dialogue avec le TPM, de l'authenticité de ce dernier[11].

La clé racine pour le stockage

La clé racine pour le stockage (*Storage Root Key* ou *SRK*) est un autre couple de clés RSA, créé lorsqu'un utilisateur **prend possession** du TPM (opération qui correspond à une primitive spécifique, *take ownership*, du TPM). Le couple est également stocké dans la partie persistante de la mémoire du TPM, et la partie privée du couple n'en sort jamais. Cette clé asymétrique est utilisée pour chiffrer de manière sûre d'autres clés, qui seront stockées à l'extérieur du TPM et utilisées pour le chiffrement sécurisé de données.

Le registre de configuration de la plate-forme

Le registre de configuration de la plate-forme (*Platform Configuration Register* ou PCR) est un condensat cryptographique stocké dans la partie volatile de la mémoire du TPM. Il constitue une signature de l'ensemble des logiciels en cours d'exécution sur la plate-forme[12].

Les clés d'identité d'attestation

Les clés d'identité d'attestation (*Attestation Identity Key* ou *AIK*) sont des paires de clés asymétriques créées par le TPM et stockées dans la partie volatile de sa mémoire. Elles sont utilisées pour authentifier la machine en tant que plate-forme de confiance sans impliquer directement la clé d'approbation. Elles sont à la base de la gestion des identités virtuelles.

5.2.1.2 L'activation du TPM

Une plate-forme peut démarrer et fonctionner avec son TPM activé ou désactivé. Dans le dernier cas, elle ne disposera pas des fonctionnalités d'une plate-forme de confiance. Pour que le TPM s'active, il est nécessaire que la phase de démarrage de la plate-forme, dans laquelle le TPM est impliqué, se passe de manière « acceptable ». Le code du BIOS (qui doit être compatible avec l'informatique de confiance) est vérifié par le TPM avant son chargement, afin de s'assurer qu'il n'a pas été modifié. Lors du chargement du système d'exploitation (qui doit également être

[11] Il suffit pour cela que l'autorité demande au TPM de signer un nombre aléatoire *challenge* avec sa clé privée *EKpub*. L'autorité vérifie que la clé privée correspond bien à la clé publique en vérifiant cette signature : $\{\{challenge\}_{EKpriv}\}_{EKpub} = challenge$.

[12] Cette présentation est une simplification de la réalité. Dans la pratique, plusieurs PCR peuvent être présents dans la mémoire volatile du TPM, représentant les environnements correspondants aux différentes sessions ouvertes sur la machines mais isolées les unes des autres en mémoire.

compatible), le BIOS vérifie l'intégrité du code du noyau. Si l'une de ces vérifications échoue, alors le TPM est désactivé[13].

L'utilisateur de la plate-forme peut également décider de démarrer celle-ci avec le TPM désactivé, renonçant ainsi aux fonctionnalités de l'informatique de confiance[14].

5.2.1.3 L'authentification en tant que *Trusted Computing Platform*

Comme nous l'avons suggéré, il est aisé pour un agent de vérifier qu'une plate-forme distante est bien une plate-forme de confiance, puisque la partie publique des clés d'approbation est diffusée par des autorités de certification. Cependant, l'utilisation de cette clé *EK* auprès de n'importe quel agent permet d'identifier la plate-forme de manière unique, autorisant donc la traçabilité totale de ses activité en cas de collusion des différents agents. Ceci constituant un risque majeur pour la protection du pseudonymat de la plate-forme (et de ses utilisateurs), ce sont d'autres protocoles qui sont utilisés. L'authentification en tant que plate-forme de confiance peut se faire soit par le biais de tiers de confiance, soit via l'utilisation d'un protocole plus complexe.

Implication de tiers de confiance

Dans le scénario impliquant un tiers de confiance, la plate-forme de confiance va générer une paire de clés d'identité d'attestation *AIK*, dont elle va signer la partie publique à l'aide de la partie privée de sa clé d'approbation *EK*. La plate-forme envoie ensuite à un tiers de confiance (utilisant une paire de clés *TC*) la partie publique de la clé *AIK*, la partie publique de sa clé *EK* ainsi que la signature (formule (5.1)). Le tiers de confiance vérifie que la clé *EKpub* est bien la partie publique d'une clé d'approbation valide, puis vérifie la signature (formule (5.2)). Ensuite, la plate-forme devra prouver au tiers de confiance qu'elle connaît la partie secrète de la clé *AIK*, ce qui peut se faire de manière simple par un défi (*challenge*) cryptographique. À l'issue de quoi, le tiers de confiance est bien convaincu que la clé *AIK* a été générée par une plate-forme de confiance authentique. Il va donc signer la partie publique de cette clé à l'aide de sa clé privée et la renvoyer à la plate-forme. Lorsque la plate-forme voudra prouver à un agent qu'elle est bien une plate-forme de confiance, elle lui fournira la partie publique d'une clé d'identité d'attestation ainsi que la signature de cette clé publique par un tiers de confiance (formule (5.3)).

$$AIKpub, EKpub, \{AIKpub\}_{EKpriv} \tag{5.1}$$

$$AIKpub \stackrel{?}{=} \{\{AIKpub\}_{EKpriv}\}_{EKpub} \tag{5.2}$$

$$AIKpub, TCpub, \{AIKpub\}_{TCpriv} \tag{5.3}$$

L'agent pourra ainsi avoir la garantie (certifiée par le tiers de confiance) que les données signées par la clé *AIKpriv* l'ont bien été par une plate-forme de confiance authentique, sans pour autant que la partie publique de la clé d'approbation (*EKpub*) n'ait eu à lui être révélée.

[13] Ceci signifie entre autres que la plate-forme ne pourra pas être identifiée comme une plate-forme de confiance et que les données précédemment chiffrées par le biais du TPM seront inaccessibles.

[14] Dans certaines mises en œuvre de l'informatique de confiance et notamment dans celle de Microsoft, le NGBSC (*New Generation Based Secure Computing*), le système d'exploitation est divisé en deux parties distinctes exécutées en parallèle : un environnement standard et un environnement de confiance. Les mémoires des deux environnements sont isolées l'une de l'autre et le passage de données de l'environnement de confiance à l'environnement standard est impossible.

De cette manière, la plate-forme pourra, à l'occasion de différentes transactions, utiliser plusieurs clés d'identité d'attestation auprès du même agent sans que ce dernier puisse établir une corrélation entre les transactions. C'est de cette manière que l'informatique de confiance peut être utilisée pour la gestion d'identités virtuelles. Ce protocole est en général utilisé dans une version légèrement plus complexe permettant d'attacher la clé d'attestation à un programme particulier sur la plate-forme de confiance (opération d'attestation proprement dite).

Utilisation du protocole DAA

Si le protocole que nous venons de présenter permet de fournir à un agent des garanties fortes sur le fait que la plate-forme distante soit bien une plate-forme de confiance sans toutefois dévoiler son identité, il comporte toutefois des risques pour la protection de l'anonymat (ou plus précisément du pseudonymat) de la plate-forme. Ce protocole est en effet sensible au risque de collusion entre tiers de confiance. Si plusieurs tiers de confiances sont contrôlés par une même entité malveillante, alors l'utilisation de la clé d'approbation *EK* auprès des différents tiers de confiance peut être tracée, et cette clé attachée à l'ensemble des clés d'identité d'attestation *AIK* utilisées par la plate-forme. L'hypothétique entité malveillante est donc en mesure de relier de manière totalement déterministe un certain nombre d'activités de la plate-forme de confiance.

Pour limiter l'impact des collusions, agents et plates-formes peuvent éventuellement s'entendre à chaque interaction pour choisir les tiers de confiance de manière aléatoire. Le risque de traçabilité s'en trouve diminué, mais non annulé[15].

Pour cette raison, le protocole DAA, pour *Direct Anonymous Attestation* [BCC04], a été proposé en 2004 en complément de l'utilisation des tiers de confiance. DAA est un protocole d'authentification de type *zero-knowledge* (également appelé en français « à divulgation nulle de connaissance »), c'est-à-dire que la plate-forme de confiance peut prouver à l'agent distant qu'elle dispose d'une clé d'approbation valide sans toutefois lui donner d'information sur l'identité de cette clé[16]. L'étude de ce type de protocoles dépasse le cadre de nos travaux et nous accepterons cette propriété[17].

Dans son rapport sur l'informatique de confiance en 2004 [Art04], le groupe de travail de la Commission européenne sur la protection des données met en lumière le risque encouru par le recours systématique aux tiers de confiance dans la phase d'authentification. Le rapport recommande, pour la protection de la vie privée des utilisateurs, que le protocole DAA soit la méthode d'authentification par défaut des plates-formes de confiance, ce qui n'était pas encore le cas dans la version courante des spécification du TCG.

5.2.1.4 La certification de la pile d'exécution

Nous avons vu que le TPM conservait en mémoire, au cours de son exécution, le registre de configuration de la plate-forme (*PCR*). Le *PCR* est un condensat résumant l'ensemble des

[15] En effet, le choix des tiers de confiance peut être en partie manipulé au détriment de la plate-forme de confiance si l'agent distant connaît et contrôle les collusions.

[16] Plus précisément, le protocole se base sur une séquence de défis (*challenges*) mathématiques proposés par l'agent distant. Pour chacun d'eux, il y a une certaine probabilité pour que la plate-forme réponde correctement au défi sans toutefois disposer d'une clé d'approbation valide. Cependant, en augmentant le nombre de défis, cette probabilité chute très rapidement. L'agent distant peut donc se convaincre que la plate-forme est bien une plate-forme de confiance avec une probabilité aussi élevée que nécessaire.

[17] En réalité, des failles (susceptible de mettre en péril le pseudonymat de la plate-forme de confiance) ont été mises en évidence (ainsi que les méthodes à mettre en œuvre pour les combler) dans les premières version du protocole [SRC07], mais nous supposerons qu'elles ont pu être corrigées et que nous disposons d'un protocole DAA « idéal ».

applications en cours d'exécution sur la plate-forme, en prenant en compte l'ordre dans lequel elles ont été lancées. Si l'on considère le PCR comme une suite dont on ne conserve que le dernier élément, le premier est obtenu par le hachage du code du BIOS (formule (5.4)). Pour chaque nouveau programme de code P lancé sur la plate-forme (en commençant par le système d'exploitation), la nouvelle valeur du PCR est obtenue par l'application de la formule d'**extension** du PCR (formule (5.5)), à savoir un hachage de la valeur précédente concaténée avec le condensat correspondant au programme.

$$PCR_0 = h(BIOS) \tag{5.4}$$

$$PCR_{n+1} = h(PCR_n + h(P)) \tag{5.5}$$

Lors de la phase de démarrage de la machine, le contrôle n'est passé à la couche supérieure que lorsque le condensat de celle-ci a été vérifié et intégré au PCR. En comparant deux PCR, il est ainsi possible de déterminer si deux contextes d'exécutions sont identiques ou non. Cependant, à cause des propriétés du hachage cryptographique, il est pratiquement impossible à partir de la valeur courante du PCR :

- de retrouver une valeur antérieure du PCR ;
- d'identifier un ou plusieurs des programmes exécutés sans connaître l'ensemble de la séquence ;
- de forger une séquence d'exécution différente aboutissant à un PCR identique.

Attestation de l'environnement d'exécution

À l'aide du PCR et de méthodes similaires à celles utilisées pour l'authentification en tant que plate-forme de confiance, il est possible de convaincre un agent distant de la nature exacte de l'environnement d'exécution de la plate-forme de confiance. Si la plate-forme envoie à l'agent la mesure du PCR signée par une clé d'identité d'attestation, alors l'agent peut avoir confiance dans le fait que le PCR a bien été calculé par un TPM valide, reflétant donc fidèlement l'environnement d'exécution distant. Si l'agent distant dispose d'un PCR de référence, il peut le comparer au PCR fourni avant de prendre des décisions.

Attestation d'un programme particulier

Il est également possible à une plate-forme de confiance de communiquer sur une application en particulier, sans utiliser le PCR (qui décrit l'intégralité des logiciels lancés, ce qui peut donc être trop peu expressif pour de nombreuses interactions). Mark Dermot Ryan décrit un protocole d'attestation d'une clé AIK attachée à un programme particulier [Rya06a]. Ce protocole est quasiment similaire à celui que nous avons donné pour l'authentification en tant que plate-forme de confiance auprès d'une autorité de certification, à ceci près qu'en plus de la clé d'identité d'attestation générée, le condensat d'une application particulière est également signé par le TPM. Les informations envoyées au tiers de confiance sont donc celles de la formule (5.6) (où A est le code de l'application en question).

$$AIKpub, EKpub, \{AIKpub + h(A)\}_{EKpriv} \tag{5.6}$$

Le tiers de confiance, en plus de vérifier la signature, vérifie alors également que l'application A fait partie d'une liste d'applications « certifiées », autorisées, garantissant le respect d'un certain nombre de propriétés. La délivrance du certificat par le tiers de confiance est alors subordonnée à cette vérification, et la clé d'identité d'attestation est attachée, dans le certificat fourni, à ce programme particulier.

Cette méthode d'approbation d'un programme distant est une propriété qui nous sera très utile pour la mise en œuvre de la protection des données personnelles. En effet, elle permet à un agent d'avoir la certitude que le programme distant qui traitera ses données a reçu l'agrément d'une autorité de certification qu'il pourra lui-même avoir choisie.

5.2.1.5 Le chiffrement de données par le TPM

Le TPM peut être utilisé pour chiffrer des données de manière plus sûre, en utilisant la clé racine pour le stockage SRK (qui est spécifique au propriétaire du TPM : si un autre utilisateur prend possession du TPM, la clé est détruite). La clé SRK est utilisée pour chiffrer d'autres types de clés RSA : des **clés de stockage**, utilisées pour chiffrer d'autres clés RSA, et des **clés de liaison** (*binding keys*), utilisées pour chiffrer des clés symétriques. La clé symétrique K, qui servira à chiffrer les données elles-mêmes, est donc stockée dans l'ordinateur après avoir été chiffrée par toute une chaîne de clés RSA, remontant jusqu'à la clé SRK stockée dans le TPM. Pour chiffrer et déchiffrer les données, il est donc nécessaire que le TPM soit activé et laisse l'utilisateur accéder à la clé racine pour le stockage.

Suivant la chaîne de chiffrement utilisée (et via l'utilisation de procédures que nous ne détaillerons pas ici), la clé peut ou pas être utilisée sur une autre plate-forme de confiance. Ceci permet de différencier les données « liées » (déchiffrables sur une autre plate-forme) des données « scellées » (déchiffrables uniquement sur la plate-forme courante, par le TPM courant, avec le PCR courant[18], par le propriétaire courant).

5.2.2 Une architecture d'auto-profilage (*self-profiling*)

En 2002, Siani Pearson présente une architecture « améliorant la vie privée de l'utilisateur par l'auto-profilage » (sic) [Pea02a]. Cette architecture, fondée sur l'informatique de confiance, est l'une des premières dans laquelle celle-ci est utilisée avec l'idée d'améliorer la protection de l'utilisateur.

5.2.2.1 Génération et gestion des profils

Dans cette architecture, la machine de l'utilisateur (siège d'un éventuel agent utilisateur) est une plate-forme de confiance, et l'agent distant (l'agent de service) en reçoit des garanties. La plate-forme de confiance gère un certain nombre d'identités virtuelles, chacune attachée à un pseudonyme choisi par l'utilisateur et à une clé d'identité d'attestation AIK. Pour chaque identité, la plate-forme crée et gère un profil utilisateur correspondant à un modèle particulier (défini dans le cadre d'une application avec un agent de service distant). Ainsi, les informations de profil de l'utilisateur demeurent la plupart du temps sur sa plate-forme matérielle, choix technologique défendu comme le plus favorable pour le contrôle et la protection de ses données[19]. Ces données y sont protégées par les mécanismes de chiffrement du TPM afin d'assurer leur confidentialité à l'échelle de la plate-forme (permettant ainsi un contrôle d'accès local élaboré). Les informations de profilage sont collectées automatiquement par la plate-forme de l'utilisateur à partir de ses activités et suivant un processus préalablement établi par l'agent de service qui les utilisera. Le processus de profilage fait l'objet d'une attestation, via les mécanismes déjà

[18] Le scellement des données implique donc que les mêmes programmes (et seulement eux) soient exécutés sur la plate-forme lors de leur chiffrement et de leur déchiffrement. Le non-renouvellement d'une licence commerciale, le remplacement d'un composant matériel ou logiciel peut donc entraîner la perte des données ainsi protégées.

[19] Notamment par Riché et Brebner [RBG02, RB03] ou Deswarte et Aguilar-Melchor [DA06a].

pointés, afin de garantir à l'agent de service que les informations qui lui sont fournies reflètent fidèlement les caractéristiques comportementales et préférentielles de l'utilisateur.

La figure 5.6 illustre ce mécanisme de base de l'architecture. On y observe une inversion entre agent utilisateur et agent de service par rapport au schéma 5.4 décrivant les exigences du quatrième niveau de confiance que nous avons défini. Ceci signifie que l'architecture fournit prioritairement des garanties sur l'utilisateur à la plate-forme de service.

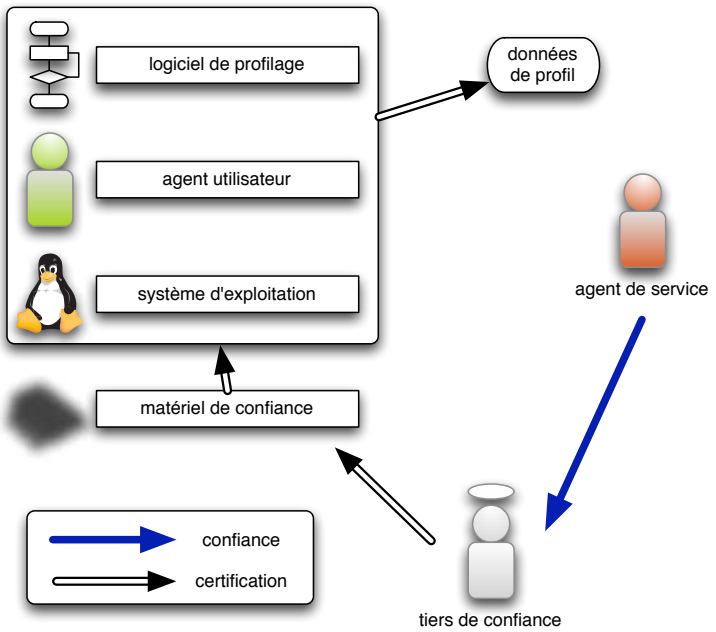

FIG. 5.6 – Principe de base de l'architecture d'auto-profilage

5.2.2.2 Garanties sur le traitement distant

L'agent utilisateur a également l'opportunité de vérifier si la plate-forme distante est une plate-forme de confiance, et éventuellement de poser des conditions sur son contexte d'exécution (attestation des programmes de traitement) avant de délivrer des informations de profilage. Ainsi, la plate-forme utilisateur peut théoriquement s'assurer que la couche logicielle qui utilisera ses données sur l'agent tiers a bien été certifiée par un tiers de confiance[20]. La figure 5.7 illustre cette amélioration optionnelle de l'architecture, qui amène davantage de symétrie dans les échanges entre agent de service et agent utilisateur[21]. Néanmoins, il semble raisonnable de penser que le contrôle en matière de production, distribution et certification du logiciel réside plutôt du côté du fournisseur de service que de l'utilisateur individuel. En effet, au vu des nombreuses applications potentielles de type client-serveur susceptibles d'utiliser des données personnelles et donc d'utiliser ce type d'architecture, il paraît intuitivement assez aisé pour un industriel d'obtenir un certificat pour son « code métier » sans que celui-ci ait été vérifié de manière approfondie par l'autorité de certification, ou encore de produire une version abusivement intrusive du code

[20] Cette possibilité n'est brièvement évoquée que comme une perspective d'extension de l'architecture.

[21] Pour des raisons de lisibilité, les relations de confiance ne sont pas représentées sur ce schéma. Chacun des agents est supposé faire confiance au tiers de confiance.

destiné à l'agent utilisateur.

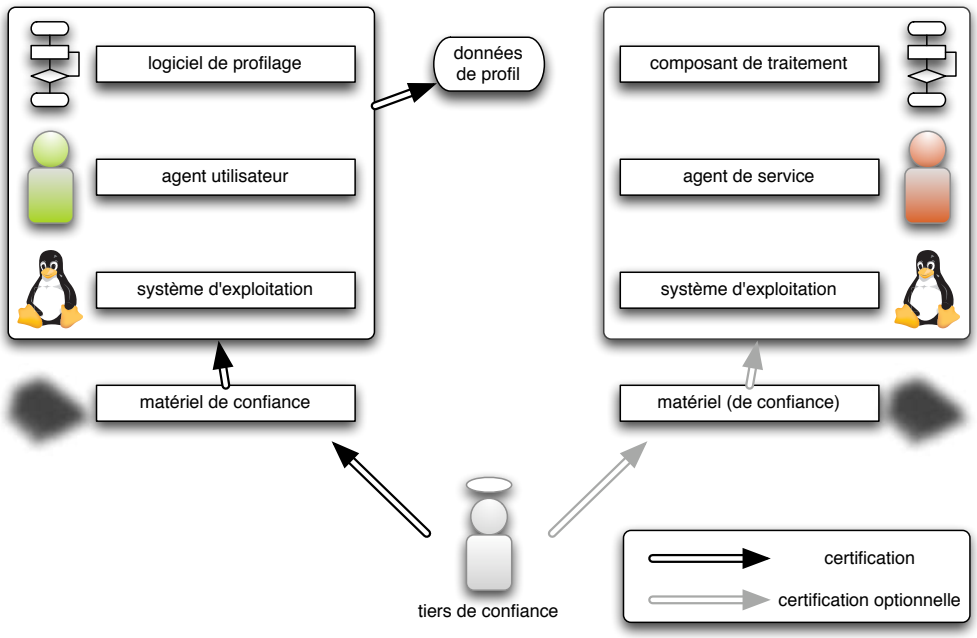

FIG. 5.7 – Amélioration optionnelle de l'architecture d'auto-profilage

5.2.2.3 Critique de l'architecture

Le point le plus important à noter concernant cette architecture semble toutefois la localisation d'un TPM, et donc d'une plate-forme de confiance, du côté de l'utilisateur. Ce sont principalement les agents de service, distants, qui obtiennent des garanties sur l'agent utilisateur (et sur ses données). Pour cette raison, la proposition semble en décalage avec le titre sous lequel elle est présentée, les agents de service étant davantage protégés contre les malveillances ou les erreurs de l'agent utilisateur que le contraire (à cause du plus grand contrôle que les fournisseurs de services peuvent avoir sur le logiciel distribué et sur sa certification par les tiers de confiance). Les mécanismes mis en place n'améliorent donc pas réellement la vie privée de l'utilisateur : ils sont clairement instaurés pour faciliter l'activité des agents de service. Cette facilitation passe par la communication d'informations de profilage générées automatiquement et sur lesquels l'utilisateur n'a finalement que peu de contrôle[22], à cause de l'utilisation qui est faite du mécanisme d'attestation.

Au final, ce type d'architecture est donc au contraire particulièrement aggressive pour la vie privée de l'utilisateur, les données étant l'objet d'un « contrôle étendu » (il est difficile ici de parler de protection) imposé par un agent tiers. Pour autant, l'architecture n'est présentée ici que dans ses principes. Sa mise en œuvre nécessiterait sans doute à plusieurs étapes du processus le développement de modules de négociations, dont les détails (suivant les choix effectués par

[22] Notamment, l'agent utilisateur n'a pas la possibilité de mentir sur les informations de son profil, ce qui est souvent considéré (par exemple par Dickinson *et al.* [DRB+03]) comme un outil efficace dans la protection de la vie privée des utilisateurs individuels.

les concepteurs) pourraient être propres à tempérer des critiques principalement fondées sur des possibilités de malveillance.

5.2.3 Une architecture utilisant les *sticky policies*

Nous avons vu que les *sticky policies* purement logicielles, introduites, par Karjoth et Schunter [KS02], relevaient d'un niveau de confiance insuffisant (correspondant au premier niveau de notre classification) pour la protection des données personnelles. Marco Casassa Mont, Siani Pearson et Pete Bramhall ont proposé en 2003, dans le cadre du projet PRIME [Eur08], une architecture utilisant l'informatique de confiance pour augmenter le niveau de sécurité d'un principe fondamentalement intéressant [CPB03].

5.2.3.1 Présentation de l'architecture

La figure 5.8 présente le scénario d'utilisation de l'architecture proposée par Casassa Mont *et al.*, impliquant un agent utilisateur, un ou plusieurs agents de services et un « agent d'audit », qui est en fait un tiers de confiance spécialisé.

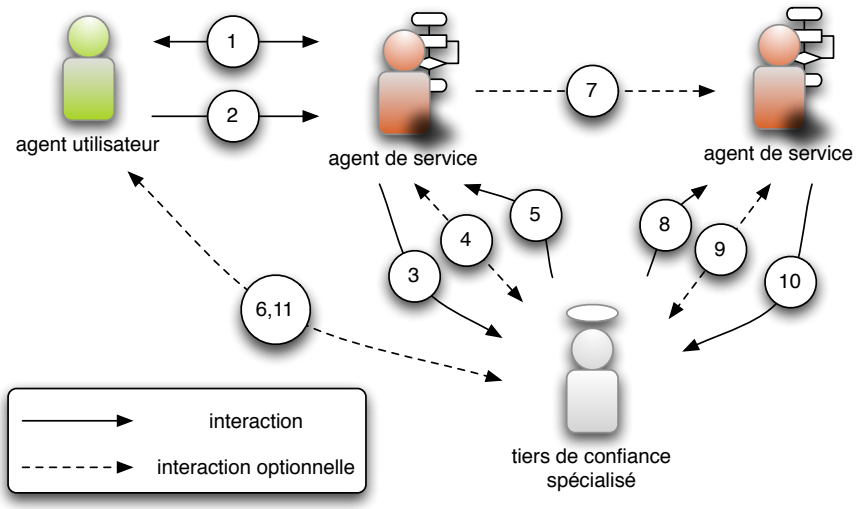

FIG. 5.8 – Architecture de Casassa Mont *et al.*

Les différentes étapes du scénario d'utilisation (comprenant des étapes optionnelles qui peuvent améliorer la protection de l'utilisateur) sont les suivantes :

1. **Négociation :** Un agent de service requiert des données de la part de l'agent utilisateur. Les deux agents négocient sur les politiques de protection des données à appliquer dans le cadre de la transaction.
2. **Transmission des données :** L'agent utilisateur chiffre les données à l'aide de l'algorithme IBE (*Identifier-Based Encryption* [BF01]). Dans le cadre de ce protocole, la clé de chiffrement est calculée à partir du texte de la politique négociée et sur la clé publique d'un tiers de confiance (l'agent d'audit) choisi par l'agent utilisateur et spécifié dans la politique. La clé de déchiffrement ne pourra être produite qu'à l'aide de cette même politique et de la clé privée de l'agent d'audit. Ce mécanisme assure que la politique attachée aux

données ne peut être substituée à une autre. L'agent utilisateur transmet ensuite à l'agent de service les données ainsi chiffrées, ainsi que la politique en clair.

3. **Demande de déchiffrement :** Lorsque l'agent de service souhaite accéder aux données, il contacte l'agent d'audit identifié dans la politique. Il lui transmet la politique d'accès aux données ainsi que ses créances (*credentials*), attestant qu'il remplit les conditions spécifiées par la politique.
4. **Authentification en tant que plate-forme de confiance (optionnel) :** Dans le cas le plus favorable à l'utilisateur, la politique impose que l'agent de service soit une plate-forme de confiance et se trouve dans un état déterminé (caractérisé par un PCR). Si c'est le cas, l'agent d'audit procède aux vérifications correspondantes.
5. **Génération et transmission de la clé de déchiffrement :** Si l'agent d'audit estime que les conditions d'accès aux données sont remplies, il génère une clé de déchiffrement pour les données en utilisant la politique et sa propre clé privée. La clé est transmise à l'agent de service, qui s'en sert pour accéder aux données. L'agent de service s'engage à ne pas conserver les données en clair. Cet engagement peut être vérifié si la politique initialement négociée requiert que l'agent de service soit une plate-forme de confiance et utilise un logiciel clairement identifié pour accéder aux données et les utiliser.
6. **Notification (optionnel) :** La politique peut prévoir qu'à chaque transmission d'une clé de déchiffrement par l'agent d'audit, celui-ci notifie l'agent utilisateur de la divulgation de ses données.
7. **Transmission des données (optionnel) :** L'agent de service peut éventuellement transmettre les données obfusquées, toujours attachées à leur politique, à un autre agent souhaitant les utiliser. Ce dernier suivra la même démarche pour accéder aux données. L'accès peut lui être refusé sur la simple base de son identité dans le cas où la politique interdit ou restreint la transmission des données.
8. **Demande de déchiffrement**
9. **Authentification en tant que plate-forme de confiance (optionnel)**
10. **Génération et transmission de la clé de déchiffrement**
11. **Notification (optionnel)**

5.2.3.2 Critique de l'architecture

Deux points faibles apparaissent à l'analyse de cette architecture.

Tout d'abord, les dispositifs qui protègent réellement les données de l'agent utilisateur, à savoir l'exigence d'une plate-forme de confiance et l'identification du logiciel qui traitera les données, sont optionnels. Il faudrait donc, lors de la phase de négociation, que l'agent utilisateur soit conscient de leur importance technique et qu'il puisse les imposer à l'agent de service. Or, l'un des problèmes sociétaux dans le domaine de la protection des données personnelles est le manque de prise de conscience de l'importance du problème par le grand public. C'est donc ici une porte ouverte à une protection de façade autorisant potentiellement de nombreux débordements, notamment la conservation (et la diffusion) de la clé de déchiffrement[23]. Les

[23] En effet, si l'agent de service n'est pas une plate-forme de confiance, alors l'interception de la clé de déchiffrement par un programme tiers est aisée à mettre en œuvre. La manipulation devra être effectuée à des niveaux différents (couches réseau, communication inter-processus, code de l'agent de service, code du composant de déchiffrement) suivant les mécanismes assurant la sécurité de la communication.

auteurs reconnaissent qu'une fois que l'agent de service a accès aux données en clair, il est difficile de l'empêcher de les utiliser à tort, malgré les outils de l'informatique de confiance.

D'autre part, un problème fondamental, commun à nombre d'architectures fondées sur l'informatique de confiance mais jamais soulevé à notre connaissance, est celui de la publicité et de la variété du code à certifier. En effet, si l'agent utilisateur peut imposer que ce soit un processus certifié qui traite les données, cette hypothèse requiert la certification de tout le code métier de tous les agents de service concernés. Les applications ne pouvant toutes reposer sur les même composants logiciels, la variété des services impose la variété du code. Le code à analyser est donc potentiellement très important en quantité et critique au regard des impératifs de l'agent de service : celui-ci ne souhaite sans doute pas (pour des raisons de concurrence entre acteurs économiques) que le fonctionnement interne de ses applications devienne public. Tout ceci, ajouté à la difficulté du problème de vérification du code (qui doit théoriquement être garanti sans faille, sans fuite de mémoire et parfaitement respectueux de n'importe quelle politique représentable) amène à saper la confiance que l'agent utilisateur peut avoir dans la valeur des certificats délivrés. Les propriétaires humains des agents utilisateurs peuvent en effet être amenés à penser que ces certificats en trop grand nombre, concernant de trop nombreuses applications, sont vendus aux agents de service sans analyse poussée des logiciels à certifier.

Cette architecture riche et prometteuse, théoriquement capable d'assurer la protection étendue des données à un niveau de confiance élevé, est donc fragilisée lors de son application à des cas réels. Des considérations très pragmatiques sur le fonctionnement de l'informatique, du commerce électronique et des sociétés numériques en général nous amènent donc à nous poser de nouvelles questions sur les solutions techniques à proposer au problème de la protection étendue.

5.2.4 Caractérisation des architectures possibles

L'observation de ces deux architectures ainsi que d'autres communications mettant en œuvre l'informatique de confiance dans le cadre de la protection de la vie privée [Pea02c, Pea05] peuvent nous aider à identifier quelques questions qu'il est bon de se poser si l'on veut caractériser la protection de l'utilisateur dans de telles architectures.

5.2.4.1 Localisation du TPM

Suivant l'architecture choisie, le TPM peut être localisé sur différentes plates-formes. Ces localisations distinctes correspondent à des caractéristiques différentes au niveau de la protection de l'utilisateur et des intérêts des diverses parties.

Sur la plate-forme utilisateur

Lorsque le TPM est situé côté utilisateur, ce dernier peut profiter des fonctionnalités de chiffrement de données offertes par la puce, et notamment la possibilité d'attacher les données à sa machine et à son environnement d'exécution (scellement de données). Néanmoins, les dispositifs de chiffrement fort sont d'une utilité limitée sur la plate-forme utilisateur, car c'est rarement cette dernière qui est l'objet d'une attaque directe[24].

Les inconvénients à cette utilisation du TPM côté utilisateur semblent largement contrebalancer ses avantages. Le scellement de données est en lui-même dangereux pour l'utilisateur.

[24] La récupération de données conservées côté service est en effet une opération nettement plus avantageuse pour un agent mal intentionné, qui dispose d'une cible unique et bien identifiée concentrant une quantité importante de données sensibles. Les attaques subies par les entreprises ou les administrations et visant les fichiers clients en sont la manifestation.

En effet, c'est un processus qui peut être imposé par un fournisseur de contenu pour accéder à des données dont l'utilisateur peut pourtant être le propriétaire. C'est ce dispositif, au cœur de la mise en œuvre des DRM matériels, qui est pointé comme le plus liberticide et qui a donné naissance aux controverses les plus significatives concernant cette technologie. Le scellement de données peut également être, au niveau économique, une entrave à la libre concurrence au sens où l'imposition du dispositif aux utilisateurs peut constituer (en fonction de la situation du marché) un outil d'aggravation de monopole. Ce biais profite aux éditeurs des logiciels et systèmes d'exploitation certifiés présents dans l'environnement d'exécution requis pour accéder aux données. Enfin, nous avons vu que le TPM, situé sur la plate-forme utilisateur, servait à produire des garanties à destination des autres parties, l'utilisateur lui-même n'en retirant que peu d'avantages.

Dans une optique centrée sur l'utilisateur, la localisation du TPM sur la plate-forme de ce dernier ne semble donc pas justifiée, voire même dangereuse pour lui.

Sur la plate-forme de service

Lorsque le TPM est situé côté service, l'agent correspondant dispose des fonctionnalités de chiffrement de la puce (qui peuvent lui être plus utiles que sur un poste utilisateur). La position du TPM permet en outre de fournir des garanties à l'agent utilisateur sur les composants logiciels utilisés côté service, ce qui constitue, nous l'avons vu, la principale condition de la protection étendue des données.

Cependant, ce type d'architecture impose une charge particulière à l'agent de service, notamment lorsque celui-ci doit maintenir un ensemble de logiciels certifiés. On observe également la même subordination de l'agent au TPM dans le cas du scellement de données. On peut néanmoins considérer que le risque associé est plus faible que lorsque le TPM est situé chez l'utilisateur, l'agent de service étant souvent en position de force par rapport à l'utilisateur sur le marché économique.

Le TPM, lorsqu'il est situé sur la plate-forme de service, ne semble donc pas présenter de danger particulier pour l'utilisateur. Au contraire, elle lui permet d'obtenir des garanties fortes sur les propriétés distantes. En revanche, les agents de service peuvent être handicapés dans leurs activités par cette configuration.

Sur une plate-forme tierce

Une alternative non envisagée dans les architectures présentées est la localisation du TPM sur une plate-forme d'exécution indépendante, ne dépendant ni de l'agent utilisateur ni de l'agent de service. Une telle plate-forme serait sous le contrôle d'un agent tiers n'ayant pas part à l'interaction entre l'agent utilisateur et le ou les agents de service. La position du TPM permettrait alors de fournir des garanties à toutes les parties concernant les propriétés valides sur la plate-forme d'exécution. L'agent tiers opérant la plate-forme n'ayant pas d'intérêt dans les transactions, il est a priori insensible aux problèmes engendrés par le scellement des données : la plate-forme étant conçue comme un environnement « neutre », elle ne traite pas de données pour le compte de l'agent tiers et celui-ci peut être considéré comme indifférent, dans une certaine mesure, aux actions qui s'y déroulent[25].

La déportation des traitements sur une telle plate-forme « banalisée » peut donc être profitable aux deux parties en leur proposant des garanties sur les propriétés distantes tout en

[25] L'agent tiers peut évidemment avoir des préoccupations de sécurité, de performances, de disponibilité... concernant sa plate-forme, mais on peut supposer en première approximation que ces notions sont peu dépendantes de la sémantique d'un traitement mettant en œuvre des données personnelles.

les libérant des contraintes de la présence d'un TPM sur leurs propres plates-formes. L'agent tiers, opérant la plate-forme d'accueil, hérite toutefois de la responsabilité de maintenir du code certifié. Il paraît donc intéressant de caractériser cette charge et son impact sur la protection étendue des données.

5.2.4.2 Nature et variété du code à certifier

Nous avons vu que dès lors que les composants de traitement utilisés par les agents de services sont variés et/ou critiques, les agents utilisateurs peuvent avoir de bonnes raisons de douter de la fiabilité de leur certification.

Le code certifié est-il privé ou public ?

Si le code source du composant est privé, alors les agents du système doivent forcément s'appuyer sur leur confiance dans l'autorité de certification pour accepter les garanties et les assertions concernant ledit composant. Il existera donc toujours la possibilité d'une certification de complaisance, celle-ci ne pouvant être contrôlée.

À l'inverse, si le code source est public (ou ouvert), alors en théorie n'importe quel agent peut effectuer des vérifications dessus et remettre en question la fiabilité de sa certification. La publication d'informations sur la vulnérabilité d'une portion de code peut amener à la révocation de son certificat, à la correction des failles et à une nouvelle certification. Cette possibilité de vérification par la communauté est l'un des arguments majeurs de la communauté *open source* d'une manière générale, et s'applique à merveille aux applications de sécurité, la politique de « sécurité par le secret » étant considérée comme une erreur en la matière depuis le XIXème siècle[26].

Quelle est la quantité de code certifié pour l'ensemble des applications ?

Lorsque la quantité et la variété du code à certifier augmentent, la difficulté à assurer la fiabilité du nombre de certificats correspondants augmente également. Plus le nombre de composants à certifier est élevé, plus le risque est élevé de voir apparaître de nouveau des certificats de complaisance, ou du moins fondés sur des analyses insuffisamment poussées.

Au contraire, si le code à certifier est limité, si la sécurité de toute une gamme d'application peut ne reposer que sur une portion de code bien identifiée, alors ce risque est amoindri.

En conséquence, le cas le plus favorable à l'utilisateur est donc celui où le code à certifier est à la fois public et peu varié, se limitant à quelques composants standardisés.

5.2.4.3 Protocole d'attestation utilisé

Il convient également de s'inquiéter du protocole utilisé pour la phase d'attestation, c'est-à-dire pour permettre à une plate-forme de confiance de s'identifier comme telle. Les alternatives possibles sont actuellement l'attestation utilisant les tiers de confiance et le protocole DAA.

Nous avons vu que le protocole utilisant les tiers de confiance était sensible à la collusion de ces derniers, mettant ainsi en péril l'anonymat des transactions effectuées par l'agent utilisateur. Nous avons également suggéré que lorsque l'agent utilisateur avait toute latitude dans le choix

[26] Le principe de Kerchoffs (reformulé au XXème siècle par Claude Shannon) affirme en effet que la sécurité d'un système doit pouvoir ne reposer que sur un secret de taille minimale (la clé), tout le reste du système étant potentiellement connu de l'attaquant.

de ces tiers de confiance, le risque pouvait diminuer en conséquence. Nous avons enfin affirmé (sans le montrer) que le protocole DAA pouvait en théorie permettre de s'affranchir tout à fait de ce risque de subornation des autorités de confiance.

Le protocole DAA serait donc plus avantageux pour la vie privée de l'utilisateur que le protocole classique utilisant les tiers de confiance. Ces deux méthodes étant théoriquement interchangeables, nous continuerons toutefois de représenter les mécanismes d'attestation comme des processus impliquant des tiers de confiance (en raison de leur moindre complexité théorique), en considérant que des protocoles à divulgation nulle de connaissance peuvent leur être substitués avec avantage.

5.3 Proposition d'architecture applicative complémentaire

À la lumière de nos observations, nous proposons une nouvelle architecture exploitant les possibilités de l'informatique de confiance. Cette proposition tient compte des axes de caractérisation que nous venons de mettre en lumière, afin d'offrir une orientation délibérément centrée sur la protection de l'utilisateur, quitte à poser des restrictions sur le type des traitements envisageables et sur les garanties offertes aux fournisseurs de services. Cette architecture applicative vient compléter l'offre déjà existante, en offrant un maximum de garanties à l'utilisateur dans les cas où elle peut être utilisée.

5.3.1 Architecture de base

Comme nous l'avons précédemment montré, la localisation du TPM sur une plate-forme d'exécution neutre comporte des avantages significatifs pour tous les agents prenant partie à la transaction. L'architecture que nous proposons repose donc sur l'existence d'une telle plate-forme, constituant un environnement d'exécution mis à disposition par une entité neutre (qui peut être un humain ou une organisation), n'ayant pas d'intérêt dans la transaction. Nous appellerons cette plate-forme (et par métonymie son propriétaire) l'**hôte** de la transaction.

5.3.1.1 Principe général

Le principe de l'architecture est le suivant :
- Phase préparatoire : un agent utilisateur, un agent hôte et un ou plusieurs agents de service s'accordent sur la fourniture d'un service numérique à l'agent utilisateur ;
- L'agent utilisateur et les agents de service projettent chacun un **agent mobile** sur la plate-forme hôte ;
- Phase d'exécution : les agents mobiles interagissent entre eux au sein de la plate-forme hôte et sous son contrôle, afin de produire le service ;
- Phase de fermeture : les agents mobiles transmettent à la plate-forme hôte le résultat de la production du service. La plate-forme transmet ce résultat à l'utilisateur et les agents mobiles sont détruits.

La plate-forme hôte est une plate-forme de confiance, sur laquelle s'exécute un système d'exploitation, un ou plusieurs **environnements d'exécution**, autant d'**agents d'interface** que d'environnements d'exécution ainsi qu'un **agent d'audit**, tous certifiés à code public. L'environnement d'exécution est un « bac à sable » généré indépendamment pour chaque interaction entre agents mobiles. Il garantit que ces derniers ne pourront communiquer avec l'extérieur et qu'ils seront convenablement détruits lors de la phase de fermeture. L'agent d'interface est l'interlocuteur des agents utilisateurs et de service. Il assure la migration des agents mobiles vers

l'environnement d'exécution ainsi que le retour des résultats du calcul à l'agent utilisateur. Il est le seul lien entre l'environnement d'exécution et l'extérieur de la plate-forme, la seule entité autorisée à communiquer. L'agent d'audit observe l'environnement d'exécution. Il effectue des mesures standardisées sur les agents mobiles afin de détecter des comportements dangereux pour la sécurité de la plate-forme et journalise les activités observables afin d'être en mesure de fournir des informations exploitables en cas de contentieux ultérieur entre les parties. Il est le garant du fonctionnement sain de la plate-forme hôte et a ainsi autorité pour mettre fin au traitement en déclenchant la fermeture de l'environnement d'exécution.

La principale caractéristique de cette architecture est l'assurance, fournie par la plate-forme hôte et mise en œuvre par ses divers composants, que les agents mobiles ne pourront pas communiquer avec l'extérieur et qu'ils seront détruits lors de la phase de fermeture. En conséquence, dans cette version de base de l'architecture, seul l'agent utilisateur bénéficie d'un retour. Les agents de service ne sont en mesure de retirer (ni donc de conserver) aucune information provenant de l'agent utilisateur.

La figure 5.9 présente les principes d'une l'architecture de base à partir d'agents mobiles. Nous allons maintenant détailler plus avant chacune des phases du protocole applicatif associé à l'architecture.

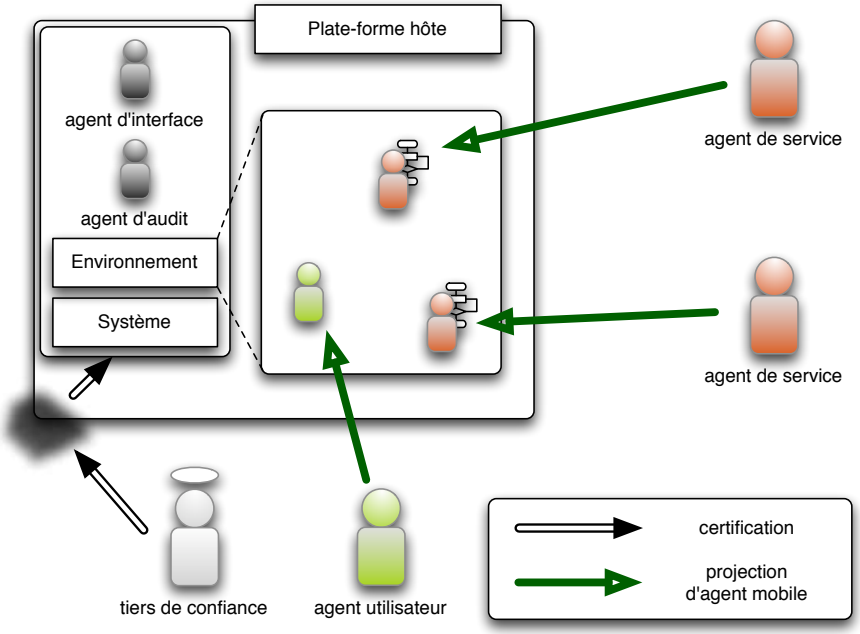

FIG. 5.9 – Architecture par agents mobiles

5.3.1.2 Phase préparatoire

Le prérequis à la mise en œuvre du protocole est la disponibilité d'une plate-forme de confiance utilisant un BIOS, un système d'exploitation et une suite logicielle d'accueil (comprenant agent d'audit, agents d'interface et environnements d'exécution) certifiés, à code public (l'ensemble de l'environnement logiciel correspondant à la valeur de registre de configuration *PCRstd*). La plate-forme matérielle comprend donc un TPM, de clé d'approbation (*EKpub*,

EKpriv).

On suppose que l'application concerne un agent utilisateur (situé sur l'un des terminaux d'un utilisateur humain) et un ensemble d'agents de service (situés sur des serveurs distants). Durant la phase préparatoire, agent utilisateur et agents de services s'accordent sur le service à rendre et négocient une politique d'utilisation de données personnelles associée[27]. À l'issue de cette négociation, les agents partagent un identifiant unique produit aléatoirement pour la transaction (*IDtrans*), l'identité d'une plate-forme d'accueil publique et une politique de traitement des données personnelles (exprimée dans un langage adapté comme P3P, *XACML privacy policies* ou SPARCLE).

Les agents procèdent ensuite à l'attestation de la plate-forme et à la vérification de l'intégrité de son environnement logiciel. Lorsque les agents font leurs premières requêtes à la plate-forme, en mentionnant l'identifiant de transaction *IDtrans*, cette dernière peut produire une clé d'identité d'attestation associée *AIK_IDtrans* ou utiliser directement sa clé d'approbation *EK*, puisque l'identité de la plate-forme n'est pas critique ici[28]. L'attestation et la vérification du registre de configuration de la plate-forme s'effectuent de manière classique, chacune des parties pouvant se référer à l'autorité de confiance de son choix ou recourir à un protocole à divulgation nulle de connaissance.

La plate-forme hôte est informée de l'intention des agents de procéder au traitement *IDtrans*, elle prend connaissance de la liste des agents participants et de la politique d'utilisation des données personnelles. Un environnement d'exécution et un agent d'interface dédiés à la transaction sont alors invoqués. Les agents fixes, localisés sur leurs plates-formes respectives, transmettent alors chacun à l'agent d'interface de la plate-forme hôte un composant logiciel spécifique, qui sera l'agent mobile les représentant.

5.3.1.3 Phase d'exécution

Durant la phase d'exécution, les agents mobiles intéragissent entre eux comme bon leur semble au sein de l'environnement d'exécution. Dans la version de base de l'architecture, les agents mobiles sont projetés avec toutes les informations dont ils ont besoin pour générer le contenu qui constituera le service rendu à l'utilisateur. Les agents mobiles n'ont pas la possibilité de communiquer avec l'extérieur.

Pendant cette phase, l'agent d'audit observe les interactions entre agents mobiles et décide, sur la base d'informations dépendant de la conception de la plate-forme, si un comportement anormal est détecté. Notamment, si les spécifications de la plate-forme imposent un langage de communication particulier entre les agents mobiles, toute tentative de l'un deux pour entrer en contact avec un autre agent mobile autrement que par ce langage sera considéré comme une attaque[29]. L'observation de métriques classiques sur le fonctionnement des agents, comme l'exploitation anormale des ressources de la plate-forme ou une absence d'activité prolongée, peut

[27] À ce stade, un agent utilisateur PAw vérifierait que cette politique est conforme avec son contexte normatif courant. Le résultat de cette évaluation peut être utilisé pour affiner la négociation et déterminer la décision finale de procéder ou non à la transaction. Néanmoins, la phase de négociation en elle-même ne rentre pas dans le cadre de nos travaux actuels.

[28] L'utilisation d'une clé d'identité d'attestation – et donc d'une identité virtuelle – pour la plate-forme peut toutefois se justifier, les différents agents pouvant conserver des historiques de leur transactions. Un agent malveillant serait donc capable d'identifier de manière unique une plate-forme hôte, cette dernière pouvant alors constituer une cible pour une attaque pouvant potentiellement mettre en péril (en fonction des options de protocoles choisies) le caractère privé de certaines informations concernant les transactions passées.

[29] Il existe quelques travaux d'évaluation des dispositifs de sécurité disponibles sur les environnements multi-agents les plus utilisés, comme l'étude effectuée par Axel Bürkle *et al.* en 2006 [BHMW06], qui identifie des pratiques « saines » de protection mutuelle de la plate-forme d'accueil et des agents mobiles.

également conduire à la caractérisation d'une anomalie de comportement de l'un des agents mobiles. Lorsque ces comportements anormaux surviennent, ils sont journalisés par l'agent d'audit. Ce dernier demande à l'agent d'interface d'informer les parties que la transaction a échoué à la suite d'une manœuvre non autorisée de la part de l'un des agents (permettant ainsi, dans une certaine mesure, l'utilisation de méthodes de régulation sociale). L'agent d'audit force ensuite la fermeture de l'environnement d'exécution.

Si la transaction s'effectue correctement, chacun des agents mobiles signale à l'agent d'interface la fin de son calcul, ainsi éventuellement qu'un résultat à transmettre à l'agent utilisateur.

5.3.1.4 Phase de fermeture

Lorsque l'agent d'interface dispose des rapports de tous les agents mobiles, il transmet les résultats fournis à l'agent utilisateur, informe les agents de service de la terminaison correcte de la transaction puis appelle la méthode de fermeture de l'environnement d'exécution, qui tue tous les agents mobiles ansi que l'agent d'interface. Les agents mobiles sont donc détruits sans avoir eu l'occasion de communiquer avec l'extérieur, et en particulier avec leur agent source.

Les informations de journalisation de l'agent d'audit sont conservées sur la plate-forme, de manière à pouvoir rester disponibles dans le cas où un contentieux juridique surviendrait à la suite de la transaction.

5.3.2 Améliorations optionnelles

L'architecture et le protocole associé, que nous avons présentés dans leur version de base, comportent des restrictions très fortes. Il est possible de lever certaines d'entre elles en introduisant des options dans le protocole.

5.3.2.1 Autoriser le paiement d'un service

L'extension qui semble probablement la plus évidente est celle qui permet le paiement du service par l'utilisateur. Une telle fonctionnalité augmenterait considérablement le champ d'application de cette architecture, l'étendant à certaines formes de commerce électronique. Les principes du *e-Cash* mis en place depuis les années 1980 permettent déjà le paiement anonyme d'un service. Il est aisé ici de proposer un prototype pour une option de paiement.

Il nous faut introduire un tiers de confiance spécialisé, à savoir un établissement bancaire[30]. On suppose que l'agent correspondant est connu des parties et dispose d'un couple de clés RSA $(BKpub, BKpriv)$. Lorsque l'identifiant de la transaction $IDtrans$ est partagé entre les parties, l'agent représentant la banque est informé de l'association entre le montant convenu et la valeur $h(IDtrans)$ par l'agent qui doit percevoir le versement. L'agent utilisateur peut procéder au paiement auprès de l'agent bancaire, en associant cette même valeur $h(IDtrans)$ à son versement. L'agent bancaire lui fournit en retour un certificat anonyme estampillé de la date courante T (5.7), à faire valoir auprès des agents de service, établissant que le montant convenu pour cette transaction particulière a bien été réglé[31]. Cette extension revient à ajouter à l'architecture des

[30] Les banques et les sociétés interfaçant les paiement électroniques sont déjà des tiers de confiance « de fait » dans les applications de commerce électronique. Si le contexte fait qu'un établissement bancaire ne peut plus être considéré comme un tiers de confiance, il est toujours possible d'adapter le protocole pour imposer que l'application de paiement soit certifiée et s'exécute sur une plate-forme de confiance.

[31] Le certificat de paiement peut éventuellement être chiffré à l'aide de la clé publique de l'agent devant recevoir le paiement, afin d'assurer que lui seul accède à l'information.

fonctionnalités d'autorisation préservant la vie privée, telles que présentées dans la section 1.2.4.

$$\texttt{payé}, h(IDtrans), \mathrm{T}, BKpub, \{\texttt{payé}, h(IDtrans), \mathrm{T}\}_{BKpriv} \tag{5.7}$$

Ainsi, la vérification de certificats de paiement par les agents fournisseurs de service peut être un prérequis à la projection des agents mobiles et à la phase d'exécution. Les certificats de paiement peuvent également être vérifiés directement par les agents mobiles sur la plate-forme d'exécution (à condition de mettre en place l'extension suivante), mais les agents de service perdent alors la possibilité de conserver le certificat de paiement. De la même manière, il est possible de mettre en place le paiement de frais de service à l'agent opérant la plate-forme d'accueil. Dans tous les cas les paiements peuvent rester anonymes du point de vue des agents prenant part à la transaction (par opposition au point de vue des établissements bancaires, qui maintiennent la nécessaire traçabilité des paiements).

5.3.2.2 Demandes de clés publiques

Une deuxième extension consiste à alléger la contrainte d'isolation des agents mobiles au sein de l'environnement d'exécution, pour les autoriser à demander des clés publiques par le biais de l'agent d'interface. Un agent mobile peut ainsi faire une demande à l'agent d'interface de l'environnement, qui prendra en charge l'exécution d'une requête paramétrée et la transmission du résultat au demandeur.

Une application de ce type de requête est l'évaluation au sein même de l'environnement d'exécution des certificats de paiement, mais les communications entre agents mobiles et le fonctionnement interne de ces derniers peuvent impliquer d'autres utilisations de clés publiques. Une plate-forme hôte peut éventuellement définir d'autres requêtes paramétrables, mais la certification de l'environnement logiciel hôte devra alors prendre en compte ces requêtes afin de déterminer leur absolue neutralité vis-à-vis des données personnelles impliquées dans la transaction entre agents mobiles.

Le caractère paramétrable de ces requêtes doit être limité pour éviter les transmissions d'informations par canal caché. Notamment, la liste des tiers de confiance auxquels il est possible de faire des requêtes doit être fixée par la plate-forme hôte et être évaluée lors de sa certification. En effet, si un agent mobile peut demander une clé publique à n'importe quel serveur, alors il est possible que le serveur choisi (complice d'un contournement de l'isolation logique de l'environnement d'exécution) permette une transmission d'information bilatérale : l'agent mobile peut remplacer l'identité de l'entité dont il demande la clé publique par des données chiffrées, le serveur peut répondre en remplaçant la clé publique par d'autres données chiffrées.

5.3.2.3 Autoriser des requêtes vers l'extérieur

L'interfaçage par la plate-forme hôte de requêtes des agents mobiles vers l'extérieur peut être généralisé, dans une certaine mesure. En 2007, Richard Cissée et Sahin Albayrak ont proposé, dans le contexte d'une architecture de recherche d'information, une méthode pour effectuer des demandes de données depuis une plate-forme isolée en préservant le caractère privé des requêtes et de leurs résultats [CA07]. Cette méthode s'appuie sur le chiffrement et déchiffrement croisé (via un algorithme de *one-time pad*) des requêtes et de leurs réponses par les deux parties prenant part à une transaction. Elle assure que les serveurs auxquels les requêtes sont adressées ne sont pas capables d'identifier les requérants ni d'effectuer de corrélations entre les requêtes.

Le protocole proposé peut être intégré à notre architecture, au prix de nouvelles fonctionnalités (notamment cryptographiques) pour l'agent d'interface. Les agents mobiles devront

également tous disposer de certaines fonctionnalités de chiffrement spécifiques pour que l'un d'eux puisse émettre des requêtes par le biais de ce protocole.

Cette nouvelle option permet de lever une autre limitation majeure de l'architecture de base, qui imposait que les agents mobiles soient projetés avec toutes les informations nécessaires à la transaction. Grâce à cette option, les agents mobiles peuvent désormais se reposer sur des bases de données extérieures pour des opérations de recherche d'information.

5.3.2.4 Conservation de données sur la plate-forme

Le protocole peut également être étendu pour permettre la conservation d'informations sur la plate-forme hôte. En fait, certaines informations sont déjà conservées par l'agent d'audit, mais elles sont de nature statistique et ne référencent pas directement les données personnelles ni les traitements. Ces informations ne constituent donc pas une menace pour le caractère privé des données appartenant aux agents mobiles ou à leurs propriétaires.

Une attention plus grande doit être apportée au problème si l'on autorise les agents mobiles à faire des requêtes de stockage et de récupération de données à l'agent d'interface. Une telle fonctionnalité permettrait d'enrichir les services fournis à l'utilisateur (en assurant la persistance de son profil ou de ses préférences, par exemple) et d'étendre le champ d'applicabilité de l'architecture. Néanmoins, deux problèmes majeurs se posent.

Tout d'abord, la responsabilité de la plate-forme par rapport à la protection des données deviendrait véritablement énorme. En effet, lorsque des données personnelles sont stockées, le responsable de l'hébergement a l'obligation d'assurer la protection de ces données. En l'absence de cette option, les agents de la plate-forme peuvent dans une certaine mesure ignorer la sémantique des informations échangées. Si l'agent d'interface devient responsable de l'accès à des données personnelles stockées sur la plate-forme, alors ce dernier doit pleinement comprendre la nature de ces données, s'assurer de manière fiable de l'identité des agents souhaitant les stocker ou y accéder, comprendre le détail des politiques de traitement de ces données, s'assurer de la persistance de ces politiques entre deux transactions (alors même que l'agent d'interface n'est pas persistant) et enfin appréhender un contexte normatif pour s'assurer que son propre comportement n'est pas délictueux. Le pas à franchir est énorme en matière de complexité de la plate-forme, et ces nouvelles fonctionnalités seraient extrêmement délicates à valider au cours de la phase de certification de l'environnement logiciel.

D'autre part, la plate-forme d'accueil étant conçue comme un service « bien connu » d'un certain nombre d'agents, celle-ci deviendrait une cible privilégiée pour des agents mal intentionnés souhaitant collecter des données sensibles de manière déloyale. Une architecture proposant cette option retomberait dans les travers de propositions comme IDsec [Int] en mettant en avant un point faible unique et en limitant le contrôle des agents utilisateurs sur leurs données personnelles.

En conséquence, il nous semble qu'une telle fonctionnalité constituerait au final un risque trop élevé pour la vie privée des utilisateurs de l'architecture.

5.3.3 Lien avec les institutions électroniques

L'architecture que nous proposons peut être reliée au concept d'**institution électronique** initialement proposé par Carles Sierra et Pablo Noriega [SN98] et enrichi depuis par de nombreux chercheurs, notamment Marc Esteva et Andrés García-Camino [ERAS+01, ERRAA04, GCNRA05]. Les institutions électroniques désignent un modèle applicatif particulier, destiné à implémenter un systèmes normatif centralisé mais ouvert, accueillant des agents pouvant avoir

été conçus par des entités distinctes. Ce modèle est notamment fondé sur le *middleware* AMELI [ERRAA04], qui constitue une plate-forme d'accueil pour des agents mobiles, à la manière de l'architecture que nous avons présentée.

Un agent désireux de participer à une transaction au sein d'une institution électronique doit projeter un agent délégué, appelé *governor*, qui doit implémenter une certaine interface d'interaction de manière à assurer sa compatibilité. Sur la plate-forme, les *governors* s'exécutent sous le contrôle d'agents *staff*. Ces derniers sont les garants du respect des normes en vigueur dans l'institution. Ils ont la capacité d'analyser le fonctionnement interne des agents *governors* et éventuellement d'appliquer des contraintes à leur exécution. Il s'agit donc, ici également, d'agents mobiles venant s'exécuter sur une plate-forme d'accueil sur laquelle certaines règles doivent être respectées.

Il semblerait au premier abord que la mise en œuvre actuelle des institutions électroniques, fondée sur AMELI, puisse encoder la partie logicielle de l'architecture que nous présentons, l'agent d'audit et l'agent d'interface étant représentés par des agents *staff*. Il faut pour cela que les normes de l'institution soient définies de manière à ne jamais laisser les agents *governors* communiquer avec l'extérieur ni conserver d'informations à la fin de la transaction. À ce sujet il convient de noter que les normes de l'institution ne jouent pas le même rôle que les normes DLP : alors que les normes DLP servent à définir des politiques de manipulation des données, les normes de l'institution vont définir le protocole d'interaction des agents mobiles, et notamment leur imposer de respecter ces politiques[32]. Afin de s'assurer que les institutions électroniques permettent de mettre en œuvre ce protocole, il faudrait donc déterminer si le langage de normes utilisé permet de représenter la sémantique des normes DLP, éventuellement restreintes aux agents mobiles.

Cependant, la plate-forme AMELI et la notion d'institution électronique ne peuvent à elles seules remplacer cette architecture, qui repose principalement sur les fonctionnalités du *Trusted Computing*. Pour correspondre à notre proposition, l'institution électronique d'accueil, en plus de modéliser le langage DLP et les contraintes inhérentes à l'architecture, doit donc être déployée sur une *Trusted Computing Platform* et mettre en œuvre dans le *middleware* les mécanismes spécifiques que nous avons présentés, de telle façon que les agents *staff* puissent raisonner sur les résultats correspondants, ce qui n'est pas nativement possible dans AMELI. Il nous semble donc que si les institutions électroniques sont un candidat potentiel pour la réalisation pratique de cette architecture, une interaction plus forte entre matériel et logiciel doit être introduite dans la plate-forme AMELI pour que cette dernière puisse assurer les fonctionnalités désirées.

5.3.4 Critique de l'architecture

L'architecture à base d'agents mobiles que nous avons proposée n'a pas pour objectif de résoudre tous les problèmes qui peuvent se poser ni d'être applicable à n'importe quel scénario transactionnel entre agents. Son intérêt principal est de mettre en lumière certains des problèmes posés par les architectures existantes utilisant l'informatique de confiance, de les présenter d'un point de vue strictement centré sur l'utilisateur et de montrer comment ils pourraient être résolus.

En conséquence, l'applicabilité de cette architecture est limitée. Ces limitations peuvent être partiellement contournées par l'utilisation des extensions que nous avons proposées, mais il de-

[32] Les auteurs proposent également des méthodes de décision de la consistance de l'ensemble des normes [EVSRA04, ERASV04], mais qui s'intéressent uniquement à la vérification de l'absence de conflit interne. Ces travaux sont davantage à mettre en relation avec la vérification formelle de protocole qu'avec nos développements sur la détection et la résolution des conflits d'obligations entre autorités multiples.

meure évident qu'elle n'est pas adaptée pour tout type d'application. Nous avons notamment pointé le fait que la persistance des informations ne peut être envisageable que sur la plate-forme de l'agent utilisateur et donc sous son contrôle, conformément aux principes jugés favorables à la protection des données personnelles. Ce choix quelque peu dogmatique peut être un frein à l'utilisabilité des applications, mettant ainsi en lumière le traditionnel compromis entre protection de l'utilisateur et fluidité d'utilisation.

D'autre part, de par le choix délibéré de ne renvoyer d'informations qu'à l'utilisateur lors de la phase de fermeture, nous interdisons les transactions impliquant deux agents utilisateurs réclamant tous deux des données en sortie de l'interaction. Si l'on autorisait de telles interactions (même en les découpant en plusieurs transactions indépendantes), on permettrait à des informations provenant d'un agent utilisateur d'être transmises à un autre agent utilisateur, sans contrôle sur la légalité de cette transmission (puisque dans l'état actuel des choses et pour des raisons de généricité, la plate-forme hôte ne maîtrise pas la sémantique des données échangées). La levée de cette limitation exigerait le passage d'un nouveau palier technique pour la plate-forme d'accueil qui consisterait en une augmentation significative de la complexité de sa conception, amputant d'autant la confiance pouvant être fondée sur sa correction.

5.4 Discussion

Si le principe des *Trusted Computing Platforms* permet d'augmenter sensiblement la confiance dans les caractéristiques d'un traitement distant (chose particulièrement difficile à assurer sans contrainte sur le matériel distant), ce n'est pas nécessairement la seule solution possible à ce problème. D'autre part, il convient de prendre acte des nombreuses critiques formulées à propos de cette technologie, à cause des dangers réels que certains types d'implémentations et d'utilisations peuvent faire courir à la vie privée des utilisateurs individuels. L'informatique de confiance a été l'objet, depuis les premières esquisses de spécifications sur la base du projet Palladium de Microsoft, d'analyses critiques et d'une polémique intense. Ce n'est pas notre intention ici de relayer cette polémique, mais il nous semble nécessaire de pointer les critiques les plus importantes et d'en tirer des conclusions sur l'utilisation de l'informatique de confiance pour assurer la protection des données personnelles.

5.4.1 Critiques générales de l'informatique de confiance

Parmi les promoteurs de l'informatique de confiance, Siani Pearson s'est essayée à plusieurs analyses des risques présentés par la technologie [Pea02c, Pea05], en les minimisant toutefois par une prise de position très optimiste et conciliante en ce qui concerne la bonne volonté des futurs utilisateurs industriels. Ross Anderson publie sur son site web une « foire aux questions » qui constitue une critique largement diffusée et particulièrement virulente de la technologie [And03], analysant son potentiel liberticide avec l'œil le plus pessimiste possible. On peut également noter une contribution de Moti Yung [Yun03] qui se veut une synthèse objective des différentes analyses. Le point de vue du groupe de travail de l'Union européenne sur la question mérite également notre attention [Art04]. Le groupe de travail a eu l'occasion de collaborer à l'élaboration de la version la plus récente des spécifications du TPM. Le document de synthèse qui résulte de sa collaboration avec le TCG comporte leur appréciation de la technologie ainsi qu'une série de recommandations informelles (comme celles de favoriser le protocole DAA ou de désactiver le TPM par défaut).

Les deux principales critiques, formulées sous l'hypothèse que le TPM est situé sur la plate-forme utilisateur, ce qui est le cas le moins favorable, sont les suivantes :

- Dans une certaine mesure, le TPM retire à l'utilisateur une partie du contrôle de sa plate-forme pour le transférer à une entité externe. C'est la critique qui a déclenché les réactions les plus importantes. Elle se base sur la très forte orientation initiale de la technologie vers les DRM matériels ainsi que sur les premières propositions de spécifications qui permettaient au TPM de détruire, sur la plate-forme utilisateur, des logiciels piratés ou dont la licence aurait expiré.
- L'informatique de confiance peut être un outil important d'aggravation de monopole. Si un fournisseur de contenu numérique dispose d'une pénétration de marché suffisante, il peut en effet imposer que ses contenus ne soient lus que par une certaine suite logicielle, certifiée par une autorité imposée. L'utilisateur est alors arbitrairement soumis à l'autorité de l'éditeur de la suite en question, et plus particulièrement de sa politique en matière de pérennité des licences, de support et maintenance des versions historiques et de révocation des certificats.

Ces critiques sont minimisées par Siani Pearson, qui met en avant la possibilité toujours donnée à l'utilisateur de prendre possession du TPM ou de le désactiver. Cet argument n'est cependant pas applicable à la seconde critique majeure, car dans ce cas l'utilisateur est justement contraint d'utiliser le TPM de la manière imposée par le fournisseur de contenu, sous peine de ne pouvoir accéder aux données.

Le groupe de travail de l'Union européenne exprime un point de vue assez proche de celui de Mark Dermot Ryan [Rya06a] (et généralement assez répandu) en affirmant que la technologie en elle-même est neutre, mais qu'il est dangereusement aisé d'en faire une utilisation néfaste pour les libertés individuelles des utilisateurs.

Il semblerait cependant que la vocation initiale de la technologie en tant que vecteur des DRM matériels et de la protection des fournisseurs de contenus contre un utilisateur supposé dangereux ne soit plus vraiment d'actualité, si l'on écoute les acteurs majeurs du consortium quant à leurs priorités stratégiques. Les développements de ces dernières années dans les spécifications du TCG sont probablement un signe de ce changement dans les usages programmés de la technologie. On peut notamment remarquer que des spécifications existent maintenant pour l'utilisation de l'informatique de confiance sur des terminaux mobiles, mais également sur des architectures de serveurs (ce qui permet éventuellement de fournir des garanties à l'utilisateur). Cependant, plusieurs implémentations existantes des spécifications de l'informatique de confiance (notamment la technologie LaGrande d'Intel et la technologie NGBSC de Microsoft) vont au-delà des spécifications officielles, allant vers une plus grande protection des entrées et sorties, ce qui montre probablement un certain intérêt pour le contrôle des flux multimédia et peut faire douter des déclarations allant dans le sens de l'abandon des DRM matériels.

5.4.2 Pertinence de l'utilisation de l'informatique de confiance pour la protection étendue des données personnelles

En 2005, Siani Pearson dit penser que l'informatique de confiance ne peut résoudre tous les problèmes de la protection des données personnelles, à cause du rapport de force existant entre prestataires et utilisateurs de services [Pea05]. Elle admet également que si les identités virtuelles, qui sont une des fonctionnalités natives de l'informatique de confiance, sont un outil majeur pour la protection de la vie privée, elles peuvent tout à fait être mises en œuvre sans utiliser un TPM, ce dernier n'apportant pas de garanties supplémentaires à l'utilisateur mais seulement à ses interlocuteurs. Elle reconnaît de plus que l'intérêt du consommateur n'a jamais été une motivation majeure pour le consortium et que la possibilité d'une utilisation commercialement abusive reste un risque non maîtrisé.

À la lumière des critiques précises et bien fondées adressées au consortium, ainsi que des éclairages apportés par les chercheurs travaillant eux-mêmes à la promotion et à l'élaboration de cette technologie, il nous apparaît que l'utilisation de l'informatique de confiance risque d'introduire davantage de risques pour l'utilisateur qu'elle n'apporte de garanties supplémentaires. Il nous semble que l'avenir de la protection étendue des données personnelles réside sans doute dans un niveau de confiance moins élevé, comme le troisième de notre classification, dans lequel le matériel distant est jugé digne de confiance sans qu'il soit besoin de le certifier. Il nous semble prometteur de rechercher des technologies de compromis, apportant des garanties moins fortes à l'utilisateur mais n'encourageant pas l'utilisation ni le développement d'une technologie par trop susceptible de lui porter atteinte.

Troisième partie

Mise en œuvre

Chapitre 6

Implémentation

6.1 Utilisation de la plate-forme JACK

Nous proposons l'implémentation d'un démonstrateur du modèle d'agent PAw à partir d'une architecture d'agent JACK [AOS], à l'aide de la plate-forme de développement correspondante, utilisant sur une surcouche orientée agent du langage Java inspirée par le modèle PRS [GL87]. Cette plate-forme dispose notamment de fonctionnalités permettant une mise en œuvre efficace de procédures de sécurité pour contrôler l'accès aux croyances sensibles. La couche logique de l'agent fait usage d'une machine virtuelle Prolog, interfacée avec le modèle d'agent JACK. Les relations de l'agent avec les protocoles applicatifs sont développés en Java, langage de base du modèle JACK.

6.1.1 Exploitation des concepts de JACK

L'utilisation de JACK nous sert principalement à figer un modèle d'agent et à nous appuyer sur des mécanismes de gestion préexistants. Ainsi, nous disposons déjà d'un agent BDI procédural, d'une gestion efficace des événements internes, des messages, de bases de croyances, de la gestion de plans assurant les activités de l'agent...

6.1.1.1 Portail, environnement et agents

Pour JACK, une machine virtuelle Java est associé à un « portail », qui constitue le point d'entrée et de sortie des communications extérieures pour tous les agents hébergés sur cette machine virtuelle, ainsi que le préfixe de leur adresse.

Nous avons conçu un agent spécifique chargé de modéliser l'environnement des agents. Il est créé au démarrage et accède aux fichiers de configuration et de sauvegarde pour donner naissance à un certain nombre d'autres agents : agents utilisateurs (PAw), agents de service (PAw ou non), autorités normatives. Tous les agents disposent d'une référence sur cet environnement, qu'ils peuvent notamment utiliser pour envoyer des données à une console de démonstration.

La communication entre agents (totalement prise en charge par JACK) se fait par le biais d'objets sérialisés de type *MessageEvent*, héritant des objets *Event* servant à modéliser les événements internes des agents. L'agent d'environnement n'a aucun rôle dans la communication entre agents, sinon celui d'annuaire. C'est le portail qui assure la transmission des messages.

6.1.1.2 *Capabilities*

Les différents éléments des agents sont regroupés dans des conteneurs nommés *capabilities*, qui nous permettent d'isoler dans un module unique les compétences nécessaires à un agent pour être, par exemple, un agent PAw, une autorité normative, un client d'autorité normative, un agent de service compatible PAw... Ce sont ces modules qui sont destinés à être publiés pour servir au développements d'autres agents (lesquels utiliseront, suivant leurs besoins, un seul ou plusieurs de ces *capabilities*).

6.1.1.3 Plans

Les plans sont les méthodes effectives de l'agent, dont l'exécution est déclenchée par la réception d'événements internes ou de messages en provenance d'autres agents. Ce sont dans les plans que sont codées les fonctionnalités de l'agent. Plusieurs plans peuvent être simultanément en cours d'exécution au sein d'un même agent, dans des *threads* séparés sous le contrôle du moteur JACK.

6.1.1.4 *BeliefSets*

Nous utilisons les *BeliefSets* (bases de croyances JACK) pour manipuler la majeure partie des données internes de l'agent :
- Contexte normatif (autorités et leur prévalence associée) ;
- Normes ;
- Historique des actions et actes de langages répertoriés en DLP ;
- Données identifiées comme ayant un caractère personnel ;
- Autres données.

Ces structures de données, sous la forme de bases de tuples, nous permettent d'aborder les croyances sous un aspect relationnel (ou logique) à l'aide de requêtes spécifiques utilisant un mécanisme d'unification. Elles disposent également de méthodes de rappel qui permettent de déclencher l'exécution d'un plan lorsqu'on tente d'accéder aux données, qu'on tente d'en insérer, d'en supprimer, qu'une erreur d'insertion se produit... Ces méthodes de rappel seront à la base de nos mécanismes de protection locale des données personnelles.

6.1.2 Interface avec Prolog

Nous nous reposons sur SWI-Prolog (au travers de l'interface bidirectionnelle JPL entre la machine virtuelle Java et la machine virtuelle SWI-Prolog) pour mettre en œuvre des traitements de nature essentiellement logique, comme la détection et l'arbitrage des conflits, ou bien des traitements plus facilement implémentables en Prolog comme l'analyse syntaxique. Les appels à Prolog, synchrones seront toujours effectués depuis l'exécution d'un plan JACK spécifique.

6.2 Couche de raisonnement normatif

La couche de l'agent PAw chargée du raisonnement normatif est répartie entre des « plans » JACK dans le modèle de l'agent et des composants spécifiques écrits en SWI-Prolog, auxquels les plans font appel. Cette couche assure les fonctionnalités suivantes :
- Elle demande et réceptionne les normes édictées par les autorités normatives que reconnaît l'agent. Ces normes sont écrites suivant notre syntaxe particulière, RDLP (pour *Restricted DLP*), qui est un sous-ensemble de DLP.

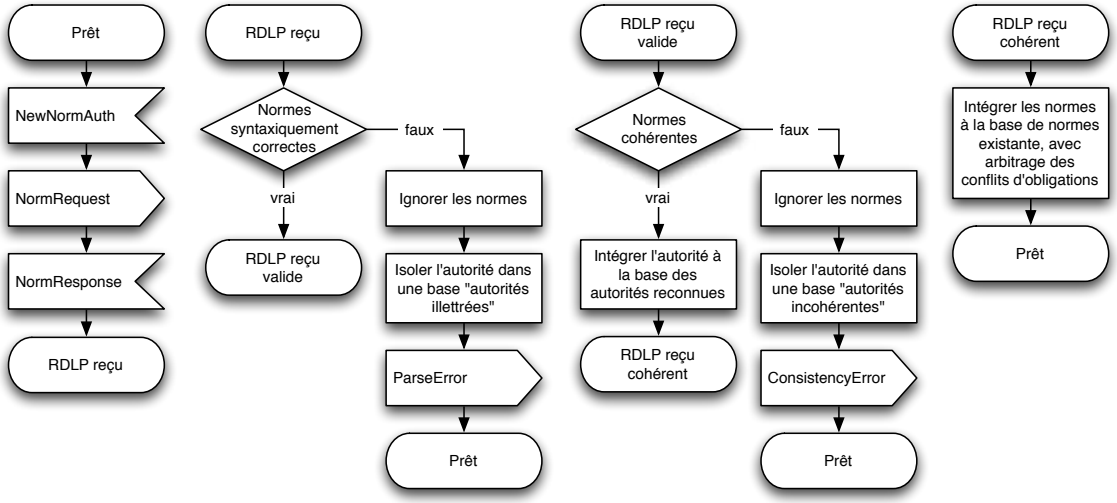

FIG. 6.1 – Automate de fonctionnement de l'agent PAw lors de l'ajout d'une nouvelle autorité normative

- Elle s'assure de la correction syntaxique des normes et vérifie leur cohérence pour chaque autorité (absence de conflits locaux à une autorité).
- Elle intègre les normes nouvellement réceptionnées et vérifiées à la base de norme active, en détectant et arbitrant les conflits d'obligations qui pourraient survenir.
- Elle met à disposition de l'agent une base de normes cohérente pouvant constituer sa politique de traitement des données personnelles.

L'agent PAw dispose au sein de ce module d'une base d'autorités actives et de deux bases d'autorités inactives : une base d'autorités « illettrées » produisant des erreurs de syntaxe, et une base d'autorités « incohérentes » émettant des normes impossibles à appréhender pour l'agent. L'agent possède également une base de normes principale, maintenue cohérente et utilisable, plusieurs bases de normes de transition sur lesquelles il travaille, une base de normes de nature permissive (traitées à part comme nous l'avons vu) et une base de normes désactivées au cours de l'arbitrage des conflits.

L'ajout d'une nouvelle autorité normative est une fonctionnalité qui concentre les activités de la couche de raisonnement normative et qui mérite que nous y prêtions particulièrement attention. La figure 6.1 est une modélisation simplifiée[1] du fonctionnement de l'agent PAw lorsqu'il doit prendre en compte une nouvelle autorité dans son contexte normatif. Dans ce schéma, les deux procédures de décision (analyse syntaxique et vérification de cohérence) ainsi que la détection et l'arbitrage des conflits d'obligations dans le dernier module de traitement sont confiées à Prolog (ou du moins à une étroite collaboration entre JACK et Prolog). Le dernier module de traitement sera détaillé plus avant dans la suite de ce chapitre.

Les états que nous avons utilisés ici pour étiqueter la progression de l'agent PAw sont les suivants :

1. Dans l'état *Prêt*, la base de normes de l'agent PAw est exempte de conflits d'obligation et à jour au vu des autorités qu'il doit considérer.

[1] On fait ici appel à une version graphique du *Specification and Description Language* (SDL) utilisant notamment les éléments suivants :

2. Dans l'état *RDLP reçu*, l'agent PAw a réceptionné un nouvel ensemble de normes à ajouter à sa base, mais ne l'a pas encore examiné.
3. Dans l'état *RDLP reçu valide*, l'agent PAw a vérifié que ces nouvelles normes sont syntaxiquement correctes.
4. Dans l'état *RDLP reçu cohérent*, il s'est également assuré qu'elles ne contenaient pas d'inconsistances logiques.

6.2.1 Le formalisme RDLP

La logique DLP permet la construction de formules (et donc de normes) fort complexes. Dans la section 4.3.2 nous avons notamment montré son expressivité dans le cadre des normes datées, qui ont un intérêt particulier en protection des données personnelles.

Cependant, dans le but de correspondre plus précisément à la structure des normes que nous sommes amenés à rencontrer (à l'image des exemples que nous avons développés dans la section 4.3.3) et également de simplifier la mécanisation de l'inférence logique, nous choisissons de définir un schéma de formules particulier, nommé *Restricted DLP* (ou RDLP). Nous posons l'hypothèse simplificatrice, restrictive mais justifiée au regard de nos précédentes observations, que les autorités normatives émettent des normes qui sont des formules RDLP plutôt que DLP. Cette hypothèse permettra par la suite des traitements plus efficaces pour la détection et la résolution des conflits d'obligations.

D'un point de vue formel, il apparaît que les logiques fondées sur un produit entre logique déontique et logique temporelle peuvent être d'une complexité calculatoire élevée. Nous travaillons ici sur l'hypothèse que le fait de limiter les interactions possibles entre modalités déontiques et modalités temporelles peut permettre de borner cette complexité [ÅvdHR+07]. Ces considérations sont néanmoins purement conjoncturelles en l'état actuel de nos connaissances, elles devront donc être soumises (en l'absence de résultats théoriques) à une évaluation quantitative par l'expérimentation.

La réduction syntaxique correspondant à RDLP repose principalement sur l'utilisation de normes en forme de règles (une condition d'application, potentiellement complexe, activant une obligation ou une interdiction) et le recours aux normes datées pour encadrer les interactions entre modalités temporelles et déontiques.

Il faut bien garder à l'esprit que RDLP n'est pas une nouvelle logique plus faible que DLP, c'est une famille de formules DLP respectant un certain formalisme. Les autorités normatives édictent leurs normes en suivant le format RDLP, mais lorsqu'on applique les mécanismes d'inférence de la logique DLP à ces normes, les formules qui en résultent sortent bien évidemment du cadre de RDLP.

6.2.1.1 Hypothèses fondatrices de RDLP

Nous posons un certain nombre d'hypothèses qui nous permettrons de poser des restrictions syntaxiques sur le langage, dans une optique de simplification des traitements :

- **Normes conditionnelles :** Les formules RDLP sont exprimées sous la forme de normes conditionnelles, comportant un antécédent (les conditions d'application, de validité ou de déclenchement) et un conséquent (la norme en elle-même). De nombreux autres travaux sur les politiques de sécurité et la logique déontique, comme le chapitre dédié au sujet dans l'ouvrage de Brian F. Chellas [Che80] ou les propositions de Laurence Cholvy et Frédéric Cuppens [CC97] soulignent leur caractère représentatif et l'importance de leur rôle.

- **Une seule autorité normative par norme :** Le langage DLP permet d'introduire dans une même formule des modalités déontiques correspondant à des autorités normatives différentes. Cette possibilité est absolument nécessaire pour permettre un raisonnement riche (notamment pour la détection et l'arbitrage des conflits d'obligations), mais au moment de leur énonciation il paraît raisonnable que les normes ne fassent référence qu'à la seule autorité qui les édicte.
- **Une seule modalité déontique par norme :** C'est une restriction supplémentaire par rapport à la précédente. Nous avons vu, de par la première hypothèse, que les modalités déontiques sont cantonnées au conséquent de la norme conditionnelle, elles n'apparaissent pas dans les conditions d'application. De manière simplifiée, les modalités déontiques seront, au sein du conséquent, articulées par des conjonctions ou des disjonctions. Une conjonction peut être très simplement éclatée en plusieurs normes ayant le même antécédent. Quant aux disjonctions, il ne paraît pas raisonnable qu'une autorité édicte une disjonction sur des concepts déontiques. Elle pourrait éventuellement (quoique cela demeure particulièrement contestable) émettre une obligation (par exemple) portant sur une disjonction de deux actions, laissant un choix à l'agent, mais l'émission d'une disjonction entre deux obligations laisserait l'agent dans le désarroi face à la sémantique non-déterministe d'un tel réglement. Il paraît donc raisonnable, et cela est conforté par les exemples que nous avons vus dans la section 4.3.3, de nous limiter à une unique modalité déontique (éventuellement datée) située en partie droite de la formule RDLP.
- **Utilisation des opérateurs datés dans le conséquent :** Nous avons vu que les opérateurs déontiques datés que nous avions définis (obligation avec échéance et interdiction maintenue) étaient suffisamment puissantz pour traduires les normes en langage naturel que nous avions prises à titre d'exemple. Nous décidons de nous limiter au contexte de ces opérateurs datés pour mêler notions déontiques et temporelles (en partie droite des formules RDLP). Les huit critères que nous avons définis pour chacun deux (et notamment les principes de monotonie et de propagation) nous permettent une mise en œuvre du raisonnement plus aisée que s'il nous fallait considérer l'intégralité du produit des deux langages. Cette hypothèse est probablement l'une de celles qui limitent le plus l'expressivité des normes RDLP.
- **Pas de modalités déontiques appliquées aux prédicats d'état :** Les prédicats d'état sont en effet uniquement là pour mémoriser des faits et assurer leur pérennité dans le modèle logique. Ils ne désignent pas des actions dont un agent pourrait être responsable, mais sont plutôt le reflet de certaines de ses croyances. Les modalités déontiques ne sauraient donc être appliquées qu'à des prédicats performatifs ou informatifs, qui représentent des actions ou des actes de langage.

6.2.1.2 Syntaxe de RDLP

La grammaire (6.2) caractérise la syntaxe RDLP en fonction des symboles non-terminaux déjà définis, ainsi que d'un non-terminal DEADLINE représentant une proposition-date. Nous définissons également le non-terminal LDLP-ACTIVE, une version restreinte du langage $\mathcal{L}_{DLP}$ excluant les prédicats d'état. En résumé, une formule sous la forme RDLP est constituée d'un antécédent appartenant au fragment temporel de DLP, d'un symbole d'implication logique et d'une modalité déontique (ou de sa négation) appliquée à une formule LDLP-ACTIVE . Afin de clarifier la lecture (et l'analyse automatique) des normes RDLP, nous introduisons un nouveau symbole d'implication logique $\Rightarrow$ pour séparer antécédent et conséquent et fixer la forme des formules. Il aura néanmoins strictement la même sémantique qu'un opérateur $\rightarrow$ de la logique

propositionnelle appliquées à deux formules parenthésées.

$$\begin{aligned}\text{LDLP-ACTIVE} &= \text{INFORMATIVE} \mid \text{PERFORMATIVE} \\ &\mid \text{LDLP-ACTIVE } \text{“}\vee\text{”} \text{ LDLP-ACTIVE} \\ &\mid \text{“}\neg\text{”} \text{ LDLP-ACTIVE ;}\end{aligned} \tag{6.1}$$

$$\begin{aligned}\text{RDLP} &= \text{DLP-TEMP } \text{“}\Rightarrow\text{”} \text{ RDLP-DEON ;} \\ \text{RDLP-DEON} &= \text{“}Ob_a^\nu\text{”} \text{ LDLP-ACTIVE} \\ &\mid \text{“}Ob_a^\nu(\text{”} \text{ LDLP-ACTIVE “,” DEADLINE “)”} \\ &\mid \text{“}For_a^\nu(\text{”} \text{ LDLP-ACTIVE “,” DEADLINE “)”} \\ &\mid \text{“}\neg\text{”} \text{ RDLP-DEON ;}\end{aligned} \tag{6.2}$$

En conséquence, si φ est une formule DLP purement temporelle, si ψ est une formule $\mathcal{L}_{DLP}$ n'utilisant pas de prédicat d'état et si δ est une proposition-date, alors les formules suivantes sont présentées sous la forme RDLP (si l'on considère par extension que $\perp$ et $\top$ sont des formules de $\mathcal{L}_{DLP}$) :

$$\varphi \Rightarrow Ob_a^\nu \psi \tag{6.3}$$
$$\varphi \Rightarrow Per_a^\nu \psi \tag{6.4}$$
$$\varphi \Rightarrow For_a^\nu \psi \tag{6.5}$$
$$Ob_a^\nu \psi \tag{6.6}$$
$$\varphi \Rightarrow Ob_a^\nu(\psi, \delta) \tag{6.7}$$
$$For_a^\nu(\psi, \delta) \tag{6.8}$$
$$\varphi \Rightarrow \neg Ob_a^\nu(\psi, \delta) \tag{6.9}$$

Par contre, les formules suivantes n'appartiennent pas au langage RDLP. En effet, (6.10) utilise deux modalités déontiques, (6.11) également, de manière imbriquée (et avec deux autorités normatives distinctes), (6.12) place la partie déontique de la norme à gauche de l'implication logique, (6.13) applique une modalité temporelle à la modalité déontique, (6.14) utilise plusieurs modalités déontiques liées par une modalité temporelle.

$$Ob_a^\nu \psi_1 \rightarrow For_a^\nu \psi_2 \tag{6.10}$$
$$Ob_a^\nu(Ob_a^{\nu'} \psi) \tag{6.11}$$
$$Ob_a^\nu \psi \Rightarrow \varphi \tag{6.12}$$
$$\varphi \Rightarrow FOb_a^\nu \psi \tag{6.13}$$
$$Ob_a^\nu \psi_1 \ \mathcal{U} \ Per_a^\nu \psi_2 \tag{6.14}$$

La forme choisie pour la représentation des normes conditionnelles n'est pas la seule possible. Trois alternatives se trouvent dans la littérature et sont notamment exposées par Brian Chellas [Che80] :

- La forme $\varphi \rightarrow Ob\ \psi$, qui ressemble fortement à celle que nous avons choisie et qui a l'inconvénient de pouvoir être dérivée de n'importe quelle situation où $\neg\varphi$ est vraie, par exemple ;
- La forme $Ob\ (\varphi \rightarrow \psi)$, qui a l'inconvénient de devenir vraie dès que $Ob\ \neg\varphi$ ou $Ob\ \psi$ sont vraies ;

– La définition de l'obligation conditionnelle comme un opérateur primitif $Ob\ (\varphi/\psi)$, doté d'une sémantique propre restant à définir.

La solution que nous avons retenue s'apparente principalement à la première des trois options. Nous ne sommes pas inquiétés outre mesure par le désagrément qui lui est associé, les formules RDLP n'ayant pas vocation à être déduites par un moteur d'inférence. De plus, l'avénement de normes non applicables, s'il pose un problème conceptuel, ne gène pas le fonctionnement de l'agent puisqu'il n'a pas à les prendre en compte. La deuxième option aurait également pu être intéressante, mais elle fait également porter l'obligation sur l'antécédent, la condition d'applicabilité, alors que dans son essence la contrainte ne concerne que la formule de droite[2]. La dernière solution nous retirerait tout l'avantage d'avoir déjà à notre portée des notions déontiques. Nous en reprenons néanmoins l'esprit en isolant l'implication centrale $\Rightarrow$ de manière à figer l'expression et à éviter qu'une disjonction malheureuse puisse être confondue avec une formule RDLP.

6.2.2 Représentation textuelle des formules RDLP

Les formules RDLP sont représentées par des chaînes de caractères, envoyées d'un agent représentant une autorité normative à un agent PAw. Ces chaînes de caractères sont conçues pour être facilement interprétable par Prolog. La représentation de ces formules suit les quelques règles suivantes :

– L'implication est représentée par `->`, l'implication centrale de RDLP par `=>` ;
– La conjonction et la disjonction sont représentées par des opérateurs `and` et `or` infixés ;
– La négation est représentée par `neg` ;
– L'opérateur $p\ \mathcal{U}\ q$ est représenté de manière infixe et parenthésée : `(p until q)` ;
– L'opérateur $X^i p$ est noté sous la forme d'un foncteur `x(p, i)`, notation valable également pour X^0 représentant X ;
– Les autres opérateurs temporels unaires sont représentés par `f`, `p`, `g`, `h` ;
– L'autorité normative et l'agent n'apparaissent pas, l'information correspondante étant contenue dans le message encapsulant la formule ;
– Les opérateurs déontiques datés sont représentés par `ob(p, i)` et `for(p, i)`, où `p` est la formule sur laquelle s'applique le concept déontique et `i` est un entier strictement positif, représentant l'intervalle de temps entre le présent et l'échéance[3] ;
– Les modalités déontiques simples peuvent s'exprimer à l'aide des foncteurs unaires `ob`, `per`, `for` et `opt`[4] ;
– Les paramètres des prédicats de $\mathcal{L}_{DLP}$ sont représentés de la même manière que des atomes Prolog (nombres entiers, chaînes de caractères commençant par une minuscule, chaînes de caractères complexes entre guillemets simples) ;
– Les entiers représentant les échéances dans les normes datées ainsi que tous les paramètres des prédicats de $\mathcal{L}_{DLP}$ peuvent être remplacés par des schémas, qui se noteront à la manière de variables Prolog (c'est-à-dire par des chaînes de caractères commençant par une majuscule).

À titre d'exemple, nous proposons la représentation d'une formule RDLP tirée des normes de la section 4.3.3. L'échéance δ, déterminée par sa distance au présent $(7*24)$ y a été remplacée

[2] On notera que cette deuxième option est tout de même utilisable dans la syntaxe RDLP, avec des formules de la forme $\varphi \Rightarrow Ob\ (\psi_1 \rightarrow \psi_2)$.

[3] On remarquera qu'en ce qui concerne l'implémentation, contrairement au modèle formel, il est plus pratique de raisonner sur des durées. Ce choix permet en effet de s'appuyer sur des solveurs de contraintes numériques pour détecter les conflits, sans pour autant perdre les bonnes propriétés des propositions-dates (existence et unicité).

[4] Dans la pratique et pour l'intérêt de la généralisation, l'analyseur syntaxique interprétera également `ob(p, 0)` et `for(p, 0)` comme une obligation immédiate et une interdiction immédiate, respectivement.

par la valeur numérique de cette distance (168) et les retours à la ligne, non présents dans le format, ont été ajoutés pour la lisibilité :

$$\left.\begin{array}{l} actionType(\text{ProcessID}, \text{ActionType}) \\ H\neg informActionType(\texttt{self}, \text{Client}, \\ \quad \text{ProcessID}, \text{ActionType}) \\ owner(\text{ProcessID}, \text{DataID}, \text{Client}) \\ date(\delta) \wedge X^{7*24}\delta \end{array}\right\} \rightarrow For_{\texttt{self}}^{\texttt{wGroupReg}}(perform(\texttt{self}, \text{ProcessID}), \delta) \quad (6.15)$$

```
actionType(ProcessID, ActionType)
and h neg informActionType(self, Client, ProcessID, ActionType)
and owner(ProcessID, DataID, Client)
=> for(perform(self, ProcessID), 168)
```
(6.16)

6.2.3 Détection et arbitrage des conflits

La syntaxe des formules RDLP nous facilite la tâche en ce qui concerne la détection de conflits. Les modalités déontiques étant cantonnées dans le conséquent, il nous suffit de considérer la cohérence de l'ensemble des conséquents. Si l'on détecte un sous-ensemble conflictuel parmi ces conséquents, alors le conflit éventuellement présent dans l'ensemble de formules RDLP correspondantes pourra être détecté puis arbitré de manière assez fine en jouant sur les intersections des antécédents des formules qui le composent.

Nous nous plaçons ici dans le scénario d'ajout d'une nouvelle autorité normative.

6.2.3.1 Vérification de la cohérence de l'autorité normative

Lorsque l'agent PAw réceptionne un ensemble de normes d'une même autorité normative, il souhaite s'assurer que cet ensemble est acceptable. Nous avons vu dans la section 4.4.2 que tout conflit normatif (et pas seulement les conflits d'obligations) est alors à proscrire, ainsi que toute incohérence avec la représentation du monde de l'agent. Il y a donc ici une procédure de détection de la cohérence d'un ensemble, mais aucune étape d'arbitrage.

Le pseudo-algorithme 2 détaille la démarche de validation d'un ensemble de formules RDLP Δ, considéré en parallèle avec un ensemble de formules DLP non normatives W, représentant la connaissance du monde par l'agent. L'algorithme fait appel à la primitive de recherche d'un sous-ensemble conflictuel minimal déjà introduite, ainsi qu'à une fonction *consequents* qui extrait, à partir d'un ensemble de formules RDLP, l'ensemble des conséquents correspondants.

Si cette procédure de validation échoue, alors au sein de l'agent PAw l'identifiant de l'autorité normative (associée à sa prévalence) est retiré de la liste des autorités reconnues et placées dans une liste d'autorités inconsistantes. Un message d'alerte est envoyé à l'autorité, qui pourra à tout moment renvoyer un nouveau jeu de normes corrigé.

Les algorithmes mis en jeu pour la détection de l'incohérence étant très proches de ceux utilisés pour la détection des conflits, nous choisissons de détailler plutôt ces derniers par la suite, car ils sont légèrement plus complexes.

6.2.3.2 Préparation des normes à examiner

Lorsque les normes envoyées par une autorité ont été validées[5], il reste à les intégrer correctement à la base de normes. Cette étape consiste à détecter et à arbitrer tous les conflits

[5] L'agent se trouve alors dans l'état *RDLP reçu cohérent* de l'automate 6.1.

Algorithme 2 valider(Δ, E)

Précondition : Δ est un ensemble fini de formules RDLP $\{N_i = ant_i \Rightarrow cons_i\}_{0 \leq i \leq m}$
Précondition : W est un ensemble fini de formules DLP-TEMP $\{\varphi_i\}_{0 \leq i \leq n}$
  $C \leftarrow consequents(\Delta) \bigcup W$
  $C' \leftarrow C$
  pour tout $[cons_0 \ldots cons_p, \varphi_0 \ldots \varphi_q] \leftarrow ssensemble_conflictuel_minimal(C')$ **faire**
    si $\{ant_0 \ldots ant_p, \varphi_0 \ldots \varphi_q\} \nvdash_{\text{DLP}} \perp$ **alors**
      retourner $false$
    fin si
  fin pour
  retourner $true$

d'obligations pouvant survenir dans l'ensemble de normes, afin d'aboutir à une base de normes saine et cohérente (et à une base de normes désactivées).

Il convient tout d'abord d'isoler toutes les normes de nature permissive dans la base qui leur est dédiée, puisque celles-ci ne participent pas aux conflits d'obligations. On acquiert ensuite un verrou exclusif sur les bases des normes actives et désactivées. La procédure diffère ensuite légèrement suivant qu'il s'agisse d'une nouvelle autorité ou que l'autorité soit déjà connue de l'agent (avec un ensemble de normes associé qu'il nous faut remplacer).

Nouvelles autorités et autorités connues

Dans le premier cas, on peut opérer une détection « incrémentale » des conflits, en considérant l'union des normes actives et des nouvelles normes. En effet, puisque l'on désactive prioritairement les nouvelles normes et que l'apparition des conflits est un processus essentiellement monotone (l'introduction de nouvelles normes ne peut qu'augmenter le nombre de conflits), les nouvelles normes ne peuvent avoir d'influence sur les désactivations déjà actées.

Autorités connues

En revanche, si l'autorité est déjà connue, certaines de ses normes peuvent être présentes dans la liste des normes actives ou désactivées et avoir participé à des conflits déjà arbitrés. Ces normes devant être remplacées, l'ensemble des normes désactivées correspondant à des conflits dans lesquels une norme au moins de cette autorité est impliquée doit être considéré à nouveau (et donc extrait de la base des normes désactivées). L'implémentation actuelle du module de raisonnement normatif réexamine dans ce cas l'ensemble des normes de toutes les autorités normatives, reconstruisant ainsi entièrement la des normes désactivées.

6.2.3.3 Normes conditonnelles et conflits potentiels

La détection des conflits est fondée sur la dérivation d'une contradiction à partir d'un ensemble de normes. Si l'on prend l'exemple de deux normes amenant à un conflit d'obligations, alors l'une des deux va être complètement désactivée par le pseudo-algorithme 1 présenté dans la section 4.4.5.3. Si ces deux normes sont des normes conditionnelles, alors le conflit d'obligations (entre des conséquents inconsistants) ne survient que lors de la conjonction des deux antécédents. La plupart des antécédents de normes RDLP désignant des conditions purement contingentes, cette notion est finalement peu utile car elle pourrait nous amener à conserver des normes problématiques. C'est pourquoi nous détectons plutôt les *conflits d'obligations potentiels*,

ceux qui peuvent survenir s'il est possible que la conjonction des antécédents devienne vraie un jour.

C'est pour cette raison que nous détectons les conflits d'obligations uniquement sur les conséquents, c'est-à-dire uniquement sur des modalités normatives (datées ou non) isolées d'autres éléments logiques. Si l'on détecte un conflit d'obligations dans un ensemble de conséquents, alors on considère qu'il y a potentiellement un conflit entre les normes correspondantes. Si l'on prend l'exemple de deux normes $\varphi_1 \Rightarrow \psi_1$ et $\varphi_2 \Rightarrow \psi_2$ (où ψ_1 et ψ_2 sont des conséquents RDLP bien formés, c'est-à-dire constitués d'une modalité normative), on a la situation suivante :

$$\{\psi_1, \psi_2\} \vdash_{DLP+} \bot \tag{6.17}$$

On va alors artificiellement scinder les normes de la manière suivante, de manière à décrire les différents cas possibles :

$$\varphi_1 \wedge \neg\varphi_2 \Rightarrow \psi_1 \tag{6.18}$$
$$\varphi_2 \wedge \neg\varphi_1 \Rightarrow \psi_2 \tag{6.19}$$
$$\varphi_1 \wedge \varphi_2 \Rightarrow \psi_1 \tag{6.20}$$
$$\varphi_1 \wedge \varphi_2 \Rightarrow \psi_2 \tag{6.21}$$

Les normes (6.18) et (6.19) ne posent maintenant plus problème dans le cadre de cet ensemble de normes car leurs conditions d'application sont disjointes. Le problème est donc limité à l'ensemble des normes (6.20) et (6.21), qui ont les mêmes conditions d'application. Dans ce cas, si l'on peut être assuré que la conjonction est inatteignable (si elle permet de dériver $\bot$), alors il n'y a pas de conflit potentiel. Dans le cas contraire, on considère qu'il y a un conflit potentiel et l'on désactive celle des deux normes qui correspond à l'autorité normative de plus faible prévalence.

Le même principe est appliqué pour les ensembles de cardinalité n, qui résultent en n normes potentiellement problématiques et n normes ne posant pas de problème (présentant des antécédents modifiés du type $\varphi_1 \wedge (\neg\varphi_2 \vee \ldots \vee \neg\varphi_n)$). De cette manière, on exploite le concept de conflit d'obligations (potentiel) de manière beaucoup plus précise, en considérant l'ensemble des normes qui pourraient poser problème mais en évitant toutefois de sacrifier des normes qui pourraient tout de même être appliquées dans certains cas[6].

6.2.3.4 Dérivation dans les ensembles de normes

Afin de concevoir la primitive *ssensemble_conflictuel_minimal* que nous avons utilisée dans les pseudo-algorithmes déjà présentés et qui constitue la pierre d'angle de la détection des conflits, il nous faut maintenant mettre au point les outils nous permettant d'obtenir des ensembles de normes fermés pour la dérivation logique, au sein desquels il nous sera facile de détecter des incohérences. Nous travaillerons ici dans un module dynamique Prolog[7] dans lequel l'ensemble des conséquents à examiner sera ajouté à l'aide d'assertions.

[6] Le procédé perd toutefois en efficacité lorsque des normes participent à plusieurs ensembles conflictuels. Certaines normes désactivées au début de la procédure pourraient en fait être partiellement réactivées (avec un antécédent limité) si une des normes avec lesquelles elles étaient en compétition se voyait elle-même partiellement désactivée lors d'un arbitrage ultérieur. L'implémentation actuelle ne met pas en œuvre cette amélioration.

[7] L'utilisation de l'interface JPL nous impose l'unicité de la machine virtuelle SWI-Prolog associée à un portail. En conséquence, il nous faut pour chaque procédure de détection de conflits travailler dans un module séparé. Le nom de ce module est constitué d'un préfixe propre à l'agent (fourni par le portail à la construction de l'agent) et d'un suffixe numérique issu d'une incrémentation assurée par l'agent.

La procédure commence par une phase d'initialisation du module (algorithme 3) visant à préparer le module avec les informations concernant les autorités normatives, les théorèmes du monde et les formules considérées par l'agent comme des théorèmes sur la représentation du monde.

Algorithme 3 initialiser(ListeAutorités, Module)

Précondition : ListeAutorités est la liste des noms de toutes les autorités du contexte
Précondition : Module est un nouveau nom de module Prolog unique
Postcondition : Le module Module est prêt à recevoir un ensemble de normes à arbitrer
Nettoyage du module (suppression d'éventuelle assertions présentes)
Chargement des théorèmes connus (assertion de prédicats *norm/3* pour toutes les autorités de la liste)
Chargement des formules du monde (assertion de prédicats *norm/3* pour toutes les autorités de la liste)
`expand(Module)`

Les normes à considérer sont ensuite introduites sous la forme de faits du type `norm(Authority, NormID, Norm)` où le premier argument est l'identifiant de l'autorité, le deuxième l'identifiant de la norme et le troisième la formule en elle-même (le conséquent de la norme RDLP). À partir de ces normes de départ, le système va déduire un certain nombre de faits du type `dnorm(NormIDList, Norm)` où le premier argument est cette fois une liste de couples `[Authority, NormID]` identifiant les normes initiales ayant été nécessaires à la déduction de `Norm`. Ainsi, lorsque l'inconsistance d'une autorité unique ou un conflit entre plusieurs autorités est détecté, on dispose des listes d'identifiants de toutes les normes initiales ayant contribué à cette inconsistance ou à ce conflit.

On notera que la déduction, matérialisée par un prédicat `deduce/2`, ne permet pas d'atteindre toutes les formules déductibles, mais seulement les formes qui nous seront utiles à la détection des inconsistances. Ainsi, les prédicats de simplification utilisés pendant l'inférence chercheront à obtenir prioritairement des symboles $\perp$, des obligations (plutôt que des interdictions) et des négations d'obligations (plutôt que des permissions ou des options). De plus, on évite bien évidemment l'introduction de disjonctions inutiles. Enfin, afin de permettre un traitement homogène de tous les concepts déontiques, on considérera uniquement des normes datées, les modalités déontiques immédiates (`ob(...)`, `for(...)`) étant transformées en normes datées avec une échéance à la date présente (`ob(..., 0)`, `for(..., 0)`).

L'algorithme 4 présente le prédicat `expand/1`, appelé après l'ajout de faits `norm/3` pour dériver tous les faits `dnorm/2` possibles. Le prédicat commence par ajouter les faits `dnorm/2` correspondant aux normes `norm/3` existantes, puis appelle le prédicat `doexpand/1` (algorithme 5) pour étendre les ensembles de normes de manière récursive. Ce dernier prédicat ajoute toutes les normes qui peuvent être déduites à partir d'une ou plusieurs normes enregistrées. Il fait appel pour cela au prédicat `deduce/2` déjà mentionné, qui modélise les déductions possibles à partir des axiomes et règles d'inférence de la logique propositionnelle et de la logique déontique standard, ainsi que des propriétés des opérateurs déontiques datés.

6.2.3.5 Traitement des normes datées

Il est intéressant ici de décrire plus précisément la manière dont les normes datées sont traitées pendant la phase de déduction logique. Nous avons vu que les échéances étaient représentées dans ces formules par un entier figurant un délai en jours. Afin de généraliser cette représentation

Algorithme 4 expand(Module)

Précondition : `Module` est un module contenant des prédicats dynamiques `norm/3`

Postcondition : `Module` contient l'ensemble de prédicats dynamiques `dnorm/2` minimal contenant les prédicats `norm/2` existants et fermé sous la procédure de déduction `deduce/2`.

```
pour tout Module:norm(Authority, NormID, Norm) faire
  SimpNorm ← simplifier(Norm)
  si \+ Module:dnorm([[Authority, NormID]], SimpNorm) alors
    assert(Module:dnorm([[Authority, NormID]], SimpNorm)
  fin si
fin pour
pour tout Authority faire
  doexpand(Module)
fin pour
```

lorsque le délai est connu de manière plus faible, ce dernier est représenté par le domaine d'une variable contrainte. Ainsi, une obligation avec échéance de type `ob(N, 5..150)` représente en fait n'importe quelle obligation (sur la même formule `N`) avec une échéance comprise entre 5 et 150 jours[8].

Ce type de normes datées est ajouté lors de la phase de déduction par le prédicat `deduce/2`, qui fait alors notamment usage des principes de monotonie et de propagation introduits dans la section 4.3.2.3. Ainsi, par le principe de monotonie, une norme `ob(N, 5)` permet la déduction d'une norme `ob(N, Dom5)`, ou `Dom5` représente le domaine des variables supérieures à 5. Par l'application du principe de propagation, une norme `for(N, 5)` permet de son côté la déduction d'une norme `for(N, Dom05)`, ou `Dom05` représente le domaine des variables comprises entre 0 et 5. Par la suite, si le domaine de l'échéance d'une obligation est entièrement inclus dans le domaine d'une obligation maintenue sur la même formule, alors il est aisé de détecter que les deux formules sont incompatibles.

6.2.3.6 Détection des conflits et contradictions

Une fois la phase de déduction terminée, on dispose pour chaque autorité d'un ensemble de formules aussi simples que possibles, dérivées des normes introduites, des théorèmes et des formules sur la représentation du monde. Il ne reste alors qu'à examiner les potentielles contradictions entre ces formules.

Si l'on se trouve dans le cas de la vérification de cohérence d'une nouvelle autorité normative, alors le prédicat `inconsistent/2` cherche à détecter, dans l'espace de normes de cette autorité, la présence de $\perp$, d'un couple $\{\varphi, \neg\varphi\}$ ou d'un couple de formules déontiques incompatibles[9].

En revanche, si l'on examine un ensemble de normes complet, alors le prédicat `conflict/2` effectue la même opération de recherche, mais en passant de la logique SDL à SDL+, c'est-à-dire en ignorant le cloisonnement par autorité normative dans le cas des modalités déontiques. Une étape de raisonnement intermédiaire (permettant de dériver artificiellement de nouvelles normes à partir de formules issues de différentes autorités normatives) permet de détecter des conflits situés plus en profondeur.

[8] Les domaines des variables d'échéances doivent alors être inclus dans un domaine de référence, par exemple `0..1000000`, qui fixe alors l'horizon temporel du module de raisonnement logique.

[9] Dans le cas d'une nouvelle autorité, il faut également noter que le moteur de déduction est légèrement modifié pour permettre la présence de normes permissives, qui ne doivent alors participer à aucun conflit sous peine de rentre l'autorité normative inconsistante aux yeux de l'agent PAw.

Algorithme 5 doexpand(Module)

Précondition : `Module` est un module contenant des prédicats dynamiques `dnorm/2`

Postcondition : L'ensemble de prédicats dynamiques `dnorm/2` est l'ensemble minimal contenant les prédicats `dnorm/2` préexistants pour cette autorité et fermé sous la procédure de déduction `deduce/2`.

```
findall([NormIDList, Norm], Module:dnorm(NormIDList, Norm), NormList)
(* NormList est une liste de couples [NormIDList, NormID] où NormIDList est une liste de
couples [Authority, NormID] représentant les normes ayant permis de déduire Norm *)
pour tout DeduceNorms tel que deduce(NormList, DeducedNorms) faire
  (* DeducedNorms est une liste de nouvelles normes déduites de NormList et représentées
  dans le même format *)
  addDeducedNorms(DeducedNorms, Module)
  (* Les normes déduites sont ajoutées après simplification, seulement si elles ne sont pas déjà
  présentes *)
fin pour
findall([NormIDList, Norm], Module:dnorm(NormIDList, Norm), NewNormList)
(* On vérifie si de nouvelles normes ont été ajoutées, ou si l'ensemble est stable (condition
d'arrêt *)
si NormList \= NewNormList alors
  doexpand(Module)
fin si
```

6.2.3.7 Arbitrage des conflits d'obligations

Lorsqu'une inconsistance ou un conflit survient, la (ou les) formule(s) constituant l'incohérence étant attachée(s) aux listes des identifiants des normes initiales nécessaires à leur production, les sous-ensembles conflictuels sont récupérables immédiatement. L'ensemble de ces données est alors transmis de Prolog à JACK, qui dispose alors de toutes les informations pour procéder à l'arbitrage.

Pour chaque conflit d'obligations, les normes entrant en jeu sont scindées par leurs antécédents comme décrit dans la section 6.2.3.3, la norme problématique correspondant à l'autorité normative de moindre prévalence est désactivée[10].

JACK est alors en mesure de publier une base de normes actives, une base de normes désactivées et une base de normes permissives (disponible si l'agent en a l'utilité, ce qui n'est pas notre cas ici).

6.3 Mise en œuvre de la protection des données

L'agent PAw dispose maintenant d'un ensemble de normes cohérent. Il nous faut maintenant mettre en œuvre les mécanismes qui nous permettront de décider lorsqu'une norme deviendra active, puis d'appliquer cette norme lors de la manipulation de données personnelles. Enfin, nous donnerons à l'agent PAw une conscience minimale des problématiques de protection étendue, utile lorsque des données devront être partagées au sein d'une transaction impliquant d'autres agents.

[10] Cette procédure pourrait être améliorée par un raisonnement plus poussé sur les expressions temporelles constituant les antécédents.

6.3.1 Activation des normes

Au cours de son fonctionnement, l'agent PAw renseigne plusieurs *BeliefSets* mémorisant les divers états, actions et actes de langages référencés par les prédicats de $\mathcal{L}_{DLP}$. Ainsi, à tout moment, il est capable d'évaluer si telle ou telle action a été effectuée dans un passé plus ou moins proche ou si tel état est vérifié.

En parallèle, le plan *watchNorms* de l'agent s'exécute en tâche de fond, examinant à intervalles réguliers l'ensemble des normes actives et en particulier leurs conditions d'application[11]. Pour chaque norme, le plan détermine si ses conditions sont remplies. Si c'est le cas, le conséquent de la norme est introduit dans une nouvelle base de normes « activées », qui comprendra donc des interdictions et des obligations, possiblement datées. Dans cette nouvelle base, les normes sont répertoriées avec une référence sur la norme complète d'origine, ainsi que sur les éléments de croyance correspondant à la condition d'application. L'entrée dans la base des normes activées comporte également une date d'entrée en vigueur et une date d'expiration[12] qui permettent au besoin de raisonner sur des normes ayant été actives par le passé. Ce sont les obligations et interdictions présentes dans cette base que l'agent doit respecter à un instant donné.

Nous avons également vu dans la section 4.3.3 que les normes pouvaient être un outil de planification. L'agent PAw, dans sa version actuelle, ne dispose que de fonctionnalités restreintes en la matière. Lorsque ses objectifs le poussent à effectuer certaine action, il observe la base des normes actives et en extrait celles qui concernent l'action qu'il doit mener. Via un éventuel passage par Prolog, il peut alors déterminer les intervalles de temps pendant lesquels cette action est autorisée ou interdite ou encore se rendre compte qu'il ne lui est pas possible de planifier cette action dans l'intervalle qui l'intéresse. Dans ce dernier cas, l'agent doit aller rechercher l'antécédent de la norme complète afin de déterminer si une action intermédiaire (par exemple, une étape d'information d'un agent tiers avant l'exécution d'un traitement) pourrait l'invalider, désactiver l'interdiction et lui permettre d'atteindre ses objectifs.

6.3.2 Protection locale : procédures de manipulation des croyances

Le modèle de protection locale accepte l'hypothèse suivant laquelle l'agent PAw est capable d'identifier les données qu'il manipule comme relevant ou non d'un caractère personnel. Lorsque c'est le cas, ces données sont isolées des autres dans une base de croyances *PersonalData*. Chaque information y est référencée avec la mention de son propriétaire et de son DATATYPE. Les informations des différentes bases relatives aux prédicats de $\mathcal{L}_{DLP}$ contiennent des réféfences sur ces données. Dans cette base de croyances, il est important de noter que la donnée elle-même est stockée avec ses méta-informations (à la différence du langage DLP).

La base *PersonalData* est dotée de méthodes de rappel (une fonctionnalité native de JACK) permettant d'appeler un plan *accessCheck* dès lors qu'un plan « métier » de l'agent cherche à accéder à une des données de la base. Cette procédure va, avant de délivrer ou de refuser l'information, vérifier l'identité du plan (qui doit être identifié au sein de l'agent comme un plan compatible avec les normes RDLP) ainsi que les normes activées qui s'appliquent à l'action

[11] La base de normes répertorie ces dernières sous une forme qui permet d'accéder séparément aux différents éléments de la formule RDLP, de manière que JACK puisse avoir une connaissance minimale de la sémantique de ces normes, même si leur interprétation plus poussée est confiée à Prolog.

[12] Cette date d'expiration est à l'origine l'échéance d'une norme datée, mais elle peut être ramenée à la date précédant immédiatement l'instant présent si le plan *watchNorms* détecte que l'antécédent n'est plus valide. C'est ainsi par exemple qu'une interdiction maintenue de procéder à un traitement peut être levée une fois qu'il est avéré que l'agent a satisfait à des obligations préliminaires. La date d'expiration peut également être repoussée, lorsqu'à une activation ultérieure du plan, les mêmes éléments et la même norme conduisent à une interdiction ou obligation similaire avec une échéance décalée d'un pas de temps.

souhaitée par ce plan (via une phase d'unification entre les normes activées et les méta-données dont on dispose).

Il est important de se rendre compte ici que dans cette phase de protection locale, les données et leur traitement sont uniquement situés à l'intérieur de l'agent, les transferts à des agents tiers seront soumis à d'autres procédures. En local, l'agent se fait en quelque sorte confiance à lui-même, il considère que les déclarations d'intention entre ses différents plans sont sincères. Ces méthodes de protection locale sont ainsi mises en échec dans le cas où l'agent PAw est corrompu et que certains de ses plans métier ont été modifiés pour effectuer, en interne, des déclarations falsifiées (affirmant par exemple effectuer un type de traitement différent de celui qui s'opère en réalité).

6.3.3 Éléments de protection étendue

Maintenant que l'agent PAw dispose des outils pour assurer la protection locale des données (dans l'enceinte de ses propres plans d'exécution), il convient de lui fournir un moyen de raisonner sur leur protection étendue, afin d'agir au mieux lorsque les circonstances exigeront que ces données soient traitées par un tiers.

6.3.3.1 Caractérisation des protocoles

Dans le cadre de notre démonstrateur, les mécanismes et architecture issus du *Trusted Computing* ne sont que simulés. En effet, notre implémentation du modèle PAw est ici purement logicielle. La mise en œuvre des différents protocoles applicatifs repose donc sur des *plans* particuliers de l'agent, interfaçant ses interactions avec les autres agents du système.

Le développement de la plate-forme d'accueil utilisée dans l'architecture applicative que nous avons proposée nécessiterait en effet des moyens qui dépassent le cadre de ce travail. Nous avons donc décidé de fournir aux agents une connaissance purement théorique de quelques familles d'architectures applicatives. L'objectif est que l'agent dispose d'une bibliothèque d'architectures parmi lesquelles choisir, en fonction des propriétés qu'il souhaite voir respecter par l'application distribuée.

La représentation des diverses architectures se fait par le biais d'un *BeliefSet* dédié, associant les informations suivantes :

- Le nom de l'architecture ;
- Les options choisies si l'architecture en propose (choix d'un protocole d'attestation par exemple) ;
- Le niveau de confiance associé (se référant à la classification que nous avons proposée dans la section 5.1.2) ;
- Les caractéristiques en termes de nécessité d'utilisation d'un TPM et de localisation de ce dernier ;
- Les caractéristiques en termes de nécessité de certification de code (code public ou privé, spécifique ou générique).

L'agent dispose également d'une base répertoriant les tiers de confiance connus, associés aux protocoles et options qu'ils supportent.

Comme nous l'avons vu, les caractéristiques des protocoles applicatifs n'ont pas de lien direct avec le respect ou le non-respect des normes liées à tel ou tel axe réglementaire de la protection des données personnelles. Elles conditionnent plutôt la manière dont l'ensemble de ces normes sont appliquées à l'échelle d'une application distribuée. Pour cette raison, les procédures de décision de l'agent quant aux protocoles applicatifs utilisent un fichier de configuration, établi

par l'utilisateur humain, attachant des préférences (en termes de niveau de confiance ou de caractéristiques spécifiques) à des DATATYPE, des ACTIONTYPE ou à des couples (DATATYPE, ACTIONTYPE). Chaque association étant attachée à une valeur, l'ensemble des règles applicables à une transaction donnée permet de calculer une utilité relative pour les différents types d'architectures répertoriés. De même, le fichier de configuration spécifie des valeurs de confiance a priori sur les tiers de confiance reconnus par l'utilisateur. Ces valeurs de confiance sont fournies de manière statique dans notre démonstrateur, mais seraient amenées, dans une application plus complexe, à évoluer au cours de l'expérience de l'agent, influencées par ses interactions, ses observations et les recommandations reçues. De nombreux modèles et implémentations de gestion de la confiance sont disponibles dans la littérature et pourraient être utilisés ici (on peut citer par exemple les modèles développés par Cristiano Castelfranchi et Rino Falcone [CF98], Robert Demolombe [Dem04] ou encore Tyron Grandison et Morris Sloman [GS02]).

Ce système de préférences dépendant grandement de la connaissance des architectures et des problèmes sous-jacents qu'a le rédacteur des règles, il reste difficilement personnalisable. En effet, les informations relatives aux différentes architectures et leur association aux méta-données des informations personnelles ne peuvent être raisonnablement fiables que si elles sont partagées par une communauté, régulièrement vérifiées et tenues à jour. Ce modèle n'a donc sans doute qu'une vocation de démonstration ou de prototype, destiné à être remplacé, dans une application complète, par un module décisionnel plus élaboré.

6.3.3.2 Outils primitifs de négociation

Afin d'illustrer l'intérêt du raisonnement normatif de l'agent et de sa représentation des caractéristiques des protocoles applicatifs, des fonctionnalités basiques de négociation ont été mises en œuvre.

On suppose tout d'abord que les agents se sont préalablement entendus sur la sémantique du service ou de la transaction. Les divers agents sont donc en mesure, chacun pour sa part, de déterminer les traitements à effectuer pour mettre en œuvre cette transaction. Pour chacun de ces traitements, l'agent responsable dudit traitement entame une négociation séparée avec chacun des interlocuteurs dont il requiert les données, afin de s'accorder sur le ou les protocoles applicatifs à suivre. On est donc dans le cas de négociations un-à-un (entre ce que nous appellerons un agent de service et un agent utilisateur), orientées tâche, comme décrites notamment par Michael Wooldridge [Woo01b, section 7.3.1]. On utilise alors un protocole de négociation simpliste utilisant les primitives suivantes :

- **Proposition :** l'agent de service associe un ensemble de prédicats informatifs et de prédicats *request* (correctement instanciés) à une spécification de protocole et éventuellement à un jeu d'options et un ou plusieurs tiers de confiance impliqués dans le protocole.
- **Contre-proposition :** si l'agent utilisateur estime qu'un autre protocole, d'autres options ou d'autres tiers peuvent constituer une solution plus appropriée au vu des exigences techniques exprimées, alors cette contre-proposition est envoyée, sous la même forme que la proposition initiale, à l'exception des prédicats $\mathcal{L}_{DLP}$ qui sont supposés acquis pour la durée de la négociation. L'agent de service est lui-même libre de relancer par une nouvelle contre-proposition. Un agent ne peut, au cours d'une même négociation, envoyer deux propositions ou contre-propositions aux contenus identiques.
- **Accord :** si l'un des deux agents estime la proposition ou la contre-proposition acceptable, alors l'expression de son accord clôt la négociation.
- **Échec utilisateur :** si l'agent utilisateur ne considère pas la proposition comme acceptable

mais qu'il ne peut en proposer de meilleure, alors il exprime cette impossibilité. Cette primitive clôt la négociation.

– **Échec service :** si l'agent de service considère que la contre-proposition de l'utilisateur est inacceptable pour lui et qu'il ne peut en proposer de nouvelle, alors il exprime cette impossibilité. Cette primitive, en elle-même, ne clôt pas la négociation : l'agent utilisateur peut relancer par une nouvelle contre-proposition ou bien exprimer à son tour un échec.

La négociation peut se terminer soit par un accord d'un des deux agents, soit par un échec de l'agent utilisateur, auquel cas la transaction dans son ensemble est irréalisable.

La terminaison de ce protocole de négociation est assurée par le nombre fini des combinaisons de protocoles, d'options et de tiers de confiance connus par chacun des deux agents pour un ensemble de prédicats donnés. Ce nombre est une borne supérieure du nombre d'échanges de contre-propositions (puisqu'un même agent ne peut effectuer deux contre-propositions identiques dans la même négociation).

Les stratégies de négociation sont propres à chaque agent. Dans la mise en œuvre actuelle de l'agent PAw, ce dernier calcule, en fonction des prédicats $\mathcal{L}_{DLP}$ de la proposition, l'ensemble des solutions techniques possibles, avec une utilité associée calculée sur la base d'une moyenne des valeurs de préférences exprimées dans le fichier de configuration cité dans la section précédente. Cette valeur peut être positive ou négative, suivant que les préférences décrivent la solution technique comme adaptée ou dangereuse. L'agent utilisateur tire ensuite ses contre-propositions de cet ensemble, en les choisissant par valeur d'utilité décroissante et en s'arrêtant lorsqu'un certain seuil (zéro dans notre démonstrateur) est atteint, signifiant que les solutions techniques situées en-deça ne sont pas adaptées. L'agent est également libre de fixer le seuil (en termes d'utilité) au-delà duquel il acceptera une proposition de l'agent de service. Encore une fois ce seuil est à zéro dans notre démonstrateur, où les valeurs positives désignent des solutions acceptables.

Il est important de noter que dans ce protocole, les prédicats $\mathcal{L}_{DLP}$ décrivant le traitement sont considérés comme stables tout au long de la négociation. Pour un même service fourni, l'agent de service a cependant parfois la possibilité de modifier ces prédicats pour demander des données différentes ou modifier une caractéristique du traitement. On considère ici que cela constitue une proposition indépendante, initiant une nouvelle négociation. L'accord sur ces prédicats, décrivant le traitement, pourrait également être atteint par un mécanisme de négociation similaire (avec un calcul d'utilité utilisant les prévalences normatives des normes violées par la proposition), mais dans l'état actuel du démonstrateur une telle négociation n'est pas mise en œuvre.

6.4 Discussion

La mise en œuvre de l'agent PAw que nous proposons ici n'a qu'une valeur de démonstration. Notre prototype est conçu pour mettre en évidence les possibilités apportées, en termes de fonctionnalités pratiques, par nos développements théoriques. Elle reste limitée notamment en ce qui concerne les capacités de déduction logique de l'agent et la gestion effective des différentes architectures d'application (notamment la gestion du matériel dans le cadre de l'informatique de confiance). L'intégration dans une application destinée à un usage réel et immédiat nécessiterait des ajustements plus ou moins importants suivant les modules.

Certaines fonctionnalités qui nous paraissent intéressantes au vu de la mission de l'agent PAw ne sont pas présentes dans la version actuelle mais pourraient sans doute être étudiées avec bénéfice. En particulier, nous ne fournissons pas à l'agent PAw les outils qui lui permettraient

de raisonner sur le contexte normatif d'autres agents, inférant ainsi les normes auxquelles ils sont soumis et permettant d'évaluer plus finement les risques encourus lors de la transmission de données personnelles. De même, nous n'avons pas doté l'agent PAw des interfaces qui lui permettraient d'être un client effectif dans les différentes architectures applicatives que nous représentons. Le développement d'une plate-forme d'accueil telle que présentée dans la section 5.3, par exemple, nécessiterait des ressources conséquentes que nous n'avons pas déployées ici. L'agent PAw ne dispose donc pas des capacités de projection et de mobilité qui lui permettrait d'en profiter. Au sein d'un agent JACK comme celui que nous proposons, toutes ces fonctionnalités sont appelées à être publiées sous la forme de *capabilities* indépendantes mais interopérables.

En ce qui concerne la couche de raisonnement normatif, nous avons vu que l'implémentation proposée consiste en une restriction du langage DLP, qui apporte des limitations formelles aussi bien que des avantages au niveau de la réalisation. Si les avantages pratiques nous sont apparus clairement (notamment en ce qui concerne la mise en œuvre de la gestion des conflits d'obligations, fondée sur les simplifications syntaxiques de RDLP), les limitations associées (en termes d'expressivité par exemple) doivent maintenant être clairement établies.

Chapitre 7

Évaluation

Nous nous proposons maintenant de fournir des éléments d'évaluation des fonctionnalités développées dans l'agent PAw. Il est tout d'abord nécessaire de valider la pertinence des modèles utilisés. La mise en œuvre de la logique DLP à travers RDLP est en particulier digne d'attention à cause des restrictions et les limitations qu'elle pose. Nous proposons ensuite trois familles de scénarios qui nous permettront de mettre en évidence les capacités du modèle PAw à s'adapter à des environnements différents. Ces scénarios seront également l'occasion d'évaluer la pertinence des alternatives offertes à l'agent (en termes d'architectures ou de préférences normatives).

7.1 Validation du modèle

Cette première étape du travail d'évaluation a pour but d'estimer dans quelle mesure les choix que nous avons faits sont adaptés au problème que nous traitons.

7.1.1 Validation du modèle logique

Comme toute implémentation, la mise en œuvre de la logique DLP dans l'agent PAw introduit des limitations qu'il nous faut identifier. Nous nous intéresserons tout d'abord au biais introduit par l'approximation de l'inférence modale en Prolog, puis aux diverses restrictions introduites par rapport au langage DLP par l'utilisation de la syntaxe RDLP.

7.1.1.1 Limitations dues à l'implémentation en Prolog

L'utilisation de logiques modales normales en Prolog va souvent de pair avec une limitation du mécanisme d'inférence. Dans le cadre de notre implémentation, ces limitations peuvent facilement être observées sur le fragment déontique de DLP. En effet, en logique déontique standard, la règle de nécessitation et l'axiome (Ob-K) signifient que toute tautologie, tout théorème est obligatoire. Dans le cas de l'agent PAw, cette règle s'applique à quelques tautologies de non-contradiction intrinsèques à la logique propositionnelle, ainsi qu'à des éléments de connaissance du monde fournis à l'agent comme étant des théorèmes. Cette connaissance du monde étant nécessairement limitée, l'agent n'est pas forcément conscient de tous les théorèmes de son environnement d'exécution et n'est par conséquent pas capable d'en dériver toutes les obligations intrinsèques.

D'autre part, lors des phases de déduction utiles à la détection des conflits, toutes les formules atteignables ne sont pas produites, pour la bonne raison qu'elles sont en nombre infini. Nous n'avons pas choisi ici une inférence à profondeur bornée, qui développerait un arbre de

formules tronqué au-delà d'un certain horizon, mais une inférence guidée, ne menant que vers certains types de formules. C'est ainsi que le nombre de modalités des formules atteintes ne croît pas indéfiniment, que les modalités permissives ne sont jamais déduites ou que les disjonctions arbitraires ne sont pas atteignables. Ces choix effectués dans le modèle de déduction sont orientés par la recherche des conflits entre les formules déduites. Ces conflits étant le fruit de contradictions observables, il est plus judicieux de réduire la diversité des modalités et de s'orienter vers des formules les plus restrictives possibles.

7.1.1.2 Restrictions dues à l'utilisation de RDLP

À cause des hypothèses que nous avons posées et des choix que nous avons faits en utilisant la syntaxe RDLP pour l'émission des normes par les autorités, certains types de normes relativement complexes ne sont pas représentables. Il convient de les identifier afin de cerner les limitations de l'outil.

Conditions sur des concepts déontiques

La principale restriction est l'impossibilité qu'il nous est faite d'écrire des normes conditionnées par d'autres normes (si nous considérons comme n'étant pas des normes les formules sans concept déontique). En effet en RDLP il n'y a qu'une modalité déontique par formule et elle est dans le conséquent, non dans la condition. D'une manière encore plus générale, il est impossible à une autorité normative de faire référence à d'autres autorités. Il est donc impossible de représenter en RDLP des réglementations fondées sur d'autres réglementations, comme par exemple une loi nationale qui prévoirait qu'un arrêté local puisse l'invalider.

Normes *Contrary-To-Duties*

Il est également délicat de représenter de manière explicite des normes dites *Contrary-To-Duty* (ou CTD), c'est-à-dire prenant pour condition la violation d'une autre norme. Ce type de norme étant néanmoins très utile (bien qu'à l'origine de problèmes théoriques difficiles à contourner[1]), il existe un moyen de les représenter de manière détournée. Supposons une norme CTD du type $(Ob\ \varphi \wedge \neg\ \varphi) \rightarrow Ob\ \psi$. On ne peut pas la représenter telle quelle en RDLP, mais si l'on connait la condition d'application de la première obligation (φ_0 par exemple), on peut utiliser la formulation suivante : $(\varphi_0 \wedge \neg\ \varphi) \rightarrow Ob\ \psi$. Celle-ci est a priori représentable sans problème en RDLP, et son seul inconvénient est qu'elle ne fait pas explicitement référence à la première norme. Cela se traduira par le fait que si la norme aboutissant à $Ob\ \varphi$ est désactivée à la suite d'un arbitrage de conflits, alors $Ob\ \psi$ pourrait être indûment déduite. Le caractère pertinent ou non de cette dernière dérivation dépend entièrement de la sémantique même que l'on a voulu donner à la norme CTD initiale : était-ce un pis-aller, une réparation, une sanction ? Les interactions entre l'arbitrage de conflits et les normes CTD constituent une problématique complexe qu'il serait très intéressant d'approfondir, mais qui ne rentre pas dans le cadre de nos travaux actuels.

Absence de modalité déontique

Il est également impossible d'utiliser des formules exemptes de toute modalité déontique. Cela fait tout-à-fait sens dans le cadre de l'utilisation de RDLP : on interdit en effet ainsi aux autorités normatives d'édicter des formules qui pourraient être considérées commes des

[1] Le plus important de ces problèmes est sans doute le paraxoxe de Chisholm, décrit dans l'annexe A.

affirmations sur la connaissance du monde, échappant aux mécanismes d'arbitrage des conflits d'obligations. De telles règles absolues peuvent toutefois exister dans un fichier de configuration de l'agent, pour formaliser justement les théorèmes du monde en cas de besoin particulier. La pérennité des prédicats d'état est assurée par ce biais. La conception de ces formules doit être menée avec soin, car les ensembles de normes entrant en conflit avec elles seront considérées comme fortement inconsistantes, et l'autorité normative correspondante rejetée (car se fondant sur une connaissance du monde incompatible avec celle de l'agent).

Interactions entre modalités temporelles et déontiques

Enfin, RDLP limite fortement l'interaction entre modalités temporelles et déontiques. Celles-ci sont réduites à l'utilisation des normes datées que nous avons définies. Ces dernières sont raisonnablement expressives, en ceci que les deux notions d'obligations avec échéances et d'interdictions maintenues permettent de représenter directement la plupart des contraintes réglementaires concernant les données personnelles. Cependant il reste certaines formules qui demeurent inatteignables. La notion de précédence entre deux actions est sans doute la plus saillante. Une autorité normative ne peut pas imposer qu'une action ait lieu avant une autre : il faudrait pour cela introduire des notions temporelles (autre que les échéances) à l'intérieur d'une modalité déontique. Suivant l'intention qui a conduit l'autorité à émettre ce type de réglementation, il est toutefois possible d'exprimer celle-ci de manière imparfaite à l'aide de formules RDLP, une action pouvant être la condition de déclenchement de l'obligation d'en effectuer une autre dans le futur, ou de l'interdiction de cette action au contraire (manière de représenter le fait que la précédence ne peut plus être respectée).

7.1.2 Validation du modèle de gestion des croyances

Le modèle de représentation que nous proposons pour les croyances portant sur les données personnelles, associé aux mécanismes JACK mis en œuvre, permet d'assurer une protection locale de ces informations. Ici, la localité est limité à l'enceinte logicielle de l'agent : tout plan JACK qui souhaite accéder à des données personnelles doit être déclaré comme ayant un comportement compatible avec les normes. Le contrôle se faisant à un niveau purement déclaratif (on ne se fonde pas sur une analyse statique du comportement du plan [WD01], par exemple), cette protection ne reste valable que si les déclarations en question sont correctes. Les plans métier (implémentation en JACK des intentions métier définies dans la section 3.3.2.4) ne faisant pas partie du paquetage PAw, ces déclarations sont à la charge des développeurs d'applications. On observe effectivement lors des tests que lorsque les déclarations des plans métier sont falsifiées, les données peuvent être utilisées en violation de l'ensemble de normes actif. Le bon fonctionnement de la protection locale des données repose donc sur une confiance dans le code métier, qui pourra éventuellement être sujet à certification.

Il est également bon de remarquer que la gestion des croyances à caractère personnel s'appuie sur une encapsulation aveugle de ces données. L'agent PAw ne peut pas reconnaître par lui-même le caractère personnel d'une information, ce problème de caractérisation automatique restant encore ouvert pour le moment. Même en se plaçant hors du cadre automatisé, la notion de caractère personnel est souvent sujette à débat et à interprétation lors des litiges. Il nous paraît donc pour le moins ambitieux d'intégrer une telle fonctionnalité dans un agent cognitif. Ce problème est comparable à celui de la traduction automatique du droit en normes formelles, qui occupe activement la communauté scientifique de l'informatique légale [Mar03, Sei07].

7.1.3 Validation de la gestion protocolaire

Les connaissances de l'agent PAw en ce qui concerne les architectures disponibles sont limitées aux quelques cas de figure que nous avons identifiés. La conception des scénarios que nous allons détailler par la suite a montré, comme nous l'avions annoncé, que certaines de ces architectures n'étaient pas applicables à tout type d'application. C'est notamment le cas de l'architecture complémentaire que nous avons proposée, qui n'est pas utilisable lorsque les agents tiers doivent conserver des données à caractères personnel une fois la transaction terminée.

En l'état actuel, les protocoles en eux-mêmes ne sont pas mis en œuvre par l'agent, comme nous l'avons expliqué dans le chapitre 6. Leur capacité à assurer la protection étendue des données personnelles n'est donc pas évaluable dans le cadre de notre démonstrateur. Nous conservons donc comme référence les études menées sur les différentes architectures à titre individuel et les analyses que nous avons portées sur ces dernières dans le chapitre 5. Une évaluation en situation réelle nécessiterait de toute manière, dans le cas des architectures fondées sur le *Trusted Computing*, une réelle expertise dans le domaine des composants cryptographiques matériels.

7.2 Scénarios applicatifs

Nous avons précédemment identifié l'informatique médicale, le commerce électronique et l'intelligence ambiante comme des domaines d'intérêt pour la conception de scénarios d'évaluation de la protection des données personnelles. En conséquence, nous proposons trois familles de scénarios permettant de mettre en valeur certaines caractéristiques de l'agent PAw. Ces scénarios ont pour but de montrer que les méthodes que nous avons développées pour permettre à l'agent PAw de raisonner sur la protection des données personnelles ne sont pas limitées à un champ applicatif donné. Ces scénarios n'ont pas vocation à être complètement réalistes pour chacun des domaines donnés, mais uniquement à illustrer des problématiques applicatives caractéristiques. Malgré leur simplicité, ils mettent en œuvre les concepts de conflits d'obligations, de protection locale et de protection étendue, permettant ainsi de mesurer l'intérêt de l'utilisation d'un agent de type PAw dans ces situations.

7.2.1 Scénario de gestion médicale

Le premier scénario se situe dans le domaine de la gestion d'un dossier médical informatisé. L'agent PAw est ici responsable de données médicales appartenant à des patients et susceptibles d'être consultées par le personnel médical et administratif d'un établissement hospitalier. Il devra faire en sorte que la consultation et l'utilisation de ces données particulièrement sensibles se fasse conformément à divers jeux de règles.

7.2.1.1 Autorités normatives

Dans ce scénario, l'autorité normative `self` représente les préférences personnelles de l'utilisateur relatives à ses données médicales. L'autorité `hospital` représente l'ensemble des réglementations qui régissent le fonctionnement d'un établissement de soins, dans lequel est admis l'utilisateur. Suivant les variantes du scénario, différents ensembles de normes seront associés à ces autorités.

7.2.1.2 Acteurs

Les acteurs suivant interviennent dans le scénario :

- `uAgent` : l'agent utilisateur PAw, embarqué sur un terminal portable ;
- `adminAgent` : un agent chargé de tâches administratives dans l'hôpital, comme l'enregistrement des patients ou la consignation des actes médicaux ;
- `docAgent` : un agent représentant un docteur en médecine rattaché à l'hôpital ;
- `nurseAgent` : un agent représentant un infirmier de l'hôpital.

7.2.1.3 Aperçu du scénario

L'agent `uAgent` est ici embarqué dans un terminal mobile, confié à un patient d'un hôpital. Cet agent gère un certain nombre d'informations personnelles de nature médicale sur le patient[2]. Ces informations sont requises par les autres agents du système, représentant le personnel hospitaler, afin de mener à bien certaines opérations. En fonction du contexte normatif et des caractéristiques des transactions proposées, `uAgent` acceptera ou non de collaborer.

7.2.1.4 Déroulement détaillé et alternatives

Ce scénario permet d'illustrer un cas de figure dans lequel une prévalence normative des préférences de l'utilisateur sur les autres autorités peut entraîner des conséquences graves. Le scénario débute à l'admission de l'utilisateur dans un hôpital, en vue d'un diagnostic et d'un traitement. `uAgent` commence donc par prendre en compte l'autorité `hospital` dans son contexte normatif, c'est-à-dire qu'il prend connaissance de son réglement interne et de ses procédures. On suppose alors que l'utilisateur a le choix de la relation de prévalence normative. Il peut donc décider soit que ses préférences propres prévalent sur les règles de l'hôpital (7.1), soit qu'il se soumet prioritairement à ces dernières (7.2). Les jeux de préférences personnelles utilisées dans ce scénario étant pour certains assez protecteurs et génériques, ils entraîneront facilement conflits et arbitrages.

$$\texttt{self} \lhd_{\texttt{self}} \texttt{hospital} \tag{7.1}$$

$$\texttt{hospital} \lhd_{\texttt{self}} \texttt{self} \tag{7.2}$$

`uAgent` entre ensuite en communication avec `adminAgent` qui se chargera de son admission, requérant un certain nombre d'informations personnelles. Si le contexte normatif de l'agent est tel qu'il lui interdit de divulguer son numéro de sécurité sociale (7.3), alors l'accès à l'hôpital lui est refusé et l'utilisateur est livré à lui-même. Dans le cas contraire, il est admis en attente d'un diagnostic.

$$\begin{array}{c} \mathit{dataType}(\mathrm{SomeProcess}, \mathrm{SSnumber}, \\ \texttt{medicalSchema.ssNumber}) \end{array} \rightarrow \begin{array}{c} For^{\texttt{self}}_{\texttt{self}}(\mathit{tell}(\texttt{self}, \mathrm{SomeAgent}, \\ \mathrm{SomeProcess}, \mathrm{SSnumber})) \end{array} \tag{7.3}$$

Un agent `nurseAgent` va alors lui poser des questions d'ordre médical afin d'effectuer un prédiagnostic et de décider si la visite d'un médecin est nécessaire. Suivant ses préférences, l'utilisateur peut accepter ou refuser de dévoiler certaines informations, pouvant empêcher l'action de l'infirmier et ainsi retarder un traitement éventuel qui aurait pu ne pas nécessiter l'intervention d'un médecin. À la fin de l'entretien, `nurseAgent` propose à `uAgent` de l'ausculter à l'aide d'un

[2] Nous avons choisi ici de stocker les informations médicales sur un terminal mobile afin d'illustrer les caractéristiques du centrage utilisateur appliqué au stockage des profils. Ce choix n'a qu'une valeur d'exemple et ne constitue pas une proposition en terme d'informatique médicale, et nous n'affirmons pas qu'il est plus sensé ni plus adapté qu'un autre. La conception des architectures d'informatique médicale dépasse largement le cadre de ce travail.

stéthoscope, ce qui va à l'encontre des normes de l'hôpital qui réserve cet acte médical à un docteur en médecine. Ici, même si les préférences de l'utilisateur sont très permissives, `uAgent` refuse l'opération car l'interdiction de l'hôpital prévaut. Ce refus n'est rendu possible que parce que l'arbitrage de conflits utilise la notion de conflits d'obligations et non de conflits normatifs.

L'utilisateur est ensuite présenté à un médecin `docAgent`. De manière similaire, le médecin procède à un diagnostic complet, qui n'est interrompu que si la configuration de `uAgent` est trop protectrice. Dans ce cas, le personnel de l'hôpital ne peut poursuivre son travail et le scénario s'achève. Dans le cas contraire, le médecin diagnostique une appendicite et informe le patient qu'il devra subir une appendicectomie. Ce contexte donné, `docAgent` et `adminAgent` collectent de nouvelles informations pour la planification de l'opération. La configuration de l'agent doit normalement permettre de fournir ces informations, sachant qu'elles seront utiles à une appendicectomie.

Le scénario s'achève normalement lorsque l'utilisateur sort de l'hôpital après avoir pris rendez-vous pour son opération.

7.2.2 Scénario d'intelligence ambiante

Le deuxième scénario que nous proposons est orienté vers l'intelligence ambiante. L'agent PAw y est embarqué dans un artéfact intelligent interagissant avec son environnement. Il devra organiser sa collaboration avec les autres artéfacts du système, en limitant la diffusion d'informations sensibles.

7.2.2.1 Autorités normatives

Dans ce scénario, l'unique autorité normative est `self`, représentant les préférences personnelles de l'utilisateur. Ce scénario n'exploite donc pas la notion de conflit d'obligations. Les préférences permettront à l'agent PAw de déterminer si les demandes de données faites par les équipements ambiants sont acceptables.

7.2.2.2 Acteurs

Les acteurs suivant interviennent dans le scénario :
- `uAgent` : l'agent utilisateur PAw, embarqué dans un appareil de type PDA ou SmartPhone ;
- `lightAgent` : un agent d'ambiance embarqué dans un dispositif d'éclairage ;
- `musicAgent` : un agent d'ambiance embarqué dans un dispositif de diffusion musicale ;
- `centralAgent` : un agent de contrôle de l'ambiance situé dans l'ordinateur de pilotage de la maison. Cet agent est aussi celui qui reçoit les signaux lui indiquant la position des différents agents PAw dans le bâtiment. Dans ce scénario, cet agent est donc capable de produire une trace pour chaque agent, pouvant révéler des informations de nature privée. Ceci constitue un problème évident, mais nous considérons ici que c'est une hypothèse simplificatrice de notre scénario, destiné à observer le traitement de données manipulées par l'agent PAw et non générées implicitement par lui. Dans une architecture plus réaliste et complètement distribuée, aucun agent n'aurait de vision aussi globale de l'occupation du bâtiment par les agents utilisateurs.

7.2.2.3 Aperçu du scénario

L'agent PAw est ici encore embarqué dans un terminal mobile, porté par un utilisateur humain évoluant dans un « bâtiment intelligent » capable d'adapter son ambiance lumineuse et

musicale aux préférences de ses usagers. L'agent PAw n'a pas d'objectif propre, sinon celui de permettre à son propriétaire de profiter au mieux des dispositifs proposés tout en protégeant ses données personnelles.

7.2.2.4 Déroulement détaillé et alternatives

Lorsque `uAgent` pénètre dans le bâtiment et est détecté par lui, `centralAgent` prend contact avec lui afin de mettre en œuvre une personnalisation adaptée de l'environnement.

En ce qui concerne l'ambiance lumineuse, le bâtiment a deux modes de fonctionnement qui vont correspondre à deux propositions possibles de `centralAgent`. Dans le premier mode, `centralAgent` manipule lui-même les préférences de l'utilisateur. Celles-ci sont stockées pour permettre la persistance des environnements entre deux visites, et les agents de type `lightAgent` sont pilotés à distance en fonction des agents utilisateurs qui sont détectés à proximité. Dans le second mode, `centralAgent` ne stocke aucune préférence et indique seulement aux agents `lightAgent` les identités des agents PAw à proximité. Les agents `lightAgent` contactent alors directement les agents PAw en leur demandant leurs préférences, qui ne sont pas stockées mais utilisées immédiatement. Lorsque plusieurs agents PAw sont à proximité d'un même agent `lightAgent`, ce dernier fait une moyenne des luminosités préférées des agents, ce qui fait que les deux modes peuvent fonctionner simultanément : un agent `lightAgent` peut travailler sur un certain nombre de préférences obtenues directement d'une part, et une moyenne anonyme, fournie par `centralAgent` et assortie d'une pondération, d'autre part. Les deux modes ont des inconvénients de nature différente. En effet, on peut constater que dans le premier mode, les informations de préférences sont stockées de manière centralisée et conservées plus longtemps, alors que les agents locaux ne traitent aucune information personnelle. Dans le second mode, aucune information n'est conservée, mais par contre un certain nombre d'agents locaux manipulent les préférences et pourraient potentiellement tracer l'utilisateur indépendamment de la sécurité du serveur central. En fonction des normes de l'autorité `self`, l'agent `uAgent` va accepter l'un ou l'autre de ces deux modes, ou encore aucun, renonçant aussi à cet aspect de la personnalisation.

Par exemple, on peut supposer que l'utilisateur interdit la transmission des informations de préférences en matière de luminosité à des tiers (7.4) et que l'agent dispose dans sa connaissance du monde d'une règle permettant d'inférer les transmissions de données futures à partir d'une déclaration de type *informForward* (7.5)[3].

$$\left.\begin{array}{l} \mathit{dataType}(\text{ProcessID}, \text{DataID}, \\ \quad \texttt{ambientSchema.pref.light}) \\ \wedge \mathit{responsible}(\text{SomeAgent}, \text{ProcessID}) \\ \wedge \neg \mathit{responsible}(\text{SomeAgent}, \text{ProcessID2}) \end{array}\right\} \rightarrow \begin{array}{c} For^{\texttt{self}}_{\texttt{self}}(\mathit{forward}(\text{ProcessID}, \text{DataID}, \\ \text{ProcessID2}, \text{DataID2})) \end{array} \tag{7.4}$$

$$\left.\begin{array}{l} P^{-}\ \mathit{informForward}(\text{SomeAgent}, \text{ProbablyMe}, \\ \quad \texttt{ProcessID}, \texttt{DataID}, \text{ThirdAgent}) \\ \wedge \mathit{consent}(\text{ProbablyMe}, \text{SomeAgent}, \text{ProcessID}) \end{array}\right\} \rightarrow \begin{array}{l} F\ \mathit{forward}(\text{ProcessID}, \text{DataID}, \\ \quad \texttt{processPlaceholder}, \\ \quad \texttt{dataPlaceholder}) \end{array} \tag{7.5}$$

[3] La représentation, dans cette formule, de l'hypothétique processus destinataire des données par le moyen d'un littéral arbitraire (`processPlaceholder`) permet de raisonner sur le transfert futur des données même si l'on ne dispose pas déjà d'informations sur les traitements mis en œuvre par ThirdAgent. Pour des raisons d'expressivité qui dépassent le cadre de cet exemple, cette formulation est en réalité associée à une seconde, prenant pour condition la préconnaissance de la responsabilité de ThirdAgent dans un traitement donné, en représentant ce traitement destinataire par une variable libre (permettant ainsi sa réutilisation efficace dans d'autres raisonnements).

Dans ce cas, `uAgent` anticipe la transmission des préférences aux agents `lightAgent` et détecte une possible violation de ses normes, ce qui le pousse à refuser le premier mode d'interaction. De la même manière, si une norme lui interdit de dévoiler directement ses préférences aux agents `lightAgent`, le second mode sera refusé. Si les deux normes sont actives, `uAgent` refusera les deux modes et ne pourra profiter de la personnalisation de l'ambiance lumineuse.

La gestion de l'ambiance musicale se fait d'une manière similaire. Dans le premier mode, `centralAgent` est capable de stocker des listes de lecture pour chaque utilisateur et d'envoyer des suggestions de titres aux agents `musicAgent`. Dans le second mode, l'agent `musicAgent` demande directement un titre aux agents `uAgents` à proximité. Les problèmes de stockage des préférences sont les mêmes, mais ici l'agent utilisateur n'a pas à dévoiler sa liste de lecture complète aux agents `musicAgent`, ce qui fait que le second mode a davantage d'intérêt pour l'ambiance musicale que pour l'ambiance lumineuse.

À cause des ressources limitées habituellement attribuées aux équipements d'informatique ambiante, seule l'architecture directe est proposée : on se situe au niveau de confiance le plus bas et l'agent PAw doit faire confiance aux agents tiers.

7.2.3 Scénario de commerce électronique

Le troisième scénario applicatif proposé place l'agent PAw dans la situation d'un agent personnel dans le cadre d'une transaction de commerce électronique. Il doit interagir avec des agents de service et des agents tiers de confiance, en vue d'obtenir un service commercial dans un environnement ouvert de type Internet. La personnalisation de ce service nécessitant des informations de profil de l'utilisateur humain, l'agent PAw devra gérer au mieux la protection de ces données.

7.2.3.1 Autorités normatives

Dans ce scénario, l'agent sera soumis à une autorité `law`, représentant le cadre légal national dans lequel il s'exécute, et à une autorité `self` représentant les préférences de l'utilisateur en matière de protection des données personnelles.

7.2.3.2 Acteurs

Les acteurs suivant interviennent dans le scénario :
- `uAgent` : un agent utilisateur PAw ;
- `sellerAgent` : un agent représentant une société commerciale marchande de biens ;
- `transporterAgent` : un agent représentant une société commerciale de livraison ;
- `bankAgent` : un agent représentant une société bancaire.

7.2.3.3 Aperçu du scénario

L'utilisateur humain, représenté par l'agent PAw `uAgent`, est un particulier cherchant à acquérir un produit auprès de la société représentée par `sellerAgent`. Cette intention déclarée, `sellerAgent` va faire une proposition à `uAgent` quant à la marche à suivre et aux flux d'informations nécessaire pour la réalisation de la transaction. En fonction de la proposition et du contexte normatif de `uAgent`, une courte négociation va s'engager pour aboutir soit à un protocole opératoire satisfaisant les deux parties, soit à un échec.

7.2.3.4 Déroulement détaillé et alternatives

Le scénario commence après que `uAgent` a exprimé auprès de `sellerAgent` son intention d'acquérir le produit. `sellerAgent` peut alors proposer deux formulations pour ses requêtes :

– La première option consiste à demander toutes les informations nécessaires, y compris celles de paiement et de livraison, et à déclarer que ces informations seront partagées, respectivement, avec `bankAgent` et `transporterAgent`. Les formules (7.6) à (7.13) sont extraite de la proposition correspondante faite par `sellerAgent`. Ce dernier informe notamment `uAgent` des données nécessaires au traitement `sale123` (formules (7.6) à (7.9)) et des destinataires potentiels de certaines de ces données (formules (7.10) à (7.12)) avant de demander son consentement (7.13).

$$\mathit{request}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{user.namedName}) \tag{7.6}$$
$$\mathit{request}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{sellerSchema.item}, \texttt{dItem}) \tag{7.7}$$
$$\mathit{request}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{Ecom.Payment.Card.Number}, \texttt{dCard}) \tag{7.8}$$
$$\mathit{request}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{user.home-info.postal}, \texttt{uAddress}) \tag{7.9}$$
$$\mathit{informForward}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{dName}, [\texttt{bankAgent}, \texttt{transporterAgent}]) \tag{7.10}$$
$$\mathit{informForward}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{dCard}, [\texttt{bankAgent}]) \tag{7.11}$$
$$\mathit{informForward}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{dAddress}, [\texttt{transporterAgent}]) \tag{7.12}$$
$$\mathit{consentRequest}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}) \tag{7.13}$$

– La deuxième option consiste à demander les informations d'identification du produit demandé ainsi qu'un certificat (utilisant une solution de monnaie électronique supposée disponible) comme quoi le produit a bien été payé auprès de `bankAgent` (il revient alors à `uAgent` de contacter lui-même `transporterAgent` en lui délivrant un nouveau certificat fourni par `sellerAgent`). Les formules (7.14) à (7.20) sont extraite de la proposition correspondante faite par `sellerAgent`, les formules (7.21) à (7.25) représentent la manière dont `bankAgent` présente la transaction de paiement à `uAgent` et les formules (7.26) à (7.32) traitent de la demande de données faite par `transpoterAgent` à `uAgent` pour la livraison du produit.

$$\mathit{request}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{user.name}, \texttt{dName}) \tag{7.14}$$
$$\mathit{request}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{sellerSchema.item}, \texttt{dItem}) \tag{7.15}$$
$$\mathit{request}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{sellerSchema.paymentCertificate}, \texttt{dCertif}) \tag{7.16}$$
$$\mathit{informForward}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{dName}, []) \tag{7.17}$$
$$\mathit{informForward}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{dItem}, []) \tag{7.18}$$
$$\mathit{informForward}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}, \texttt{dCertif}, []) \tag{7.19}$$
$$\mathit{consentRequest}(\texttt{sellerAgent}, \texttt{uAgent}, \texttt{sale123}) \tag{7.20}$$

$$\mathit{request}(\texttt{bankAgent}, \texttt{uAgent}, \texttt{payment123}, \texttt{user.name}, \texttt{dName}) \quad (7.21)$$
$$\mathit{request}(\texttt{bankAgent}, \texttt{uAgent}, \texttt{payment123}, \texttt{Ecom.Payment.Card.Number}, \texttt{dCard}) \quad (7.22)$$
$$\mathit{informForward}(\texttt{bankAgent}, \texttt{uAgent}, \texttt{payment123}, \texttt{dName}, []) \quad (7.23)$$
$$\mathit{informForward}(\texttt{bankAgent}, \texttt{uAgent}, \texttt{payment123}, \texttt{dCard}, []) \quad (7.24)$$
$$\mathit{consentRequest}(\texttt{bankAgent}, \texttt{uAgent}, \texttt{payment123}) \quad (7.25)$$

$$\mathit{request}(\texttt{transporterAgent}, \texttt{uAgent}, \texttt{delivery123}, \texttt{user.name}, \texttt{dName}) \quad (7.26)$$
$$\mathit{request}(\texttt{transporterAgent}, \texttt{uAgent}, \texttt{delivery123}, \texttt{user.home-info.postal}, \texttt{dAddress}) \quad (7.27)$$
$$\mathit{request}(\texttt{transporterAgent}, \texttt{uAgent}, \texttt{delivery123}, \texttt{sellerSchema.saleCertificate}, \texttt{dCertif}) \quad (7.28)$$
$$\mathit{informForward}(\texttt{transporterAgent}, \texttt{uAgent}, \texttt{delivery123}, \texttt{dName}, []) \quad (7.29)$$
$$\mathit{informForward}(\texttt{transporterAgent}, \texttt{uAgent}, \texttt{delivery123}, \texttt{dAddress}, []) \quad (7.30)$$
$$\mathit{informForward}(\texttt{transporterAgent}, \texttt{uAgent}, \texttt{delivery123}, \texttt{dCertif}, []) \quad (7.31)$$
$$\mathit{consentRequest}(\texttt{transporterAgent}, \texttt{uAgent}, \texttt{delivery123}) \quad (7.32)$$

Pour la première option, les architectures qui peuvent être proposées lors de la phase de négociation sont la communication directe, le *self-profiling* et les *sticky policies*. La livraison physique de l'objet faisant partie de la transaction globale, l'architecture déportée n'est pas utilisable. La seconde option, en revanche, ne fait appel qu'à des concepts logiciels et peut tirer parti de n'importe laquelle des architectures proposées.

L'agent `uAgent`, en fonction de son contexte normatif (suivant si celui-ci autorise ou non le transfert d'informations bancaires d'un agent à l'autre), va accepter ou refuser l'option proposée, et le cas échéant négocier l'architecture qui lui convient le mieux. La transaction s'opère ensuite. Dans notre cas, elle est uniquement simulée, représentée au sein des agents par les prédicats performatifs de $\mathcal{L}_{DLP}$. Dans le cas de la deuxième option, des négociations indépendantes ont également lieu entre `uAgent` et les deux autres agents sur les architectures à utiliser.

7.3 Discussion

Un des principaux avantages de l'utilisation des scénarios présentés est qu'ils permettent d'observer l'influence du contexte normatif sur les décisions prises par l'agent, et donc sur les transactions opérées. En modifiant les normes émises par telle ou telle autorité (ou même en modifiant uniquement les normes émises par le propriétaire de l'agent), on peut observer un changement radical dans le comportement de l'agent PAw, par exemple :

- Dans le premier scénario, la moindre dérive dans les normes exprimées ou dans la précédence des autorités normatives peut avoir des conséquences sur la santé de l'utilisateur humain ;
- Dans le deuxième, la représentation interne des opérations effectuées par l'environnement d'informatique ambiante peut amener l'agent PAw à fournir des informations de profil soit localement à un contrôleur disposant de peu de ressources, soit de manière centralisée à un terminal capable de les conserver et de les traiter ;

– Dans le troisième scénario, les valeurs numériques de préférences pour les différentes architectures distribuées peuvent amener l'agent à traiter avec trois agents tiers au lieu d'un seul.

L'agent PAw s'avère donc finalement, au cours de sa propre évaluation, un outil de visualisation de certaines problématiques secondaires, ou masquées, de la protection des données personnelles lorsqu'elle est déléguée à un agent cognitif.

L'utilisation des scénarios met en avant un besoin d'amélioration assez pressant, qui est l'intégration des divers modèles de fonctionnement (par exemple, le nombre d'interlocuteurs de l'agent PAw dans le premier scénario) au sein de la phase de négociation. Dans le cadre des scénarios proposés, on se limite à un ensemble prédéterminé et fini de quelques modèles d'interaction. Ces informations sont décrites en DLP, il est donc possible de raisonner dessus, mais les mécanismes de négociation basiques que nous utilisons ne permettent pas d'interagir de manière riche sur ces descriptions tout en conservant la garantie de terminaison. Il y a donc ici une nécessité de rechercher des méthodes de négociation plus riches et plus appropriées au formalisme DLP.

7.3.1 Fusion des scénarios

On peut de plus légitimement s'interroger sur la pertinence d'un méta-scénario, permettant au même agent PAw d'interagir à la fois dans le cadre de l'informatique médicale, dans des environnements d'intelligence ambiante et dans des transactions de commerce électronique. Le même `uAgent` doit donc disposer des normes se rapportant à ces trois domaines. Les normes étant des formules partiellement instanciées, il faudrait alors les concevoir avec soin afin d'éviter les collisions. Une norme définie pour une application médicale, si elle est conçue de manière trop générique, pourrait influencer le fonctionnement de l'agent lors d'une transaction commerciale, lui interdisant par exemple l'utilisation d'une architecture applicative donnée. Il existe au moins deux manières de concevoir un agent PAw à la fois efficace et polyvalent.

On peut tout d'abord imaginer de caractériser spécifiquement toutes les normes qui s'appliquent à l'agent, en ajoutant notamment dans l'antécédent des normes des conditions impliquant les prédicats de type *informActionType* ou *actionType*. L'efficacité du cloisonnement dépend alors de la capacité des rédacteurs à anticiper l'ensemble des scénarios possibles, ce qui n'est guère raisonnable : un gérant d'entreprise ne devrait pas avoir à s'inquiéter de l'impact que pourraient avoir ses directives sur les interactions de ses employés avec leurs appareils électroménagers.

La deuxième option, plus raisonnable, consisterait à compartimenter les domaines d'agissement de l'agent PAw. En définissant un contexte normatif (soit une liste d'autorités et une relation de prévalence) distinct pour chaque type d'activité, on interdit les interactions abusives. L'agent PAw procèderait alors à une commutation de contexte (comme suggéré dans la section 4.1.4.3) à chaque changement d'activité. Par exemple, le gérant d'entreprise cité plus haut ne serait plus considéré comme une autorité normative lorsque l'employé est chez lui en dehors des heures de travail. Les contextes normatifs pourraient alors correspondre aux différentes identités virtuelles définies par l'utilisateur.

Le recours à la commutation de contexte pose toutefois quelques limitations. D'une part, les informations de contexte devraient être définies à nouveau pour chaque nouvelle identité virtuelle, ce qui peut s'avérer rébarbatif pour l'utilisateur (une hypothèse simplificatrice étant de conserver la même relation de prévalence et de faire varier uniquement l'ensemble des autorités). D'autre part, il faut que les croyances à caractère personnel soient enregistrées par l'agent en relation avec un contexte, de manière à leur assurer une protection appropriée. Cela implique un

cloisonnement des bases de croyances et, suivant les méthodes d'implémentation, une possible duplication de certaines informations.

Conclusion et Perspectives

Synthèse des travaux

Nous avons discuté dans cette thèse des fonctionnalités spécifiques nécessaires à un agent pour assurer une protection efficace et contextualisée des données personnelles.

État de l'art

Nous avons tout d'abord présenté et discuté le concept de sphère privée tel qu'il apparaît au quotidien, et notamment dans les réglementations opposables aux utilisateurs humains. Nous avons limité notre champ d'application à la protection des données personnelles en informatique et extrait les six axes réglementaires concernant cette protection. Nous avons mis en exergue le besoin spécifique d'une protection étendue des données et discuté de la pertinence de quelques familles de méthodes et d'outils techniques existants. Nous avons mis en avant le besoin de raisonner directement et automatiquement sur une expression formelle de réglementations hétérogènes et examiné les outils existant pour ce faire. Les points délicats se sont avérés être en particulier la gestion conjointe des notions déontiques et temporelles, ainsi que le traitement des incohérences normatives.

Outils théoriques

Nous avons ensuite proposé une architecture générique pour un agent PAw (*Privacy-Aware*), incluant des modules destinés au raisonnement normatif et à la gestion de la protection étendue des données personnelles. Nous avons présenté un modèle logique, agencé autour d'un formalisme spécifique (la logique DLP), pour traiter des réglementations en matière de données personnelles, incluant des méthodes de raisonnement sur les conflits d'obligation et les notions déontiques avec échéances. Le besoin de mettre en œuvre une protection étendue des données personnelles nous a amené à examiner les possibilités offertes par les technologies liées au *Trusted Computing*. Il s'est avéré que suivant l'utilisation qui était faite de ces outils, ils pouvaient assurer des propriétés distantes sur les données mais également constituer un risque non négligeable pour l'utilisateur. Après avoir discuté les problèmes posés par les architectures existantes, nous en avons proposé une complémentaire, utilisable dans certains cas de figure applicatifs et mettant en avant une utilisation alternative du *Trusted Computing* délibérément tournée vers la protection de l'utilisateur.

Mise en œuvre

Nous nous sommes ensuite appliqués à implémenter le modèle théorique proposé, au sein d'un agent développé sur la plate-forme multi-agent JACK [AOS]. Une couche de raisonnement normatif, utilisant principalement le langage Prolog, a été mise en place pour prendre en charge

les fonctionnalités du modèle logique de traitement des réglementations. Cette couche fournissant une politique de traitement des données personnelles à l'agent PAw, la gestion des croyances du modèle JACK a été adaptée pour que tout accès d'une méthode de l'agent à des croyances de nature personnelle soit subordonnée au respect des normes applicables. Nous avons doté l'agent PAw de connaissances factuelles sur les propriétés de quelques familles d'architectures d'application, ainsi que de fonctionnalités de négociation lui permettant de choisir en toute circonstance celle qui sera la plus adaptée à la protection étendue des données personnelles dont il a la charge. Nous nous sommes enfin penchés sur les caractéristiques de notre implémentation, et notamment sur la correspondance de la mise en œuvre en Prolog avec le modèle logique original. Nous nous sommes assurés que les nécessaires limitations en termes d'inférence ne désservaient pas les fonctionnalités recherchées. Trois scénarios applicatifs ont été proposés, dans les domaines de la gestion médicale, de l'intelligence ambiante et du commerce électronique, pour illustrer le possible rôle d'un agent PAw dans les interactions entre utilisateurs humains, organisations et agents artificiels. L'utilisation de l'agent PAw dans le cadre de ces scénarios nous a notamment permis de mettre en lumière des relations critiques entre l'expression formelle des réglementations, les préférences de l'utilisateur en matière d'autorités normatives et les résultats effectifs d'un ensemble de transactions.

Bilan

L'objectif de cette thèse était de fournir des modèles et des outils pour la réalisation d'un **agent personnel capable de protéger, de manière sensée et efficace, les informations à caractère personnel qui lui sont confiées**. Dans la mesure des limitations que nous nous sommes fixées (à savoir notamment la restriction à la « protection des données personnelles » telle que nous l'avons définie dans la section 1.1.2.5, la préconnaissance de la nature des informations et la cohérence, à titre individuel, des autorités normatives), nous pouvons affirmer que nous avons rempli cet objectif avec la conception de l'**agent PAw**. En effet :

- L'agent PAw **prend en compte le contexte législatif et réglementaire** par le moyen de la logique DLP ;
- Il s'**adapte dynamiquement à ce contexte** grâce au composant de raisonnement associé, permettant de produire une politique de sécurité cohérente ;
- Il **contrôle sa propre utilisation des informations** grâce à la subordination des intentions métier à cette politique ;
- Il **s'assure que les informations sont utilisées correctement par les autres agents** grâce aux informations dont il dispose sur les architectures d'application disponibles et à ses capacités de négociation.

Nos travaux nous ont également permis d'établir ou de confirmer les résultats suivants, qui n'étaient pas initialement pressentis comme directement liés à l'objet de la thèse :

- L'association entre la logique déontique standard et la logique temporelle linéaire permet de construire des modèles théoriques utilisables pour le raisonnement normatif, la production automatisée de politiques et le raisonnement sur les inconsistances ;
- La notion de système normatif doit être adaptée pour pouvoir prendre en compte les problématiques de protection des données personnelles [CPBD07, Pio08] ;
- Introduction de **six axes réglementaires** pour caractériser les contraintes en la matière [PDC06] ;
- Introduction de la notion de **conflit d'obligations**, qui doit être préférée dans certains cas à celle de conflit normatif issue de la logique déontique standard, et proposition d'une

procédure de détection et d'arbitrage de ces conflits [PD08c] ;
- Établissement de critères génériques pour la conception d'**opérateurs déontiques datés** et propositions dans le cadre du produit entre logique déontique standard et logique temporelle linéaire, sans extension au langage [PD08a, PD08b] ;
- Proposition d'une architecture alternative utilisant les *Trusted Computing Platforms*, utilisable dans certains cas applicatifs et ne présentant pas de risques significatifs pour les parties impliquées.

D'autre part, il convient de considérer que dans le cadre des travaux présentés ici, nous nous sommes limités à une certaine vision du problème de la protection de la sphère privée. Les résultats et interrogations qui découlent de cette recherche ouvrent des axes de travail plus précis, concernant des points délicats que nous avons soulevés, ou bien au contraire visant à étendre et élargir ces travaux à des problématiques connexes. Ces nouvelles perspectives de recherche portent sur les aspects théoriques du modèle logique, sur la protection des données personnelles ou sur la gestion de la sphère privée dans son ensemble.

Perspectives sur les aspects de modélisation

Prise en compte des conflits au sein d'une autorité normative

Lorsque nous nous sommes attelés à la gestion des conflits d'obligations, nous avons travaillé sous l'hypothèse simplificatrice qui veut que l'ensemble des normes émises par une même autorité doive être cohérent. Dans le cas contraire, l'agent PAw ignore l'autorité en question. Or, ces normes sont issues de réglementations écrites par des humains en langue naturelle, et il est fort probable qu'elles soient intrinsèquement inconsistantes. Dans le domaine légal, les juristes humains savent évaluer ces inconsistances au regard de l'esprit de la loi ou de la jurisprudence, mais cette expertise n'est sans doute pas encore à la portée d'un programme informatique.

Néanmoins, même si la traduction et l'interprétation de la loi (ou des textes réglementaires d'une manière générale) restent pour l'instant de la compétence de l'humain, ce dernier peut souhaiter conserver, dans l'expression formelle résultante, les inconsistances du texte original. Il peut également ne pas les déceler ou ne pas savoir les résoudre. Dans cette optique, l'acceptation par l'agent PAw d'ensembles de normes intrinsèquement inconsistants pourrait être une amélioration du système (dès lors que l'on accepte cette altération philosophique des hypothèses de travail).

Au sein d'une même autorité, permissions et obligations ont été énoncées explicitement. Si les conflits dans ces énoncés ont été conservés, c'est qu'ils peuvent être d'importance, une permission pouvant éventuellement, dans certains cas, prendre le pas sur une interdiction[4]. La stratégie visant à ignorer les permissions pour ne considérer que les obligations n'est donc plus forcément valable. Il conviendrait donc de concevoir un mécanisme de défaisabilité différent, prévu pour cohabiter et interférer avec l'algorithme d'arbitrage des conflits d'obligations présenté ici. Si l'on considère qu'il n'est pas du ressort de l'agent PAw de trancher de telles incohérences, alors l'algorithme d'arbitrage (ainsi que l'ensemble des outils de raisonnement de l'agent PAw) doit être repensé pour permettre un raisonnement sur des formalismes paraconsistants [PT04], permettant de représenter ces conflits sans pour autant chercher à les résoudre. Ce choix de conception, s'il prend davantage en compte la réalité du contexte normatif, implique une plus grande interactivité avec l'utilisateur humain, qui doit lui-même arbitrer certains des conflits.

[4] À l'inverse, si permission et interdiction ont été énoncées par deux autorités distinctes, cette collision n'était vraisemblablement pas envisagée à l'origine et peut ne pas avoir de signification fondamentale. C'est ce qui motive en partie notre choix de mise à l'écart des conflits impliquant des permissions.

Raisonnement sur les croyances déontiques d'autres agents

Des capacités de méta-raisonnement déontique pourraient permettre à un agent de type PAw d'améliorer de manière significative sa mise en œuvre de la protection étendue des données. Le comportement de l'agent pourrait en effet être influencé par une prise en compte de la manière dont les autres agents du système considèrent les données personnelles. Dans le cadre de nos travaux, nous avons conçu un agent bienveillant, capable de respecter les réglementations sur la sphère privée, en fonction d'un certain nombre d'autorités reconnues par lui. Il nous semble maintenant souhaitable et possible de lui permettre d'évaluer le risque inhérent à la divulgation d'une information donnée à un agent en particulier, en fonction d'un ensemble de croyances sur les autorités normatives que ce dernier reconnaît ou sur la valeur qu'il octroie au caractère privé des informations.

Si l'agent PAw sait que l'agent distant et lui ont en commun un certain nombre d'autorités, il pourra être plus enclin à lui confier des données sensibles. De la même manière, si l'agent sait qu'une autorité particulière est vue avec une prévalence normative très forte par un autre agent, alors il pourra anticiper l'arbitrage d'éventuels conflits d'obligations et en déduire des informations sur certaines des normes que l'agent distant considère comme importantes. Ces informations portant sur les croyances déontiques d'agents tiers seraient de nature à alimenter une nouvelle mesure de confiance cognitive, dont la valeur dépendrait de la qualité des informations obtenues sur le contexte normatif des autres agents du système.

Perspectives sur la protection des données personnelles

Raisonnement sur l'inférence de données personnelles

Un axe de travail intéressant consiste à s'affranchir d'une des limitations de nos travaux jusqu'à présent, qui consistait à considérer individuellement les données personnelles manipulées par les agents. La corrélation d'informations étant une menace majeure pour la vie privée, il nous semble nécessaire de la modéliser de manière appropriée. Le langage DLP permet en partie cela (de manière quelque peu implicite), de par les liens exprimés entre les traitements et les données. La problématique est cependant plus complexe, l'information divulguée pouvant être en elle-même une règle d'inférence logique, ou encore une formule destinée à être utilisée dans un processus déductif.

Ici encore c'est par une forme de méta-raisonnement que l'on pourra améliorer significativement le modèle cognitif de l'agent PAw. Il s'agirait ici de lui permettre de produire un raisonnement portant directement sur les processus d'inférence distants. De telles capacités sont nécessaire à l'agent pour déterminer avec une meilleure précision les informations supplémentaires qui peuvent être inférées par un agent distant, en fonction des données déjà divulguées (potentiellement par d'autres agents).

Qualification des croyances à caractère personnel

Une orientation de recherche qui nous semble intéressante au vu du bilan des travaux accomplis se rapporte de manière plus générale à la notion de données personnelles. Dans le cadre de la conception de l'agent PAw, nous avons supposé que les informations à caractère personnel étaient clairement identifiées et typées (au sens du référencement dans une ontologie) par l'agent, ce dernier pouvant les représenter au sein d'ensembles de croyances spécialisés. Cependant, à l'étape de la collecte des données (que ces dernières soient fournies par un agent tiers ou directement extraites par l'agent au cours de son activité), la question du typage et de l'identification

des données personnelles n'est pas du tout triviale. Il nous semblerait intéressant d'explorer les critères qui permettraient à un agent de considérer une information comme personnelle ou d'en déterminer la nature. La source de l'information ainsi que les corrélations possibles avec d'autres données personnelles jouent sans doute ici un rôle déterminant, mais qui reste à préciser. De plus, une qualification automatique des données par un agent artificiel appelle à une modélisation et à un raisonnement spécifiques sur la possibilité que cette qualification soit erronée. Cette dernière perspective nous conforte dans l'idée que la modélisation de phénomènes cognitifs artificiels est une approche essentielle dans l'élaboration des procédés techniques visant à améliorer la protection de la sphère privée d'utilisateurs humains.

Perspectives sur la gestion de la sphère privée

Protection des données non explicites de la sphère privée

La problématique des données non explicites est liée, au niveau du principe de raisonnement, à celle de l'inférence de données personnelles déjà mentionnée. Comme nous l'avons précisé dans notre définition de la protection des données personnelles (voir section 1.1.2.5), nous nous sommes limités à travailler sur des informations « explicitement représentées sous forme numérique et mises en jeu dans le cadre d'une application informatique ». Les présents travaux ne permettent donc pas à l'agent PAw de travailler sur des données traitant de l'utilisateur, mais dont ce dernier n'a pas explicitement connaissance. Une protection plus efficace de la sphère privée de l'utilisateur exige également un raisonnement sur les informations pouvant être produites ou déduites par d'autres agents sur la base du comportement de l'utilisateur, du comportement de son agent personnel ou même de croyances erronées. Les informations déduites à partir de traces d'exécution ou d'informations obtenues sous la forme de recommandations ou de réputation rentrent dans ce cadre et ne sont pour l'instant que difficilement contrôlables par l'utilisateur. Ici encore nous souhaiterions améliorer les capacités de méta-raisonnement d'un agent, mais également son activité d'introspection sur l'observabilité de ses propres actions.

Méthodes formelles pour la protection de la vie privée

Nous avons présenté ici des travaux portant à la fois sur des outils logiques pour le raisonnement cognitifs et sur des architectures d'applications distribuées. Il serait particulièrement intéressant d'approfondir cette expérience dans le domaine des méthodes formelles par la construction d'un langage logique permettant le raisonnement sur les propriétés d'une architecture applicative en regard de la protection de la sphère privée. Un tel outil logique, serait un candidat pour une contrepartie, dans le domaine de la vie privée, à ce que représente la logique BAN [BAN89] pour l'authentification. Il pourrait ainsi servir de base commune pour l'évaluation d'applications sur le plan de la protection des données personnelles.

D'une manière plus spécifique, il serait intéressant de développer un calcul formel pour raisonner sur les primitives spécifiques aux *Trusted Computing Platforms*. Si l'on pouvait décrire de manière formelle une architecture d'application, en précisant la localisation des éléments matériels et des processus cryptographiques, il serait possible d'associer à cette description une caractérisation des garanties et des risques (relatifs à la vie privée) associés à cette architecture ou à ce protocole. Une telle logique serait alors un outil formidable pour la conception et la certification d'architectures d'applications orientées vers la protection de la vie privée.

Ouverture : agents personnels et libertés individuelles

Nos travaux sur le centrage utilisateur par le biais de la protection de la sphère privée laissent entrevoir une possible évolution dans les interactions entre agents humains et agents artificiels, présageant d'une réelle intégration de l'utilisateur dans les méthodologies de conception des systèmes multi-agents. L'annexe C présente par exemple comment cette dimension peut être prise en compte comme une extension de la méthode VOYELLES [Dem96]. Le nouveau rôle joué ici par les agents personnels, vus comme les garants de la protection des données personnelles, constitue de plus une motivation pour la conception de systèmes informatiques destinés à aider les utilisateurs à défendre leurs droits. Il paraît de plus en plus crédible que des agents artificiels soient plus à même que des utilisateurs humains de comprendre, promouvoir et protéger leurs libertés individuelles dans le cadre de leurs interactions avec des organisations.

La perspective de l'utilisation d'un agent personnel de type PAw dans les transactions de la vie courante alimente également la réflexion sur la manière dont les utilisateurs perçoivent les interactions entre les textes réglementaires et leurs propres activités. S'il devient possible à un agent utilisateur d'assimiler le contenu de textes de loi et d'informer l'utilisateur des possibles influences qu'ils peuvent avoir sur des choix personnels, professionnels ou purement techniques, la compréhension de ces lois par le public s'en trouvera sans doute améliorée. Ainsi, de manière assez paradoxale, le passage d'un texte légal dans un langage formel a priori inaccessible à l'utilisateur peut aider à sa compréhension. Cela nécessite bien évidemment d'adapter les principes que nous avons introduits, de manière à fournir à l'utilisateur des informations claires, dans la langue naturelle (ou juridique) d'origine, sur les sources des normes utilisées par l'agent PAw. Nous pensons que ce type d'outils et de mécanismes peut alors participer à une nécessaire convergence et une meilleure compréhension mutuelle entre le monde juridique, le monde informatique et le grand public.

Quatrième partie

Annexes et bibliographie

Annexe A

Paradoxes de la logique déontique standard

La logique déontique standard est sujette à de nombreux paradoxes et dilemmes, issus de l'essence de l'implication logique ou de la nature de logique modale normale classique de la logique déontique. Nous présentons ici quelques-uns des plus connus.

On peut par exemple citer le dilemme de Sartre (A.1) et le dilemme de Platon (A.2), décrits par ces deux philosophes, respectivement dans *L'existentialisme est un humanisme* [Sar46] et dans *La République* [Pla], et formalisés en logique déontique par E. J. Lemmon [Lem62]. Le dilemme de Sartre dénote l'incapacité de SDL à raisonner sur des obligations portant sur une formule et sa négation. Il est en effet aisé de montrer qu'à cause de l'axiome (Ob-D), $Ob\ \varphi$ implique $\neg Ob \neg \varphi$, inconsistant avec $Ob \neg \varphi$. Si l'on raisonne sur la sémantique, $Ob\ \varphi$ implique que tous les mondes accessibles vérifient φ, $Ob \neg \varphi$ implique que tous les mondes accessibles vérifient $\neg \varphi$, et la sérialité impose l'existence d'un monde accessible. Il existe donc un monde où la contradiction $\varphi \wedge \neg \varphi$ est vraie. Il est donc impossible de décrire en logique déontique standard la situation d'un agent obligé simultanément à une chose et son contraire. Le dilemme de Platon en est une variante plus générale, qui dit l'impossibilité de représenter des obligations sur des formules inconsistantes entre elles, sans que l'une d'elle soit forcément la négation de l'autre. Ce second dilemme est plus intéressant car il caractérise l'incapacité à raisonner sur des obligations avec priorités. En effet, si φ représente le fait pour une personne d'aller à un rendez-vous au restaurant et ψ le fait d'emmener à l'hôpital son enfant qui s'étouffe, alors cet agent est bien obligé simultanément à φ et à ψ, en dépit du fait que l'un soit incompatible avec l'autre. Une de ces obligation est évidemment plus prioritaire, mais la logique déontique standard ne permet pas de raisonner sur cette notion.

$$(Ob\ \varphi \wedge Ob \neg \varphi) \vdash \bot \tag{A.1}$$

$$\text{Si } (\varphi \wedge \psi) \vdash \bot \text{ alors } (Ob\ \varphi \wedge Ob \neg \psi) \vdash \bot \tag{A.2}$$

Le paradoxe le plus sérieux de la logique déontique standard est sans dout celui de Chisholm, ou paradoxe des obligations contraires au devoir (ou encore CTD pour *contrary-to duties obligations*) [Chi63]. Ce paradoxe apparaît lorsque le fait de violer une obligation en entraîne une nouvelle. La formule (A.3) exprime l'obligation initiale, par exemple « Jones doit aller aider ses voisins ». La formule (A.4) est l'obligation d'une implication logique, comme « Il est obligatoire que, si Jones va aider ses voisins, il les en prévienne ». La formule (A.5) constitue l'obligation CTD, c'est une obligation conditionnée par la violation d'une autre obligation. Ici, « Si Jones ne

va pas aider ses voisins, alors il ne doit pas les prévenir ». La formule (A.6), quant à elle, n'est pas une formule déontique, elle exprime l'état de fait de la réalité : « Jones ne va pas aider ses voisins ».

$$Ob\ \varphi \tag{A.3}$$

$$Ob(\varphi \rightarrow \psi) \tag{A.4}$$

$$\neg\varphi \rightarrow Ob\neg\psi \tag{A.5}$$

$$\neg\varphi \tag{A.6}$$

Dans cette situation, alors même que l'ensemble de formules semble décrire une situation possible et cohérente, on peut en déduire la contradiction logique (A.7) : « Jones doit prévenir ses voisins et Jones ne doit pas prévenir ses voisins ».

$$Ob\ \psi \wedge Ob\neg\psi \tag{A.7}$$

Les travaux portant sur des tentatives pour circonvenir ce paradoxe dans le cadre de SDL (en se fondant sur des formalisations différentes des obligations CTD) conduisent à conclure que le problème d'inconsistance présenté ici ne peut être évacué qu'au prix d'une perte artificielle d'indépendance entre les quatre équations constitutives du problème.

Annexe B

Extraction de formules DLP à partir de politiques P3P

Nous présentons ici les principaux éléments d'une politique P3P [Wor06] qui peuvent servir à extraire des formules DLP (ou plus précisément des formules de $\mathcal{L}_{DLP}$). Les informations contenues dans un document P3P permettent évidemment d'enregistrer des prédicats d'état, mais le fait que la politique soit publiée constitue en lui-même un acte d'information de l'utilisateur, c'est pourquoi les prédicats informatifs correspondants sont produits également[1]. Dans les schémas de réécriture (représentée par l'opérateur $\Longrightarrow$), nous considérerons que la politique P3P émane d'un agent `serviceAgent` et est lue par un agent `userAgent` souhaitant obtenir un service d'identifiant `service`.

B.1 Éléments utilisés pour l'extraction

B.1.1 `<POLICY>`

L'élément `<POLICY>` est l'élément parent d'une politique (un document P3P pouvant en contenir une ou plusieurs au sein d'un élément racine `<POLICIES>`). La présence de cet élément signifie que l'entité éventuellement responsable d'un traitement effectue une démarche d'information auprès de l'utilisateur, dans le but de lui permettre d'exprimer son consentement ou son refus vis-à-vis de la collecte et du traitement de ses données personnelles. En conséquence, l'existence même de la politique permet la production d'un prédicat performatif *consentRequest* (figure B.1), ainsi que des prédicats d'information, comme nous le verrons par la suite.

$$\begin{array}{lcl} \texttt{<POLICY>} & & \\ \vdots & \Longrightarrow & \mathit{consentRequest}(\texttt{serviceAgent}, \texttt{userAgent}, \texttt{service}) \\ \texttt{</POLICY>} & & \end{array}$$

Fig. B.1 – Exemple de réécriture d'un élément `<POLICY>`

[1] On se souviendra que nous avons refusé, dans la section 4.2.3.8, l'axiomatisation de la correspondance entre prédicats d'état et prédicats informatifs. Cependant, dans le contexte présent, nous travaillons sur un outil d'extraction automatique de P3P vers DLP, et donc indépendant de la mécanique interne de la logique DLP. L'objectif ici étant de traduire au mieux l'information fournie à l'utilisateur, nous prenons la liberté d'introduire directement les prédicats d'état correspondant aux informations déclarées dans la politique P3P analysée.

B.1.2 `<ENTITY>`

L'élément `<ENTITY>` identifie l'entité responsable des traitements se rapportant à la politique. Il servira donc à générer une instance de prédicat d'état *responsible* pour un ou plusieurs traitements identifiés. Cependant, les informations présentes dans l'élément `<ENTITY>` sont composites et comportent des informations non nécessaires au raisonnement DLP, comme des adresses ou des numéros de téléphone. En conséquence, c'est un identifiant unique qui est produit pour chaque entité, représentant un agent connu dans le système ou bien faisant référence à une information détaillée sur l'entité en question dans la base de croyances de l'agent PAw. De plus, comme les politiques P3P ne permettent pas d'identifier une entité de contact différente de l'entité responsable du traitement, c'est l'élément `<ENTITY>` qui permet également de générer les prédicats *contact* et *informContact* (figure B.2)[2].

```
<ENTITY>
  <EXTENSION>
    <p3p11:datagroup>
      <p3p11:datatype>
        <p3p11:business>
          <p3p11:orgname>
            serviceAgent
          </p3p11:orgname>
        </p3p11:business>
      </p3p11:datatype>
    </p3p11:datagroup>
  </EXTENSION>
  <DATA-GROUP>
    <DATA ref="#business.name">
      serviceAgent
    </DATA>
  </DATA-GROUP>
</ENTITY>
```

$\Longrightarrow$

$$\begin{aligned}&\mathit{responsible}(\texttt{serviceAgent}, \texttt{service})\\&\mathit{contact}(\texttt{serviceAgent}, \texttt{service})\\&\mathit{informContact}(\texttt{serviceAgent},\\&\quad \texttt{userAgent}, \texttt{service}, \texttt{serviceAgent})\end{aligned}$$

FIG. B.2 – Exemple de réécriture d'un élément `<ENTITY>`

B.1.3 `<STATEMENT>`

Une politique P3P présente un élément `<STATEMENT>` pour chaque traitement mettant en jeu des données personnelles. Ainsi, pour chaque élément `<STATEMENT>` rencontré, un nouvel ID de traitement sera produit, afin de pouvoir y associer divers prédicats dans $\mathcal{L}_{DLP}$.

[2] Dans la figure B.2, nous avons représenté une portion de politique compatible à la fois avec les standards 1.0 et 1.1 de P3P, d'où la redondance des informations. Cette redondance, nécessaire à la compatibilité descendante du format, est imposée par le schéma normatif des politiques P3P 1.1. Par la suite, nous ne mentionnerons plus que les formulations les plus modernes, les expressions datant du standard 1.0 étant habituellement produites automatiquement.

B.1.4 <PURPOSE> et <PPURPOSE>

L'élément <PURPOSE> est un fils de l'élément <STATEMENT>, identifiant la finalité du traitement. Un ensemble fini de valeurs est proposé par le W3C afin de décrire la nature du traitement. Dans la version 1.1 de P3P, l'expressivité est améliorée avec un nouvel ensemble de valeurs correspondant à un élément d'extension <PPURPOSE>. Lors du passage à $\mathcal{L}_{DLP}$, chaque valeur trouvée dans l'un de ces éléments déclenche la production d'un prédicat d'état *actionType*. L'ensemble des instances du prédicat permet en fait de décrire à la fois le type du traitement et sa finalité, les deux notions étant pas parfois confondues dans les scénarios d'utilisation comme dans le langage P3P. Le fait que cette information soit dans une politique publiée permet également de générer un ou plusieurs prédicats performatifs *informActionType*. La figure B.3 présente un exemple de réécriture d'un élément <PURPOSE> concernant un service destiné à la reprise de contact avec l'utilisateur dans le futur, au télémarketing, à la personnalisation et au retour d'information vers l'utilisateur.

```
<PURPOSE>
  <contact />
  <telemarketing />
    <EXTENSION>
      <PPURPOSE>
        <custom />
        <feedback />
      </PPURPOSE>
    </EXTENSION>
</PURPOSE>
```

$\Longrightarrow$

$$
\begin{array}{l}
actionType(\texttt{service}, \texttt{contact}) \\
actionType(\texttt{service}, \texttt{telemarketing}) \\
actionType(\texttt{service}, \texttt{custom}) \\
actionType(\texttt{service}, \texttt{feedback}) \\
informActionType(\texttt{serviceAgent}, \texttt{userAgent}, \\
\quad \texttt{service}, \texttt{contact}) \\
informActionType(\texttt{serviceAgent}, \texttt{userAgent}, \\
\quad \texttt{service}, \texttt{telemarketing}) \\
informActionType(\texttt{serviceAgent}, \texttt{userAgent}, \\
\quad \texttt{service}, \texttt{custom}) \\
informActionType(\texttt{serviceAgent}, \texttt{userAgent}, \\
\quad \texttt{service}, \texttt{feedback})
\end{array}
$$

FIG. B.3 – Exemple de réécriture d'éléments <PURPOSE> et <PPURPOSE>

B.1.5 <DATA-GROUP>

L'élément <DATA-GROUP> décrit les données collectées par le site web, à l'aide de l'extension <datatype>. Cette extension représente le type des données suivant un arbre XML [Wor06, 5.5], chaque site web pouvant étendre le schéma de données proposé par un schéma personnalisé. Un élément <CATEGORIES> peut également être associé à une feuille d'un arbre de type, afin de préciser une ou plusieurs catégories parmi une liste finie (contenant par exemple `financial` ou `health`). Comme dans le cas du prédicat *actionType*, chaque élément fils de l'élément <CATEGORIES> ainsi que chaque feuille dans l'arborescence d'une éventuelle extension <datatype> déclenche la production d'une instance de prédicat d'état *dataType* et d'un prédicat performatif *request*. La figure B.4 présente un exemple d'une telle traduction dans le cas d'une politique réclamant une adresse IP, comportant des informations de localisations géographiques, ainsi qu'un numéro de téléphone personnel.

B.1.6 <RETENTION>

L'élément <RETENTION> a pour vocation d'annoncer à l'utilisateur, pour chaque traitement, la durée de conservation des informations collectées. Cet élément en particulier pose problème,

```
<DATA-GROUP>
  <EXTENSION>
    <datatype>
      <user>
        <home-info>
          <telecom>
            <telephone />
          </telecom>
        </home-info>
      </user>
      <dynamic>
        <clientip>
          <CATEGORIES>
            <location />
          </CATEGORIES>
        </clientip>
      </dynamic>
    </datatype>
  </EXTENSION>
</DATA-GROUP>
```

$\Longrightarrow$

$$
\begin{array}{l}
\mathit{request}(\texttt{serviceAgent}, \texttt{userAgent}, \texttt{service}, \\
\quad \texttt{user.home-info.telecom.telephone}, \texttt{data1}) \\
\mathit{request}(\texttt{serviceAgent}, \texttt{userAgent}, \texttt{service}, \\
\quad \texttt{dynamic.clientip}, \texttt{data2}) \\
\mathit{request}(\texttt{serviceAgent}, \texttt{userAgent}, \texttt{service}, \\
\quad \texttt{location}, \texttt{data2}) \\
\mathit{dataType}(\texttt{service}, \texttt{data1}, \\
\quad \texttt{user.home-info.telecom.telephone}) \\
\mathit{dataType}(\texttt{service}, \texttt{data2}, \texttt{dynamic.clientip}) \\
\mathit{dataType}(\texttt{service}, \texttt{data2}, \texttt{location})
\end{array}
$$

FIG. B.4 – Exemple de réécriture d'un élément `<DATA-GROUP>`

c'est l'un des points sur lesquels le langage P3P manque d'expressivité. En effet, il est impossible de donner des durées de conservation explicites, les valeurs sont prises dans un ensemble fini comprenant les possibilités suivantes :

- Aucune conservation ;
- Conservation telle que requise par la loi ;
- Conservation « pour la finalité déclarée » (*stated purpose*) ;
- Conservation conforme à une politique d'entreprise précisée par ailleurs ;
- Conservation pour une durée indéterminée.

Un utilisateur soucieux de la protection de ses données ne pourrait être qu'inquiété par les choix proposés. D'autre part, la durée de conservation s'applique à l'ensemble des données du traitement, il n'est pas possible de travailler avec des rétentions d'informations différenciées. Les motivations qui nous ont amenés à proposer la logique DLP nous ont au contraire poussés à représenter les durées explicitement, et ce pour chaque information de chaque traitement. Lors de l'extraction d'informations à partir d'une politique DLP, il est donc évident que les durées de rétention dans les prédicats performatifs *informDuration* et les prédicats d'état *duration* produits alors seront identiques pour l'ensemble d'un traitement. Il reste à préciser ces durées en fonction des options proposées par P3P :

- En l'absence de conservation, on prendra tout naturellement une durée de `0` ;
- Dans les cas de la « finalité déclarée » et de la durée requise par la loi, on s'appuiera sur un raisonnement DLP pour calculer la durée maximale que pourrait s'autoriser l'agent PAw s'il devait mettre en place ce traitement, au vu des autorités normatives auxquelles il est soumis. En l'absence de réglementation couvrant le cas courant, on se placera dans le cas

de la durée de rétention indéterminée[3] ;
- Dans le cas de la politique d'entreprise, si cette politique est connue en tant qu'autorité normative alors on se référera à ses normes pour déterminer la durée de rétention maximale autorisée. Dans le cas contraire ou en cas d'incomplétude de la politique en question, on se placera dans le cas de la durée de rétention indéterminée ;
- Dans le cas de la durée de rétention indéterminée, on se heurte à une limite de l'expressivité du langage DLP. En effet, la durée de rétention doit être un entier. L'agent PAw utilisera alors un entier représentant une durée de rétention au regard de laquelle la persistance des intérêts d'un individu ou d'une organisation sont négligeables, dix mille ans par exemple. Le choix d'un tel entier garantit que la durée de rétention ainsi représentée entrera en conflit avec toute norme ayant pour but de limiter explicitement la conservation des informations.

La figure B.5 donne des exemples de réécriture en se plaçant dans le pire cas (lorsque l'on ne dispose d'aucune information par ailleurs).

```
<RETENTION>
  <no-retention />
</RETENTION>
```

$\Longrightarrow$ *informDuration*(`serviceAgent`, `userAgent`, `service`, AnyData, `0`) *duration*(`service`, AnyData, `0`)

```
<RETENTION>
  <stated-purpose />
</RETENTION>

<RETENTION>
  <legal-requirement />
</RETENTION>

<RETENTION>
  <business-practices />
</RETENTION>

<RETENTION>
  <indefinitely />
</RETENTION>
```

$\Longrightarrow$ *informDuration*(`serviceAgent`, `userAgent`, `service`, AnyData, `10000*365*24`) *duration*(`service`, AnyData, `10000*365*24`)

FIG. B.5 – Exemples de réécriture d'un élément `<RETENTION>`

B.1.7 `<RECIPIENT>`

L'élément `<RECIPIENT>`, qui déclare à qui pourront être transmises les informations collectées, est également délicat à traiter (de par son manque d'expressivité) pour la production des prédicats *informForward* et *forwardList*. Les valeurs suivantes peuvent être utilisées dans une politique P3P :
- `ours` : la transmission est limitée à l'entité responsable du traitement, aucun prédicat *forward* n'est donc produit et le prédicat *informForward* utilisera une liste vide.

[3] En d'autres termes, les réglementations connues par l'agent PAw sont incomplètes au sens de Cholvy et Roussel [CR08] et il lui faut mettre en place une stratégie de complétion en se plaçant dans le pire cas.

- `public` : les données sont totalement publiques, n'importe quel agent peut y avoir accès. C'est donc un schéma d'agent (équivalent d'une variable non instanciée) qui est utilisé dans les prédicats *informForward* et *forwardList* produits.
- `other-recipient` : la transmission concerne des agents ne respectant pas les mêmes politiques de traitement des données personnelles. En l'absence d'information utilisable, on se placera dans le même cas que pour l'option `public`.
- `unrelated` : la transmission concerne des agents dont les politiques de traitement des données personnelles ne sont pas connues du responsable du traitement courant. En l'absence d'information utilisable, on se placera dans le même cas que pour l'option `public`.
- `delivery` : la transmission concerne les agents chargés de la livraison. Si ce n'est pas déjà le cas, des prédicats *actionType* et *informActionType* sont produits avec le type d'action correspondant. Concernant les prédicats de transmission des données, un nouvel identifiant d'agent est produit pour représenter l'entité de livraison (si elle n'est pas déjà associée à un agent). On notera que du côté de l'agent utilisateur, aucun nouveau service n'est créé pour la livraison, c'est le service initial qui comprend maintenant un type `delivery`.
- `same` : la transmission concerne n'importe quel agent respectant les mêmes politiques de traitement des données personnelles. En dépit de l'insuffisance qui peut transparaître de cette option, fondée sur une confiance absolue de l'utilisateur en des déclarations vagues, nous prenons ici le parti de suivre à la lettre la sémantique de cet élément. L'ensemble des formules décrivant la politique de l'entité responsable du traitement est donc utilisée pour caractériser un schéma d'agent « SimilarAgent », qui sera le récipiendaire identifié dans les prédicats de transmission des données.

La version 1.1 des spécifications de P3P introduit une extension `<JURISDICTION>`, surtout destinée à fournir des informations à destination de lecteurs humains sur le cadre légal de référence, mais pouvant également contenir une URI analysable par une machine. Dans ce dernier cas, si l'URI fait référence à une autorité normative identifiable par l'agent PAw, celui-ci peut se servir des normes correspondantes pour enrichir les formules produites et éviter de se placer dans le pire cas. La figure B.6 présente quelques exemples de productions de prédicats à partir d'éléments `<RECIPIENT>` relatifs à une donnée `data1`, toujours en se plaçant dans le pire cas.

La production des formules dans le cas d'un récipiendaire de type `same` est beaucoup plus complexe. Tout d'abord, il faut générer un ensemble de formules que nous symboliserons par le prédicat $matches(\mathrm{A1}, \mathrm{T1}, \mathrm{D1}, \mathrm{A2}, \mathrm{T2}, \mathrm{D2})$ (B.1). Cette conjonction de prédicats $\mathcal{L}_{DLP}$ devient vraie si et seulement si les agents A1 et A2 sont responsables respectivement pour des traitements T1 et T2 de même type, mettant en jeu des données D1 et D2 de même type. On peut ensuite définir un prédicat $similarPolicies(\mathrm{A1}, \mathrm{A2})$ (B.2), vrai si et seulement si les deux agents ont les mêmes politiques connues en matière de conservation et de retransmission des données dans des situations comparables. Ceci établi, la figure B.7 détaille la réécriture d'un élément `<RECIPIENT>` en une formule $\mathcal{L}_{DLP}$.

$$
\begin{aligned}
&matches(\mathrm{A1}, \mathrm{T1}, \mathrm{D1}, \mathrm{A2}, \mathrm{T2}, \mathrm{D2}) \\
&\stackrel{def}{=} \begin{cases} responsible(\mathrm{A1}, \mathrm{T1}) \wedge responsible(\mathrm{A2}, \mathrm{T2}) \\ actionType(\mathrm{T1}, \mathrm{ActionType1}) \leftrightarrow actionType(\mathrm{A2}, \mathrm{ActionType2}) \\ dataType(\mathrm{T1}, \mathrm{D1}, \mathrm{DataType1}) \leftrightarrow dataType(\mathrm{T2}, \mathrm{D2}, \mathrm{DataType2}) \end{cases}
\end{aligned} \tag{B.1}
$$

```
<RECIPIENT>
  <ours />
</RECIPIENT>
```

$\Longrightarrow$ *informForward*(`serviceAgent`, `userAgent`, `service`, data1, [])

```
<RECIPIENT>
  <public />
</RECIPIENT>

<RECIPIENT>
  <other-recipient />
</RECIPIENT>

<RECIPIENT>
  <unrelated />
</RECIPIENT>
```

$\Longrightarrow$ *informForward*(`serviceAgent`, `userAgent`, `service`, data1, [AnyAgent])
forwardList(`service`, `data1`, AnyAgent)

```
<RECIPIENT>
  <delivery />
</RECIPIENT>
```

$\Longrightarrow$ *informActionType*(`serviceAgent`, `userAgent`, `service`, `delivery`)
actionType(`service`, delivery)
informForward(`serviceAgent`, `userAgent`, `service`, `data1`, [`deliveryAgent`])
forwardList(`service`, `data1`, `deliveryAgent`)

Fig. B.6 – Exemples simples de réécriture d'un élément `<RECIPIENT>`

$$\begin{aligned} &\mathit{similarPolicies}(\mathrm{A1}, \mathrm{A2}) \stackrel{def}{=} \\ &\mathit{matches}(\mathrm{A1}, \mathrm{T1}, \mathrm{D1}, \mathrm{A2}, \mathrm{T2}, \mathrm{T2}) \\ &\quad \rightarrow \begin{cases} \mathit{forwardList}(\mathrm{T1}, \mathrm{D1}, \mathrm{SomeAgent}) \leftrightarrow \mathit{forwardList}(\mathrm{T2}, \mathrm{D2}, \mathrm{SomeAgent}) \\ \mathit{duration}(\mathrm{T1}, \mathrm{D1}, \mathrm{Duration}) \leftrightarrow \mathit{duration}(\mathrm{T2}, \mathrm{D2}, \mathrm{Duration}) \end{cases} \end{aligned} \tag{B.2}$$

```
<RECIPIENT>
  <same />
</RECIPIENT>
```

$\Longrightarrow$ *informForward*(`serviceAgent`, `userAgent`, `service`, `data1`, [SimilarAgent])
similarPolicies(`serviceAgent`, SimilarAgent)
forwardList(`service`, `data1`, SimilarAgent)

Fig. B.7 – Exemple plus complexe de réécriture d'un élément `<RECIPIENT>`

B.2 Exemples d'éléments P3P ignorés à l'extraction

B.2.1 `<DISPUTES>`

Une politique contient un ou plusieurs éléments `<DISPUTES>` (fils d'un élément `<DISPUTES-GROUP>` qui décrivent (de manière limitée pour la partie analysable par une machine) les possibilités de résolution de contentieux liés aux pratiques de l'entité responsable du

traitement. Cette partie de la politique est principalement destinée à fournir des informations aux utilisateurs humains et nécessite des traitements que la communauté de l'informatique légale s'accorde à réserver à des juristes humains. Les éléments fils de `<DISPUTES>`, à savoir par exemple `<REMEDIES>` ou `<CONSEQUENCE>`, sont ignorés de la même manière.

B.2.2 `<NON-IDENTIFIABLE>`

L'élément `<NON-IDENTIFIABLE>` indique que le traitement en question ne collecte ni ne traite d'informations personnelles. Il ne concerne donc pas notre champ de travail et n'aura pas d'influence sur les formules DLP produites.

B.3 Synthèse des prédicats de $\mathcal{L}_{DLP}$ pouvant être produites lors de l'analyse d'une politique P3P

Élément P3P	**Prédicat $\mathcal{L}_{DLP}$ produit**
`<POLICY>`	*consentRequest*
`<ENTITY>`	*informContact*, *contact*, *responsible*
`<STATEMENT>`	$\emptyset$
`<PURPOSE>` `<PPURPOSE>`	*informActionType*, *actionType*
`<DATA-GROUP>`	*request*, *dataType*
`<RETENTION>`	*informDuration*, *duration*
`<RECIPIENT>`	*informForward*, *forwardList*, *actionType*, *informActionType*
$\emptyset$	*consent*, *tell*, *updateRequest*, *forgetRequest* *perform*, *update*, *forget*, *forward* *owner*

TAB. B.1 – Correspondances entre éléments P3P et prédicats de $\mathcal{L}_{DLP}$

Le tableau B.1 résume quels prédicats $\mathcal{L}_{DLP}$ peuvent être produits à partir de quels éléments P3P.

- *consent*, *tell*, *updateRequest* et *forgetRequest* ne sont pas produits par l'analyse d'une politique P3P puisqu'ils font référence à des actions de l'utilisateur ;
- *perform*, *update*, *forget* et *forward* ne sont pas produits par l'analyse d'une politique P3P puisqu'ils font référence à des action indépendantes de l'existence de la politique ;
- *owner* n'est pas produit par l'analyse d'une politique P3P puisqu'il dépend de la mise en place d'un traitement qui n'est pas une conséquence de la politique.

Annexe C

Positionnement dans le paradigme VOYELLES

L'étude de la protection des données personnelles dans le cadre des systèmes multi-agents nous a amené à aborder, outre le raisonnement intra-agent, les concepts spécifiques d'interaction et d'organisation, ainsi que la place de l'utilisateur humain. Dans notre travail, les liens qui peuvent exister entre ces différents concepts, ainsi que leurs importances relatives, sont vitaux. À ce titre, il nous paraît intéressant d'observer comment notre réflexion et nos propositions peuvent être considérés au travers des principes de VOYELLES.

La méthodologie du paradigme VOYELLES [Dem96] se fonde sur les relations possibles entre les cinq dimensions majeures des systèmes multi-agents que sont les agents eux-mêmes (A), l'environnement (E), l'interaction (I) et l'organisation (O). Un système multi-agent est un agencement particulier de ces dimensions. Suivant le domaine d'application ou le type de problème, certaines de ces dimensions peuvent avoir une place plus centrale, ou bien des relations deux à deux peuvent être mises en avant. Si l'on prend l'exemple de la robotique, le postulat de départ est sans doute l'existence d'un robot, ou agent matériel (A). La deuxième étape à prendre en compte est l'environnement dans lequel cet agent évolue (A + E) et avec lequel il va interférer par perception ou actuation. Le système se complexifie à l'introduction d'un deuxième robot, l'interaction entre les deux machines devant être explicitement prise en compte ((A + E) + I). Si plus de deux robots sont présents, on peut parler de système multi-robot et l'on peut s'intéresser aux organisations préétablies ou émergentes qui régissent l'ensemble de leurs interactions pair à pair au sein de cette société artificielle ((A + E) + I) + O). Malgré la difficulté qu'il y a à gérer des systèmes multi-robots, on peut également envisager des applications davantage orientées vers ces interactions que vers les influences réciproques entre robots et environnement ((A + I) + O) + E). VOYELLES fournit ainsi pour le moins une manière de caractériser un système multi-agent en fonction de ses priorités.

Si nous souhaitons situer nos travaux par le biais d'une description de type VOYELLES, il nous faut absolument représenter l'utilisateur (U), qui constitue l'extension la plus récente du paradigme (permettant ainsi la conception de systèmes multi-agents hybrides, comprenant à la fois agents artificiels et agents humains). Pour une application de protection des données personnelles, et plus généralement de la sphère privée, c'est en effet l'utilisateur humain qui est à la source et au centre de toutes les problématiques techniques et scientifiques. Le lien fort entre l'utilisateur et son agent personnel (U + A) est pour nous la clé qui lui permet de contrôler ses données (et donc de se protéger) lorsque les interactions (sources de risques) sont nécessaires ((U + A) + I). Les organisations viennent ensuite comme contexte de ces interactions. Elles

peuvent sous-tendre les outils permettant à l'utilisateur de se protéger, mais dans une application ouverte leur importance est minimisée par rapport aux précédents concepts car elles peuvent être dynamiques, émergentes, subies ou inexistantes (((U + A) + I) + O). L'environnement est le dernier concept à entrer en jeu. Il a pris peu de place dans nos travaux et ne représente que le reste du monde (ainsi que les supports matériels et logiciels), duquel nous avons isolé et protégé les données qui nous étaient sensibles ((((U + A) + I) + O) + E).

On peut également considérer que les architectures très fermées, comme celles fondées sur le *Trusted Computing*, mettent en œuvre une autre expression VOYELLES. En effet, dans ces structures imposées les interactions sont pleinement contraintes par une organisation préétablie et figée. L'ensemble peut donc être appréhendé comme un tout par le couple agent humain - agent artificiel : (((U + A) + (I + O)) + E).

Il nous paraît donc possible d'appréhender les problématiques liées à la vie privée et aux agents personnels de plusieurs manières différentes dans le paradigme VOYELLES, et ce uniquement à condition de prendre en compte l'utilisateur humain.

Bibliographie

[ACC+06] C. Ardagna, J. Camenisch, Marco Casassa Mont, S. Clauss, S. Crane, Yves Deswarte, G. Hogben, Siani Pearson, L. Pimenidis, T. Roessler, and D. Sommer. An architecture for privacy-enhancing identity management. LAAS report 05206, LAAS-CNRS, Toulouse, France, march 2006.

[ACI04] ACI Masses de Données. Projet APMD : Accès Personnalisé à des Masses de Données., 2004.

[AEB+03] A. Abou El Kalam, R. El Baida, Philippe Balbiani, S. Benferhat, Frédéric Cuppens, Yves Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization based access control. In *Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks (POLICY 2003)*, Lake Como, Italy, june 2003.

[AEG+06] Carole Adam, Fabrice Évrard, Benoît Gaudou, Andreas Herzig, and Dominique Longin. Modélisation logique d'agents rationnels pour l'intelligence ambiante. In *Actes des 14èmes Journées Francophones des Systèmes Multiagents (JFSMA'06)*, Annecy, France, October 2006. Hermès.

[All84] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23 :123–154, 1984.

[And03] Ross Anderson. "Trusted Computing" frequently asked questions, August 2003. http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html.

[AOS] AOS Autonomous Decision-Making Software. JACK. http ://www.aosgrp.com/products/jack/.

[Åqv04] Lennart Åqvist. Combinations of tense and deontic modalities. In Alessio Lomuscio and Donald Nute, editors, *Proceedings of the 7th International Workshop on Deontic Logic in Computer Science (DEON 2004)*, volume 3065 of *LNCS*, pages 3–28, Madeira, Portugal, May 2004. Springer.

[Art04] Article 29 Data Protection Working Party. Working document on trusted computing platforms and in particular on the work done by the trusted computing group (TCG group). Working Document of the European Commission, January 2004. 11816/03/EN – WP 86.

[AS99] Giuseppe Amato and Umberto Straccia. User profile modeling and applications to digital libraries. In S. Abiteboul and A.-M. Vercoustre, editors, *Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries (ECDL'99)*, number 1696 in LNCS, pages 184–197. Springer-Verlag, 1999.

[ÅvdHR+07] Thomas Ågotnes, Wiebe van der Hoek, Juan A. Rodríguez-Aguilar, Carles Sierra, and Michael Wooldridge. On the logic of normative systems. In *Proceedings*

of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07), Hyderabad, India, January 2007.

[Baa03] Sara Baase. *A Gift of Fire : Social, Legal, and Ethical Issues in Computing.* Prenctice-Hall, 2003.

[Bal05] Philippe Balbiani. Constitution et développement d'une logique des modalités aléthiques, déontiques, dynamiques, et temporelles en vue de la formalisation du raisonnement sur les actions et sur les normes. In Andreas Herzig, Yves Lespérance, and Abdel-Illah Mouaddib, editors, *Troisièmes journées francophones sur les modèles formels de l'interaction (MFI'05)*, Caen, France, 2005. Cépaduès éditions.

[BAN89] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. In *Proceedings of the Royal Society of London, Series A, 426*, pages 233–271, London, UK, 1989.

[BBF06] Julien Brunel, Jean-Paul Bodeveix, and Mamoun Filali. A state/event temporal deontic logic. In *Eighth International Workshop on Deontic Logic in Computer Science (DEON'06)*, number 4048 in LNCS, 2006.

[BCC04] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and communications security (CCS'04)*, pages 132–145, Washington DC, USA, 2004. ACM Press.

[BDH+01] Jan Broersen, Mehdi Dastani, Joris Hulstijn, Zisheng Huang, and Leendert van der Torre. The BOID architecture : conflicts between beliefs, obligations, intentions and desires. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 9–16, Montreal, Canada, 2001. ACM Press.

[BdV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic.* Cambridge University Press, 2001.

[BF01] Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. In *Crypto 2001*, volume 2139 of *LNCS*, pages 213–229, Santa Barbara, California, USA, august 2001. Springer-Verlag.

[BH06] Rafael H. Bordini and Jomi F. Hübner. *Jason : A Java-based AgentSpeak interpreter used with SACI for multi-agent distribution over the net*, March 2006. Release version 0.8.

[BHMW06] Axel Bürkle, Alice Hertel, Wilmuth Müller, and Martin Wieser. Evaluating mobile agent platform security. In Klaus Fischer, Ingo J. Timm, Elisabeth André, and Ning Zhong, editors, *Proceedings of the 4th German Conference on Multiagent System Technologies (MATES 2006)*, volume 4196 of *LNCS*, pages 159–171, Erfurt, Germany, September 2006. Springer.

[BK05] Mokrane Bouzeghoub and Dimitre Kostadinov. Personnalisation de l'information : aperçu de l'état de l'art et définition d'un modèle flexible de profils. In *Conférence en Recherche d'Information (CORIA)*, pages 201–218, 2005.

[BL76] David Eliott Bell and Len J. LaPadula. Secure computer systems : unified exposition and multics interpretation. Technical Report MTR 2997 rev. 1, MITRE corporation, Bedford, Massachusetts, USA, 1976.

[Bor00] John J. Borking. Privacy incorporated software agent (pisa) : proposal for building a privacy guardian for the electronic age. In *International Workshop on Design*

Issues in Anonymity and Unobservability, volume 2009/2001 of *LNCS*, pages 130–140, Berkeley, California, USA, july 2000. Springer Verlag.

[BP88] Nuel Belnap and Michael Perloff. Seeing to it that : a canonical form for agentives. *Theoria*, 54 :175–199, 1988.

[Bra87] Michael E. Bratman. *Intentions, plans and practical reason.* Harvard University Press, Cambridge, Massachusetts, USA, 1987.

[BvV06] Guido Boella, Leendert van der Torre, and Harko Verhagen. Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 12(2-3) :71–79, 2006.

[Bé02] Jean-Yves Béziau. S5 is a paraconsistent logic and so is first-order classical logic. *Logical Studies*, 9, 2002.

[Bé06] Jean-Yves Béziau. Paraconsistent logic ! (a reply to slater). In Lorenzo Peña, editor, *SORITES, An International Digital Journal of Analytical Philosophy*, number 17, october 2006.

[CA07] Richard Cissée and Sahin Albayrak. An agent-based approach for privacy-preserving recommender systems. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'07)*, pages 1–8, Honolulu, Hawaii, 2007. ACM.

[Cas04] Cristiano Castelfranchi. Trust mediation in knowledge management and sharing. In *Proceedings of the Second International Conference on Trust Management (iTrust 2004)*, pages 304–318, Oxford, UK, 2004.

[CC97] Laurence Cholvy and Frédéric Cuppens. Analyzing consistency of security policies. In *Proceedings of the 18th IEEE Symposium on Research in Security and Privacy*, Oakland, CA, USA, May 1997.

[CC98] Laurence Cholvy and Frédéric Cuppens. Reasoning about norms provided by conflicting regulations. In Henry Prakken and Paul McNamara, editors, *Norms, Logics and Information Systems : New Studies in Deontic Logic and Computer Science*, pages 247–264, Amsterdam, the Netherlands, 1998. IOS Press.

[CDA99] Ibrahim Cingil, Asuman Dogac, and Ayca Azgin. A broader approach to personalization. Technical report, SRDC, Middle East Technical University, July 1999.

[CF98] Cristiano Castelfranchi and Rino Falcone. Principles of trust for multi-agent systems : Cognitive anatomy, social importance, and quantification. In *Proceedings of the Third International Conference of Multi-Agent Systems (ICMAS'98)*, pages 72–79, Paris, july 1998.

[Che80] Brian F. Chellas. *Modal Logic, an Introduction.* Cambridge University Press, 1980.

[Chi63] Roderick M. Chisholm. Contrary-to-duties imperatives and deontic logic. *Analysis*, 24 :33–36, 1963.

[Chi04] Guillaume Chicoisne. *Dialogue entre agents naturels et agents artificiels. Une application aux communautés virtuelles.* PhD thesis, Institut National Polytechnique de Grenoble, 2004.

[Cho99] Laurence Cholvy. Checking regulation consistency by using SOL-resolution. In *International Conference on Artificial Intelligence and Law*, pages 73–79, 1999.

[CJ94] José Carmo and Andrew J. I. Jones. Deontic database constraints and the characterisation of recovery. In *Second international workshop on deontic logic in computer science (DEON'94)*, pages 56–85, Amsterdam, The Netherlands, 1994.

[CL90] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3) :213–261, 1990.

[CNV97] Michael A. Covington, Donald Nute, and André Vellino. *Prolog Programming in Depth*. Prenctice Hall, Upper Saddle River, NJ, USA, 1997.

[CPB03] Marco Casassa Mont, Siani Pearson, and Pete Bramhall. Towards accountable management of identity and privacy : Sticky policies and enforceable tracing services. HP Laboratories Bristol, March 2003. HPL-2003-49.

[CPBD07] Ludivine Crépin, Guillaume Piolle, Olivier Boissier, and Yves Demazeau. Des systèmes normatifs comme outils de protection de la vie privée. In *Intelligence Artificielle et Web Intelligence (IAWI'07)*, Grenoble, France, July 2007. AFIA.

[CR08] Laurence Cholvy and Stéphanie Roussel. Raisonner avec une réglementation incomplète : le cas d'une politique d'échange d'informations. In *16e congrès francophone AFRIF-AFIA sur la Reconnaissance de Formes et l'Intelligence Artificielle (RFIA'08)*, Amiens, France, January 2008. AFRIF-AFIA.

[Cup93a] Frédéric Cuppens. A logical analysis of authorized and prohibited information flows. In *IEEE Symposium on Research in Security and Privacy*, pages 100–109, Oakland, California, USA, 1993. IEEE Computer Society Press.

[Cup93b] Frédéric Cuppens. A logical formalization of secrecy. In *Computer Security Foundations Workshop VI*, pages 53–62, Franconia, USA, 1993. IEEE Computer Society Press.

[CVJ+08] Ludivine Crépin, Laurent Vercouter, François Jacquenet, Yves Demazeau, and Olivier Boissier. Hippocratic multi-agent systems. In *Proceedings of the International Conference on Enterprise Information Systems (ICEIS'08)*, pages 301–307, 2008.

[DA06a] Yves Deswarte and Carlos Aguilar-Melchor. Current and future privacy enhancing technologies for the internet. *Annals of Telecommunications*, 61(3-4) :399–417, 2006.

[DA06b] Yves Deswarte and Carlos Aguilar Melchor. *Sécurité des Systèmes d'Information*, chapter Technologies de Protection de la Vie Privée sur Internet, pages 49–71. Hermès, Paris, France, 2006.

[DBDM04] Frank Dignum, Jan Broersen, Virginia Dignum, and John-Jules Meyer. Meeting the deadline : Why, when and how. In Michael G. Hinchey, James L. Rash, Walt Truszkowski, and Christopher Rouff, editors, *Third International Workshop on Formal Approaches to Agent-Based Systems (FAABS'04)*, number 3228 in LNCS, pages 30–40, Greenbelt, MD, USA, april 2004. Springer Verlag.

[DDM03] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion : Design of a type iii anonymous remailer protocol. In *2003 IEEE Symposium on Security and Privacy*, pages 2–15, Oakland, CA, USA, 2003. IEEE.

[Dem96] Yves Demazeau. Vowels. In *1st Ibero-American Workshop on Distributed AI and Multi-Agent Systems (IWDAIMAS'96)*, Mexico, 1996.

[Dem04] Robert Demolombe. Reasoning about trust : a formal logical framework. In *2nd international conference on trust management (iTrust'04)*, pages 291–303, 2004.

[Dem05] Robert Demolombe. Formalisation de l'obligation de faire avec délais. In *Troisièmes journées francophones des modèles formels de l'interaction (MFI'05)*, Caen, France, may 2005.

[dEW06] Marie desJardins, Eric Eaton, and Kiri L. Wagstaff. Learning user preferences for sets of objects. In *ICML '06 : Proceedings of the 23rd international conference on Machine learning*, pages 273–280, Pittsburgh, Pennsylvania, 2006. ACM Press.

[dKLW97] Mark d'Inverno, David Kinny, Michael Luck, and Michael Wooldridge. A formal specification of dMars. In *Intelligent Agents IV : Proceedings of the Fourth International Workshop on Agent Theories, Architectures and Languages (ATAL'97)*, number 1365 in LNAI. Springer-Verlag, 1997.

[DM02] Irène Degirmenciyan and Frédéric Marc. Des objets aux agents : JACK et ses applications en aéronautique. In *OFTA - groupe Systèmes Multi-Agents*, June 2002.

[DRB+03] Ian Dickinson, Dave Reynolds, Dave Banks, Steve Cayzer, and Poorvi L. Vora. User profiling with privacy : A framework for adaptive information agents. In Matthias Klusch, Sonia Bergamaschi, Peter Edwards, and Paolo Petta, editors, *Intelligent Information Agents - The AgentLink Perspective*, LNCS 2586, pages 123–151. Springer, 2003.

[dW05] Marie desJardins and Kiri L. Wagstaff. DD-PREF : a language for expressing preferences over sets. In *Proceedings of the 20th National Conference on Artificial Intelligence*, JPL TRS 1992+, Pittsburgh, Pennsylvania, July 2005. Jet Propulsion Laboratory, National Aeronautics and Space Administration.

[ERAS+01] Marc Esteva, Juan Antonio Rodríguez-Aguilar, Carles Sierra, Pere Garcia, and Josep Lluís Arcos. On the formal specification of electronic institutions. In Frank Dignum and Carles Sierra, editors, *Agent-mediated Electronic Commerce (The European AgentLink Perspective).*, volume 1991 of *LNCS*, pages 126–147, London, UK, 2001. Springer Verlag.

[ERASV04] Marc Esteva, Juan Antonio Rodríguez-Aguilar, Carles Sierra, and Wamberto Vasconcelos. Verifying norm consistency in electronic institutions. In *Proceedings of the AAAI-04 Workshop on Agent Organizations : Theory and Practice*, pages 8–14, San José, California, USA, 2004. Technical Report WS-04-02.

[ERRAA04] Marc Esteva, Bruno Rosell, Juan Antonio Rodríguez-Aguilar, and Josep Lluís Arcos. Ameli : An agent-based middleware for electronic institutions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 236–243, New York, USA, July 19-23 2004.

[Eur08] European Union's Sixth Framework Programme. PRIME - Privacy and Identity Management for Europe. IST-507591, 2004–2008. https://www.prime-project.eu/.

[EVSRA04] Marc Esteva, Wamberto Vasconcelos, Carles Sierra, and Juan Antonio Rodríguez-Aguilar. Norm consistency in electronic institutions. In *Proceedings of the XVII Brazilian Symposium on Artificial Intelligence (SBIA'04)*, volume 3171 of *LNAI*, pages 494–505. Springer Verlag, 2004.

[Fer95] Jacques Ferber. *Les systèmes multi-agents, vers une intelligence collective.* InterEditions, Paris, France, 1995.

[FG98] Jacques Ferber and Olivier Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In Yves Demazeau, editor, *Third International Conference on Multiagent Systems (ICMAS'98)*, Paris, France, july 1998. IEEE Computer Society.

[FK92] David Ferraiolo and Richard Kuhn. Role-based access controls. In *Proceedings of 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[FM02] Michael J. Freedman and Robert Morris. Tarzan : A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*, pages 193–206, Washington DC, USA, November 2002.

[Gal90] Julia Rose Galliers. The positive role of conflict in cooperative multi-agents systems. In Yves Demazeau and Jean-Pierre Müller, editors, *Decentralized A.I., Proceedings of the first international workshop on Modeling Agents in a Multi-Agent World (MAMAW'89)*, pages 33–46, Amsterdam, 1990. Elsevier Science Publ.

[GCNRA05] Andrés García-Camino, Pablo Noriega, and Juan Antonio Rodríguez-Aguilar. Implementing norms in electronic institutions. In *Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*, pages 667–673, Utrecht, The Nederlands, July 2005.

[GL87] Michael P. Georgeff and Amy L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI'87)*, Seattle, WA, USA, 1987.

[GMP90] Janice I. Glasgow, Glenn H. MacEwen, and Prakash Panangaden. A logic for reasoning about security. In *Computer security foundations workshop*, pages 2–13, Franconia, USA, 1990. IEEE Computer Society Press.

[GS02] Tyron Grandison and Morris Sloman. Specifying and analysing trust for internet applications. In *Proceedings of the Second IFIP Conference on e-Commerce, e-Business and e-Government*, 2002.

[GSCM07] Susan Gauch, Mirco Speretta, Aravind Chandramouli, and Alessandro Micarelli. User profiles for personalized information access. *The Adaptive Web, LNCS*, 4321 :54–89, May 2007.

[Gut01] Olivier Gutknecht. *Propositions d'un modèle organisationnel générique de systèmes multi-agents*. PhD thesis, Université de Montpellier II, Montpellier, France, 2001.

[Han02] Sven Ove Hansson. Semantics for more plausible deontic logics. In *DEON'02*, London, UK, May 2002.

[HaSB02] Jomi Fred Hübner, Jaime Sim ao Sichman, and Olivier Boissier. Moise+ : towards a structural, functional, and deontic model for mas organization. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems : part 1*, pages 501–502, New York, NY, USA, 2002. ACM Press.

[HB95] John F. Horty and Nuel D. Belnap, Jr. The deliberative stit : a study of action, omission, ability, and obligation. *Journal of Philosophical Logic*, 24 :583–644, 1995.

[HFL04] Yvon Haradji, Nils Ferrand, and Haijiang Li. *Systèmes Multi-Agents*, chapter Relations à l'Utilisateur et Nouveaux Usages, pages 215–260. Observatoire Français des Techniques Avancées, 2004.

[Hin62] Jaako Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

[HL04] Andreas Herzig and Dominique Longin. C&L intention revisited. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR2004)*, pages 527–535, Whistler, Canada, June 2004. AAAI Press.

[Hor89] John F. Horty. An alternative stit operator. Technical report, Philosophy Department, University of Maryland, 1989.

[Ide] Identity 2.0. http://www.identity20.com/.

[IK05] Yannis Ioannidis and Georgia Koutrika. Personalized systems : models and methods from an IR and DB perspective. In *Proceedings of the 31st international conference on Very large data bases (VLDB '05)*, pages 1365–1365, Trondheim, Norway, 2005. VLDB Endowment.

[Int] Internet Engineering Task Force. IDsec : Virtual identity on the internet. http://idsec.sourceforge.net/.

[ISO99] ISO/IEC. Information technology - security techniques - evaluation criteria for it security, part 2 : Security functional requirements. Technical Report 15408-2, International Organization for Standardization, 1999.

[JL02] Xiaodong Jiang and James A. Landay. Modeling privacy control in context-aware systems. *IEEE Pervasive Computing*, 1, issue 3 :59–63, 2002.

[JS92] Andrew J. I. Jones and Marek J. Sergot. Formal specifications of security requirements using the theory of normative positions. In Yves Deswarte, G. Eizenberg, and J.-J. Quisquater, editors, *European Symposium on Research in Computer Security (ESORICS 92)*, number 648 in LNCS, pages 103–121, Toulouse, France, November 1992. Springer-Verlag.

[JS93] Andrew J. I. Jones and Marek J. Sergot. *Deontic Logic in Computer Science : Normative System Specification.*, chapter On the Characterisation of Law and Computer Systems : The Normative Systems Perspective, pages 275–307. John Wiley and Sons, Chichester, England, 1993.

[Kan57] Stig Kanger. *Provability in Logic*. Almqvist & Wiksell, 1957.

[Kan71] Stig Kanger. *Deontic Logic*, chapter New foundations for ethical theory, pages 36–58. D .Reidel Publishing Company, Dordrecht, The Netherlands, 1971.

[Kob07] Alfred Kobsa. Privacy-enhanced personalization. *Communications of the ACM*, 50(8) :24–33, 2007.

[Kra97] Gerhard K. Kraetzschmar. *Distributed Reason Maintenance for Multiagent Systems*. PhD thesis, University of Ulm, Department of Neural Information Processing, Ulm, Germany, 1997. LNCS 1229, Springer Verlag.

[Kri59] Saul Aaron Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24 :1–14, 1959.

[Kri63a] Saul Aaron Kripke. Semantic analysis of modal logic I, normal propositional calculi. *Zeitshcrift für mathemathische Logik und Grundlagen der Mathematik*, 9 :67–96, 1963.

[Kri63b] Saul Aaron Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16 :83–94, 1963.

[KS86] Robert A. Kowalski and Marek J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4 :67–95, 1986.

[KS02] Günter Karjoth and Matthias Schunter. A privacy policy model for enterprises. In *15th IEEE Computer Security Foundations Workshop*. IEEE, IEEE Computer Society Press, 2002.

[KVGC07] Martin J. Kollingbaum, Wamberto Weber Vasconcelos, and Norman Timothy J. García-Camino, Andrés. Conflict resolution in norm-regulated environments via unification and constraints. In Matteo Baldoni, Tran Cao Son, M. Birna van Riemsdijk, and Michael Winikoff, editors, *Declarative Agent Language and Technologies 2007 (DALT'07)*, number 4897 in LNCS, Honolulu, HI, USA, May 2007. Springer.

[Lem62] E J. Lemmon. Moral dilemmas. *Philosophical Review*, 71 :139–158, 1962.

[LY02] Yuefeng Li and Yiyu Yao. User profile model : A view from artificial intelligence. In *Proceedings of the Third International Conference on Rough Sets and Current Trends in Computing (TSCTC '02)*, pages 493–496, London, UK, 2002. Springer-Verlag.

[MAG06] MAGMA recherche. Projet 2007-2010 : Systèmes multi-agents centrés utilisateurs. Technical report, Laboratoire d'Informatique de Grenoble, 2006.

[Mar] Manuel Martinez Ribas. Towards legal programming : Some legal problems of agent-based mobile commerce (legal issues of e-commerce oriented intelligent agents). European Project Grocer.

[Mar03] Manuel Martinez Ribas. Systèmes multi-agents et loi. In *OFTA - groupe Systèmes Multi-Agents*, May 2003.

[Mas] Massachusetts Institute of Technology. Kerberos : the network authentication protocol. http://web.mit.edu/Kerberos/.

[McD82] Drew V. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6 :101–155, 1982.

[MH69] John M. McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4 :463–502, 1969.

[MLd03] Miquel Montaner, Beatriz Lopez, and Josep Lluis de la Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence review*, 19(4) :285–330, June 2003.

[MM92] Philip Morris and John A. McDermid. *Database Security, VI : Status and Prospects*, chapter Formalising and Validating Complex Security Requirements, pages 113–124. Vancouver, Canada, 1992.

[MP81] Zohar Manna and Amir Pnueli. Verification of concurrent programs : temporal proof principles. In D.C. Kozen, editor, *Proceedings of the Workshop on Logics of Programs*, number 131 in LNCS, 200-252, 1981. Springer-Verlag.

[MW93] John-Jules Charles Meyer and Roel J. Wieringa. *Deontic logic in computer science : normative system specification*. John Wiley and Sons Ltd., 1993.

[Mü06] Günter Müller. Introduction of privacy and security in highly dynamic systems. *Communications of the ACM*, 49(9) :1013–1022, September 2006.

[Nat93] National Institute of Science and Technology. US secure hash algorithm 1. Federal Information Processing Standard (FIPS) 180-1, United States of America, April 1993.

[NDB05] An-Te Nguyen, Nathalie Denos, and Catherine Berrut. Cartes de communautés pour l'adaptation interactive de profils dans un système de filtrage d'information. In *Actes du XXIIIème Congrès INFORSID*, pages 253–268, Grenoble, France, May 2005.

[Nut97] Donald Nute. *Defeasible Deontic Logic : Essays in Nonmonotonic Normative Reasoning*, chapter Apparent obligation, pages 287–316. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[Nut01] Donald Nute. Defeasible logic. In *Web Knowledge Management and Decision Support : 14th International Conference on Applications of Prolog (INAP 2001)*, number 2543 in LNCS, pages 151–169, Tokyo, Japan, October 2001. Springer.

[OLMN08] Nir Oren, Michael Luck, Simon Miles, and Timothy J. Norman. An argumentation inspired heuristic for resolving normative conflict. In *Fifth Workshop on Coordination, Organizations, Institutions and Norms in Agent Systems (COIN'08)*, Estoril, Portugal, may 2008.

[Ope] Openid : an actually distributed identity system. http://openid.net/.

[Org05] Organization for the Advancement of Structured Information Standards. Privacy policy profile of XACML v2.0. Technical report, OASIS, 2005.

[Ort96] Rodolphe Ortalo. Using deontic logic for security policy specification. LAAS report 96380, LAAS-CNRS, Toulouse, France, october 1996.

[PD08a] Guillaume Piolle and Yves Demazeau. Addressing temporal aspects of privacy-related norms. In *18th European Conference on Artificial Intelligence (ECAI'08)*, Patras, Greece, July 2008. IOS Press.

[PD08b] Guillaume Piolle and Yves Demazeau. Obligations with deadlines and maintained interdictions in privacy regulation frameworks. In *Proceedings of the 8th IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'08)*, pages 162–168, Sidney, Australia, December 2008. IEEE Computer Society.

[PD08c] Guillaume Piolle and Yves Demazeau. Une logique pour raisonner sur la protection des données personnelles. In *16e congrès francophone AFRIF-AFIA sur la Reconnaissance de Formes et l'Intelligence Artificielle (RFIA'08)*, Amiens, France, January 2008. AFRIF-AFIA.

[PDC06] Guillaume Piolle, Yves Demazeau, and Jean Caelen. Privacy management in user-centred multi-agent systems. In Gregory O'Hare, Michael O'Grady, Oguz Dikenelli, and Alessandro Ricci, editors, *Proceedings of the 7th Annual International Workshop on Engineering Societies in the Agents World (ESAW'06)*, volume 4457/2007 of *LNCS*, pages 354–367, Dublin, Ireland, September 2006. Springer Verlag.

[Pea02a] Siani Pearson. Trusted agents that enhance user privacy by self-profiling. In *AAMAS-02 Workshop on Deception, Fraud and Trust in Agent Societies*, pages 113–121, Bologna, Italy, 2002.

[Pea02b] Siani Pearson. *Trusted Computing Platforms : TCPA Technology in Context*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[Pea02c] Siani Pearson. Trusted computing platforms, the next security solution. Technical Report HPL-2002-221, HP Laboratories, Bristol, UK, November 2002.

[Pea05] Siani Pearson. Trusted computing : Strengths, weaknesses and further opportunities for enhancing privacy. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *Proceedings of the Third International Conference on Trust Management (iTrust 2005)*, volume 3477 of *LNCS*, pages 305–320. Springer Verlag, 2005.

[Pio08] Guillaume Piolle. Les systèmes multi-agents normatifs : organisations, institutions et normes. In *Colloque pluridisciplinaire Ordre et Désordre*, Grenoble, France, May 2008.

[Pla] Platon. *La République*.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on the Foundations of Computer Science (FOCS-77)*, pages 46–57, Providence, Rhode Island, USA, 1977. IEEE Computer Society Press.

[Pri57] Arthur Norman Prior. *Time and modality*. Oxford Universitiy Press, 1957.

[Pri67] Arthur Norman Prior. *Past, present and future*. Oxford University Press, 1967.

[PT04] Graham Priest and Koji Tanaka. Paraconsistent logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Stanford University, 2004. http://plato.stanford.edu/entries/logic-paraconsistent/.

[Rao96] Anand S. Rao. AgentSpeak(L) : BDI agents speak out in a logical computable language. In Rudy van Hoe, editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

[RB03] Stéphanie Riché and Gavin Brebner. Storing and accessing user context. In *4th International Conference on Mobile Data Management*, pages 1–12, Melbourne, Australia, january 2003.

[RBG02] Stéphanie Riché, Gavin Brebner, and Mickey Gittler. Client-side profile storage. In *NETWORKING 2002 Workshops on Web Engineering and Peer-to-Peer Computing*, pages 127–133, Pisa, Italy, May 2002.

[Rey03] Mark Reynolds. The complexity of the temporal logic with until over general linear time. *Journal of Computer and System Sciences*, 66(2) :393–426, March 2003.

[RSA78] Ronald Linn Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2) :120–126, February 1978.

[Rya06a] Mark Dermot Ryan. Trusted computing : Concepts. University of Birmingham, School of Computer Science, 2006.

[Rya06b] Mark Dermot Ryan. Trusted computing : Tcg proposals. University of Birmingham, School of Computer Science, 2006.

[Ré03] République française. Article 9. In *Code civil*, 1803.

[Ré78] République française. Loi numéro 78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés. In *Journal Officiel de la République Française*, January 1978.

[Ré04] République française. Loi numéro 2004-801 du 6 août 2004 relative à la protection des personnes physiques à l'égard des traitements de données à caractère personnel. In *Journal Officiel de la République Française*, August 2004.

[SA00] Silvia N. Schiaffino and Analía Amandi. User profiling with case-based reasoning and bayesian networks. In Maria Carolina Monard and Jaime Simão Sichman, editors, *Open Discussion Track Proceedings of the International Joint Conference, 7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI (IBERAMIA-SBIA 2000)*, pages 12–21. ICMC/USP, 2000.

[Sah75] Henrik Sahlqvist. Correspondence and completeness in the first- and second-order semantics for modal logic. In *Proceedings of the Third Scandinavian Logic Symposium*. North-Holland, Amsterdam, 1975.

[Sar46] Jean-Paul Sartre. *L'existentialisme est un humanisme*. Nagel, 1946.

[SB05] B. Subirana and M. Bain. *Legal Programming : Designing Legally Compliant RFID And Software Agent Architectures For Retail Processes and Beyond*. Springer, USA, november 2005.

[Sea69] John Rogers Searle. *Speech acts*. Cambridge University Press, 1969.

[Seg03] Krister Segerberg. Some Meinong/Chisholm thesis. *Logic, Law, Morality. A festrichft in honor of Lennart Åqvist*, 51 :67–77, 2003. Uppsala Philosophical Studies.

[Sei07] Antoine Seilles. Modèles et outils de raisonnement argumentatif dans les communautés et organisations virtuelles. Master's thesis, Université Montpellier 2, 2007. encadrants : Abdelkader Gouaich et Jean Sallantin.

[Sho93] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1) :51–92, 1993.

[SL01] Sybil Shearin and Henry Lieberman. Intelligent profiling by example. In *Proceedings of the 2001 International Conference on Intelligent User Interfaces (IUI'01)*, pages 145–151, January 2001.

[SN98] Carles Sierra and Pablo Noriega. Institucions electròniques. In *Primer Congrès Català d'Intelligència Artificial*, 1998.

[SRC07] Ben Smyth, Mark Dermot Ryan, and Liqun Chen. Direct Anonymous Attestation (DAA) : Ensuring privacy with corrupt administrators. In F. Stajano, editor, *Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS'07)*, volume 4572 of *LNCS*, pages 218–231. Springer-Verlag, 2007.

[Str02] Tiberiu Stratulat. *Systèmes d'agents normatifs : concepts et outils logiques*. PhD thesis, Université de Caen, Caen, France, december 2002.

[Sxi] The sxipper plugin for firefox. http://www.sxipper.com/.

[TB06] Lynda Tamine and Wahiba Bahsoun. Définition d'un profil multidimensionnel de l'utilisateur : vers une technique basée sur l'interaction entre dimensions. In *COnférence en Rechercherche d'Information et Applications (CORIA)*, pages 225–236, Lyon, march 2006. Association Francophone de Recherche d'Information et Applications (ARIA).

[The95] The European Parliament and the Council. Directive 1995/46/EC of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. In European Union, editor, *Official Journal of the European Communities*, October 1995.

[The02] The European Parliament and the Council. Directive 2002/58/EC of the european parliament and of the council of 12 july 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector. In European Union, editor, *Official Journal of the European Communities*, July 2002.

[Tho75] Judith J. Thomson. The right of privacy. *Philosophy and Public Affairs*, 4 :295–314, 1975.

[Tru] Trusted Computing Group. https://www.trustedcomputinggroup.org/home.

[Tru03a] Trusted Computing Group. TCG main specification, version 1.1b. available via http ://www.trustedcomputinggroup.org/, 2003.

[Tru03b] Trusted Computing Group. TCG TPM specification, version 1.2. Available via http ://www.trustedcomputinggroup.org/, 2003.

[Tuo95] Raimo Tuomela. *The Importance of Us : A Philosophical Study of Basic Social Norms*. Stanford University Press, 1995.

[vdT94] Leendert W. N. van der Torre. Violated obligations in a defeasible deontic logic. In A. Cohn, editor, *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 371–375. John Wiley and Sons, 1994.

[von51] Georg Henrik von Wright. Deontic logic. *Mind*, 60 :1–15, 1951.

[von86] Franz von Kutschera. Bewirken. *Erkenntnis*, 24 :253–281, 1986.

[WB90] Samuel D. Warren and Louis D. Brandeis. The right of privacy. *Harvard Law Review*, 4 :193–195, 1890.

[WD01] David Wagner and Drew Dean. Intrusion detection via static analysis. In *IEEE Symposium on Security and Privacy*, pages 156–168, Oakland, California, USA, May 2001. IEEE.

[Wes67] Alan F. Westin. *Privacy and freedom*. New York, USA, 1967.

[Wik08a] Logique déontique. In Wikimedia Foundation, Inc., editor, *Wikipédia, l'encyclopédie libre*, 2003-2008. http://fr.wikipedia.org/wiki/Logique_d%C3%A9ontique.

[Wik08b] Logique modale. In Wikimedia Foundation, Inc., editor, *Wikipédia, l'encyclopédie libre*, 2003-2008. http://fr.wikipedia.org/wiki/Logique_modale.

[Wik08c] Formule de Sahlqvist. In Wikimedia Foundation, Inc., editor, *Wikipédia, l'encyclopédie libre*, 2007-2008. http://fr.wikipedia.org/wiki/Formule_de_Sahlqvist.

[Wik08d] Sémantique de Kripke. In Wikimedia Foundation, Inc., editor, *Wikipédia, l'encyclopédie libre*, 2007-2008. http://fr.wikipedia.org/wiki/S%C3%A9mantique_de_Kripke.

[Woo01a] Michael Wooldridge. *Introduction to Multiagent Systems*, chapter Logics for Multiagent Systems, pages 267–302. John Wiley & Sons, 2001.

[Woo01b] Michael Wooldridge. *Introduction to Multiagent Systems*, chapter Reaching agreements, pages 129–162. John Wiley & Sons, 2001.

[Wor99] World Wide Web Consortium. Using P3P for e-commerce, 1999. http://www.w3.org/TR/P3P-for-ecommerce.

[Wor02] World Wide Web Consortium. A P3P Preference Exchange Language 1.0 (APPEL 1.0), 2002. http://www.w3.org/TR/P3P-preferences/.

[Wor06] World Wide Web Consortium. Platform for Privacy Preferences specification 1.1., 2006. http://www.w3.org/P3P/.

[Yun03] Moti Yung. Trusted computing platforms : The good, the bad and the ugly. In Rebecca N. Wright, editor, *Proceedings of the 7th International Conference of Financial Cryptography*, number 2742 in LNCS, pages 250–254, Guadeloupe, French West Indies, January 2003. Springer Verlag.

Résumé

Les usages dans le domaine des systèmes multi-agents ont évolué de manière à intégrer davantage les utilisateurs humains dans les applications. La manipulation d'informations privées par des agents autonomes appelle alors à une protection adaptée des données personnelles. Les présents travaux examinent d'abord le contexte légal de la protection de la vie privée, ainsi que les divers moyens informatiques destinés à la protection des données personnelles. Il en ressort un besoin de solutions fondées sur les méthodes d'IA, autorisant à la fois un raisonnement sur les réglementations et l'adaptation du comportement d'un agent à ces réglementations. Dans cette perspective, nous proposons le modèle d'agent *PAw* (*Privacy-Aware*) et la logique DLP (*Deontic Logic for Privacy*), conçue pour traiter des réglementations provenant d'autorités multiples. Le composant de raisonnement normatif de l'agent analyse son contexte hétérogène et fournit une politique cohérente pour le traitement des données personnelles. L'agent PAw contrôle alors automatiquement sa propre utilisation des données en regard de cette politique. Afin d'appliquer cette politique de manière distante, nous étudions les différentes architectures d'applications distribuées orientées vers la protection de la vie privée, notamment celles fondées sur les principes du *Trusted Computing*. Nous en proposons une complémentaire, illustrant la possibilité d'utiliser différemment cette technologie. L'implémentation de l'agent PAw permet la démonstration de ses principes sur trois scénarios, montrant ainsi l'adaptabilité de l'agent à son contexte normatif et l'influence des réglementations sur le comportement de l'application.

Abstract

Usage in the domain of multi-agent systems has evolved so as to integrate human users more closely in the applications. Manipulation of private information by autonomous agents has then called for an adapted protection of personal data. This work first examines the legal context of privacy protection and the various computing methods aiming at personal data protection. Surveys show a significant need for AI-based solutions, allowing both reasoning on the regulations themselves and automatically adapting an agent's behaviour to these regulations. The Privacy-Aware (PAw) agent model and the Deontic Logic for Privacy, designed to deal with regulations coming from multiple authorities, are proposed here in this perspective. The agent's normative reasoning component analyses its heterogeneous context and provides a consistent policy for dealing with personal information. PAw agent then automatically controls its own usage of the data with regard to the resulting policy. In order to enforce policies in a remote manner, we study the different distributed application architectures oriented towards privacy protection, several of them based on the principles of Trusted Computing. We propose a complementary one, illustrating a different possible usage of this technology. Implementation of the PAw agent allows demonstration of its principles over three scenarios, thus showing the adaptability of the agent to its normative context and the influence of the regulations over the behaviour of the application.