Vérification formelle et éthique dans les SMA

Grégory Bonnet[a] Gregory.Bonnet@unicaen.fr
Bruno Mermet[a] Bruno.Mermet@unicaen.fr
Gaële Simon[a] Gaele.Simon@unicaen.fr

[a]Laboratoire GREYC - UMR 6072,
Université du Havre, France

Résumé

L'utilisation croissante d'agents autonomes artificiels dans des secteurs comme le milieu hospitalier ou les transports amène à réfléchir au respect de règles morales partagées par tous. Le problème est d'autant plus crucial que les règles morales informellement validées par tous sont souvent incompatibles les unes avec les autres, et que ce sont souvent des règles éthiques qui amènent l'humain, suivant les circonstances à privilégier telle ou telle règle morale. Utiliser la preuve pour vérifier qu'un agent respecte bien des règles morales et éthiques pourraient aider à accroître la confiance que nous pouvons avoir en de telles unités logicielles. Dans cet article, nous montrons, en nous appuyant sur une étude de cas, comment, à partir de règles morales a priori contradictoires mais ordonnées par des règles éthiques, nous parvenons à définir un ensemble de propriétés formelles qui, lorsqu'elles sont établies par un agent, permettent d'assurer que l'agent en question respecte bien la règle éthique souhaitée.

Mots-clés : *spécification formelle, systèmes multi-agents, éthique computationelle*

Abstract

The increasing use of autonomous artificial agents in hospitals or in transport control systems leads to consider whether moral rules shared by many of us are followed by these agents. This a particularly hard problem because most of these moral rules are often not compatible. In such cases, humans usually follow ethical rules to promote one moral rule or another. Using formal verification to ensure that an agent follows a given ethical rule could help in increasing the confidence in artificial agents. In this article, we show how a set of formal properties can be obtained from an ethical rule ordering conflicting moral rules. If the behaviour of an agent verifies these properties (which can be proven using our existing proof framework), it means that this agent follows this ethical rule.

Keywords: *Formal Specification, Multi-Agent Systems, Ethic, Computational Ethics*

1 Introduction

L'introduction d'agents autonomes artificiels dans des domaines comme le milieu hospitalier, le trading haute fréquence ou les transports pourrait soulever de nombreux problèmes si ces agents ne sont pas en mesure de comprendre et suivre certaines règles morales. Par exemple, des agents capables de comprendre et d'utiliser le code de déontologie médicale pourraient s'appuyer sur des motivations éthiques afin de choisir quelles informations diffuser, à qui et sous quelles conditions, conformément au principe du secret médical. C'est pour cela que la communauté [17, 32] semble récemment faire preuve d'intérêt pour l'éthique des agents autonomes, comme en témoignent de nombreux articles [6, 26, 28, 31] et conférences [1].

Cet article se place dans le cadre du projet ETHICAA [2] dont l'objectif est de traiter de la gestion des conflits moraux et éthiques entre agents autonomes. Si les travaux de la littérature s'intéressent essentiellement aux questions de raisonnement et de décision éthique [7, 39, 42, 4, 11, 20, 21], il n'y a que peu de travaux s'intéressant à la vérification formelle de tels comportement. Or la spécificité des codes moraux et éthiques est qu'ils peuvent ne pas être respectés en fonction des circonstances sans pour autant que cela pose problème. Par exemple dans un contexte humain, si voler est immoral, celui qui vole parce qu'il a faim n'est que rarement taxé d'immoralité.

Aussi, cet article présente un travail qui a pour but de proposer un environnement de spécification et de vérification formelle du comportement éthique d'un agent autonome. En reprenant les travaux d'Abramson et Pike, une règle morale est représentée par une propriététe formelle qu'un agent doit respecter [1] et un agent a un comportement éthique s'il respecte toutes

1. Symposium on Roboethics, International Conference on Computer Ethics and Philosophical Enquiry, Workshop on AI and Ethics, International Conference on AI and Ethics.

2. http://ethicaa.org/

les règles morales attendues selon les circonstances.

Considérant qu'une règle morale peut être représentée par une formule $\mathcal{F}$ en logique du premier ordre suffisamment expressive pour bien des cas pratiques, nous cherchons alors à établir qu'un agent a un comportement éthique si son comportement vérifie $\mathcal{F}$ ou qu'il n'a pas un comportement éthique si son comportement ne vérifie pas $\mathcal{F}$. Toutefois, un tel système logique est simplement semi-décidable : il n'est pas possible de prouver dans le cas général qu'un système ne vérifie pas une formule $\mathcal{F}$. En effet, si un prouveur automatique ne réussit pas à prouver qu'un agent vérifie bien une formule $\mathcal{F}$, il n'est pas possible d'identifier automatiquement si cela est dû au fait que l'agent ne vérifie pas $\mathcal{F}$ ou si c'est parce que le prouveur n'a pas réussi à faire la preuve du contraire.

Aussi, nous proposons d'utiliser un cadre formel qui réduise au maximum le nombre de formules correctes qui ne sont pas prouvées automatiquement. Nous présentons alors en section 2 de tels systèmes formels en nous attardant plus spécifiquement sur ceux consacrés aux SMA avant de définir ce que nous appelons règles morales et éthiques. Dans la section 3, nous détaillons plus précisément le système formel, puis nous présentons son utilisation dans un contexte éthique en section 4.

2 État de l'art

Depuis les débuts de l'informatique, la nécessité de garantir la correction d'un logiciel est apparue comme étant un problème majeur pour les développeurs. Cette nécessité est devenue vitale pour les systèmes critiques, c'est-à-dire les applications dédiées à des domaines où la sûreté de fonctionnement est vitale (comme les transports par exemple). Cependant, vérifier formellement un logiciel est un processus long et difficile qui va à l'encontre des critères de rentabilité et d'efficacité de certaines entreprises. Il existe principalement deux grands types de procédés de validation : le test et la preuve. Ici, nous ne nous intéresserons qu'à cette dernière. Les preuves peuvent être effectuées soit par des *model-checkers*, soit par des prouveurs de théorèmes qui peuvent ponctuellement faire appel à des *model-checkers*. Les *model-checkers* reposent fondamentalement sur un principe de test exhaustif, tandis que les prouveurs de théorème utilisent le calcul des séquents pour essayer, de manière heuristique, de générer des démonstrations.

Le processus de preuve peut être long et difficile, mais il permet de prouver des spécifications très préliminaires et de les raffiner progressivement jusqu'à l'obtention d'un code exécutable, avec des preuves progressives. Les erreurs sont ainsi détectées plus tôt, réduisant leur coût. Les formules à prouver sont également simplifiées, facilitant leur démonstration automatique. Ces preuves reposent alors sur une spécification formelle écrite grâce à un langage, un modèle ou une méthode formelle (ces termes sont souvent employés indifféremment pour exprimer la même chose).

2.1 Modèles et méthodes dédiés aux SMA

Le but premier des modèles dédiés aux SMA était d'aider les développeurs à concevoir des SMA. Bien qu'il en existe de nombreux modèles [38], le plus célèbre est certainement le modèle BDI [35] devenu aujourd'hui un standard aux nombreuses extensions, comme l'architecture BOID qui ajoute la notion d'*obligation* aux notions de croyance, de désir et d'intention du modèle BDI [10].

Deux des premières méthodes formelles dédiées aux SMA furent MetateM [19] et Desire [9]. Cependant, aucun des deux ne permet de spécifier les propriétés que le système doit garantir. Au contraire, les méthodes reposant sur la notion de rôle introduisent une abstraction qui aide à effectuer la phase d'analyse des besoins dans le travail de développement. Dans Gaia, un SMA est spécifié deux fois : par son comportement (via des propriétés de vivacité) et par des propriétés d'invariance. Aussi, cette méthode introduit les bases nécessaires pour effectuer des preuves de comportement. Cependant, effectuer de telles preuves n'est malheureusement pas possible avec Gaia car les propriétés ne sont pas associées aux agents mais aux rôles et, que lorsqu'un agent implante plusieurs rôles, aucune sémantique formelle ne vient déterminer comment ces rôles se combinent.

Une autre famille de méthodes est celle des méthodes orientées buts. La plupart se situent au niveau des agents et non au niveau du SMA, ce qui implique que la phase d'agentification doit avoir été effectuée préalablement. On peut cependant trouver deux exceptions : Moise [23] et PASSI [13]. Par exemple dans PASSI, les types d'agents sont produits en regroupant des *cas*

d'utilisation identifiés lors de la phase d'analyse. Il n'y a cependant pas de guide pour grouper ces cas d'utilisation.

Enfin, plus récemment, Dastani *et al.* ont proposé le langage 2APL [15]. Malheureusement, ce langage formel n'inclut pas de système de preuve. De plus, 2APL n'est pas compositonnel, ce qui rend le système trop monolithique s'il devait être utilisé dans un contexte de preuve.

2.2 Travaux centrés sur la preuve

Comme cela a été évoqué précédemment, il y a principalement deux façons de vérifier la correction d'une spécification : le *model-checking* et la preuve de théorèmes.

La plupart des travaux menés sur les agents utilisent le *model-checking* [8, 3, 34, 24]. Cependant, tous ces travaux partagent la même limite : l'explosion combinatoire des trajectoires possible du système rend la preuve de SMA complexes difficile, voire impossible. Ces systèmes se réduisent d'ailleurs la plupart du temps à de la preuve sur des formules propositionnelles et non sur des prédicats.

Les travaux sur l'utilisation de la preuve de théorèmes pour la vérification de SMA sont beaucoup plus rares. La principale raison est certainement due au fait que, la logique du premier ordre étant semi-décidable, les tentatives de preuves sont faites en utilisant des heuristiques et la preuve d'une propriété juste peut échouer. Cependant, de nombreux prouveurs de théorèmes peuvent maintenant réussir à prouver des propriétés très complexes automatiquement. C'est notamment le cas de PVS [33] et de KRT, le prouveur de l'atelier B [2].

Une autre approche intéressante utilise le raisonnement abductif [37]. Cependant, son modèle, basé sur les événements, la rend peu facile à utiliser pour les SMA.

D'autres modèles existent, reposant sur la programmation logique. C'est notamment le cas de CaseLP [25] et DCaseLP [5], plutôt adaptés à la preuve de théorèmes. Cependant, les seules preuves qui semblent applicables sur ces modèles semblent être des preuves de protocoles d'interaction.

Congolog [16] et CASL [40] sont aussi deux langages intéressants, reposant sur le calcul de situation. De plus, ils permettent tous les deux d'effectuer des preuves. Cependant, ces preuves ne concernent que l'enchaînement des actions, pas leur sémantique.

2.3 Règles morales et éthiques

De la philosophie jusqu'aux des travaux récents en neurologie et sciences cognitive [14, 22], les concepts de morale et d'éthique ont été discutés. Bien que ces termes désignent initialement une même idée, certains auteurs les nuancent [12, 36, 41]. En effet, la morale établit des règles permettant d'évaluer comme bien ou mal des situations ou des actions et l'éthique permet de concilier les règles morales lorsqu'elles entrent en conflit ou posent des problèmes dans leur applicabilité. C'est sur cette distinction que nous fondons nos définitions.

Règles morales. Ainsi, nous définissons une règle morale comme une règle qui doit préciser, dans un contexte donné, quels sont les états du système qui sont bien ou mal. Nous pouvons alors représenter une règle morale sous la forme $contexte \rightarrow P_{var}$, où p_var est un prédicat défini sur l'ensemble des variables connues. Une règle morale peut donc être vue comme un règle d'invariance conditionnelle particulière, dans le sens où sa vérification n'est pas nécessaire pour garantir un fonctionnement correct du système, mais qu'il est nécessaire de la vérifier si l'agent doit être utilisé dans un système où la règle morale en question s'applique. Par exemple, dans le cadre d'une voiture autonome, on peut considérer qu'une propriété de sûreté serait $voie = Autoroute \rightarrow vitesse \leq 130$. La voiture **doit** vérifier cette propriété. Cependant, afin de ne pas mettre de vie en danger, une règle morale de prudence r_p précise qu'en cas de verglas, la voiture ne doit pas dépasser 30 km/h, que nous notons formellement ainsi : $temps = verglas \rightarrow vitesse \leq 30$. Cette propriété n'est pas indispensable pour que la voiture ait un comportement valide, elle n'a pas à être obligatoirement vérifié mais elle prend du sens dans un système où les utilisateurs considèrent la préservation des vies humaines.

Règles éthiques. Lorsqu'un individu ou un agent suit plusieurs règles morales, il arrive que deux de ces règles, ou plus, soient en conflit. Une règle éthique permet notamment, dans un tel cas, de préciser comment agir. Si certains règles comme la doctrine du double effet [27] peuvent être complexes, nous considérons ici qu'une règle éthique est une règle qui précise, dans un tel cas, comment l'agent doit ordonner les

rè egles morales. Nous considérons également que, suivant les circonstances, une règle éthique peut donner des résultats différents. Si nous reprenons l'exemple du véhicule autonome, une règle morale de respect des autres conducteurs r_r consiste également à préciser que sur autoroute, il ne faut pas rouler à moins de 80km/h, ce qui s'exprime formellement par $voie = Autoroute \rightarrow vitesse \geq 80$. Nous avons alors un conflit entre la règle de prudence r_p évoquée précédemment et cette nouvelle règle r_r : s'il y a du verglas et que la voiture est sur l'autoroute, sa vitesse doit être inférieure à 30km/h d'après r_p, mais supérieure à 80 d'après r_r. Une règle éthique peut alors préciser que, dans tous les cas, la prudence (exprimée par r_p) est préférable au respect (exprimé par r_r). Mais une règle éthique pourrait aussi préciser que c'est le cas si on n'est pas en route pour les urgences et que dans le cas contraire, alors l'ordre de priorité des règles morales r_p et r_r est inversé.

2.4 Peu de travaux pour vérifier l'éthique

Traiter les problèmes éthiques de manière formelle est notamment étudié dans [1]. Dans cet article, les auteurs expliquent pourquoi utiliser des méthodes formelles pourrait être intéressant pour garantir que des agents respectent des règles éthiques. Cependant, il ne s'agit que d'une prise de position : aucune méthode concrète n'est proposée.

Dans [18], les auteurs proposent de formaliser et de vérifier la procédure de décision éthique suivante, décrite dans [43] : quand un choix doit être effectué entre plusieurs actions, une valeur est affectée à chaque action potentielle selon le niveau de sûreté garanti par l'action. Si une action est plus sûre qu'une autre, c'est elle qui est exécutée. Il y a cependant un inconvénient majeur au système : l'aspect éthique n'est pris en compte que lorsqu'un choix entre des actions doit être effectué en utilisant la procédure de sélection d'action décrite précédemment. Ainsi, ce travail n'est pas assez général pour fournir une réponse satifaisante.

3 GDT4MAS

Pour prendre en considération les problèmes éthiques, nous avons choisi d'utiliser la méthode GDT4MAS [29, 30]. En effet, cette méthode, qui intègre également un modèle, présente plusieurs caractéristiques intéressantes pour la prise en compte de problèmes éthiques :

- Cette méthode propose un langage formel pour exprimer non seulement les propriétés qu'un agent ou qu'un SMA doit respecter, mais aussi le comportement des agents ;
- Les propriétés sont spécifiées en logique du premier ordre, une notation formelle bien connue et expressive ;
- Le processus de preuve peut être effectué automatiquement.

Nous présentons rapidement dans la suite la méthode GDT4MAS. Plus de détails peuvent être trouvés dans les articles cités ci-dessus.

3.1 Concepts principaux

La méthode GDT4MAS nécessite de spécifier trois notions : l'environnement, les types d'agents et les agents eux-mêmes, qui sont des instances des types d'agents. Dans la suite de cette section, nous décrivons brièvement chacune des ces parties.

Environnement L'environnement est spécifié par un ensemble de variables typées et une propriété d'invariance $i_\mathcal{E}$.

Types d'agents Les types d'agents sont chacun spécifiés par un ensemble de variables typées, un invariant et un comportement. Le comportement d'un agent est principalement défini par un *arbre de décomposition des buts* (GDT [3]). Un GDT est un arbre de buts, dont la racine correspond au but principal de l'agent. Un plan est associé à chaque but : lorsque ce plan est exécuté avec succès il établit le but auquel il est associé. Un plan peut être constitué soit d'une simple action, soit d'un ensemble de buts reliés par un *opérateur de décomposition*. Un but B est principalement décrit par un nom n_B, une condition de satisfaction sc_B et une propriété garantie en cas d'échec gpf_B. La condition de satisfaction (SC [4]) d'un but est spécifiée formellement par une formule qui doit être satisfaite lorsque l'exécution du but réussit. Dans le cas contraire, la propriété garantie en cas d'échec (GPF [5]) d'un but spécifie ce qui est malgré tout garanti lorsque l'exécution d'un plan associé à un but échoue (ceci n'a pas de sens dans le cas d'un but dont le plan réussit toujours, ce qui est appelé un but NS [6]). Les SC et les GPF sont des *formules de transition d'état* (STF [7]) car elles

3. Goal Decomposition Tree.
4. Satisfaction Condition
5. Guaranted Property in case of Failure.
6. Necessarily Satisfiable.
7. State Transition Formula.

expriment la relation entre deux états, appelés état initial et état final. Par la suite, nous utiliserons le terme *STF non déterministe* quand, pour un état initial donné, plusieurs états finaux peuvent satisfaire la STF. Par exemple, la formule $x' > x$ est une STF non déterministe car, pour un état initial donné (comme $x = 0$), plusieurs états finaux peuvent la satisfaire (comme $x' = 2$ ou $x' = 10$).

Agents Les agents sont spécifiés comme des instances paramètrées de types d'agents, avec des valeurs effectives pour les paramètres des types d'agents.

3.2 Exemple de GDT

La figure 1 montre un exemple de GDT. Le but de ce comportement est d'allumer la lampe d'une pièce n (n est un paramètre du GDT). Pour procéder ainsi, l'agent essaie de rentrer dans la pièce. En effet, une cellule photo-électrique est censée détecter quand quelqu'un entre dans la pièce et allume alors la lampe, et cela semble donc être un plan valable. Pourtant, la cellule photo-électrique ne marche pas systématiquement (c'est pourquoi le but *Entering into the room* est NNS, c'est-à-dire non NS), et l'agent peut donc devoir utiliser l'interrupteur. Plus de détails peuvent être trouvés dans [30].

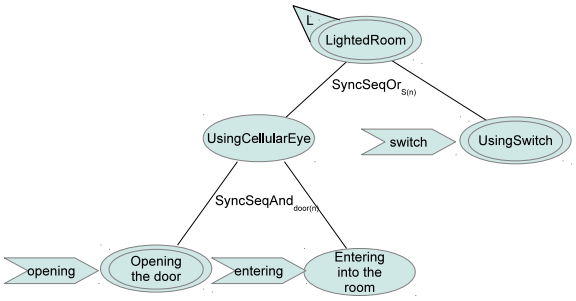

FIGURE 1 – Example of a GDT

3.3 Principes de preuve

Le mécanisme de preuve proposé par GDT4MAS a pour but de prouver les propriétés suivantes :

- Les agents préservent leurs propriétés d'invariance [30] qui sont des propriétés précisant que l'agent reste dans des états valides ;
- Le comportement des agents est consistant (autrement dit, les plans associés aux buts sont corrects) ;
- Les agents établissent leurs propriétés de vivacité, qui formalisent certaines caractéristiques dynamiques attendues de l'agent.

De plus, le mécanisme de preuve repose sur quelques principes essentiels : les *obligations de preuve* (propriétés à prouver pour garantir la correction du système) peuvent être générées automatiquement à partir d'une spécification GDT4AMAS. Elles sont exprimées en logique du premier ordre et peuvent être vérifiées par tout prouveur adéquat. Enfin, le système de preuve est compositionnel : la preuve de la correction d'un agent est décomposée en plusieurs petites obligations de preuves indépendantes.

4 Prouver une éthique

4.1 Caractérisation du problème

Considérons un agent ag dont le comportement a été formellement spécifié et vérifié, par rapport aux propriétés qu'il doit garantir. Supposons maintenant que cet agent doit être utilisé dans un univers donné avec une règle éthique er reposant sur un ensemble de règles morales. La question qui nous intéresse est la suivante : est-ce-que ag vérifie la règle éthique er ?

Comme GDT4MAS permet notamment de prouver les propriétés d'invariance, nous proposons d'exprimer les règles morales et les règles éthiques sous forme de telles propriétés. Il s'avère que la plupart des règles morales peuvent être facilement traduites en propriétés d'invariance. Nous proposons de structurer chaque règle morale ainsi :

$$\{(when_i, \{(var_i, set_i)\})\}$$

Cela signifie que chaque règle morale contraint, dans différents contextes ($when_i$), les valeurs (set_i) qui peuvent être assignées à différentes variables (var_i). Ainsi, si nous reprenons l'exemple de la règle de prudence r_p donnée dans la section 2.3, nous pourrions exprimer cette règle ainsi :

$$\{(temps = Verglas, \{(vitesse, \{0 \dots 30\})\})\}$$

Cependant, exprimer les règles éthiques sous la forme de propriétés d'invariance n'est pas si évident. En effet, elles ne caractérisent pas des états du système mais fournissent des ordres de priorité sur les règles morales suivant différents contextes.

Soit MR l'ensemble de règles morales et soit $\mathcal{P}$ l'ensemble des prédicats définissables sur les variables qui peuvent être vues par un agent donné a. Une règle éthique er est définie ainsi :

$$er \in \mathcal{P} +\!\!> (1 \ldots card(MR) >\!\!\!\gg MR)$$

Ici, $X +\!\!> Y$ représente l'ensemble des fonctions partielles de X vers Y et $X >\!\!\!\rightarrow\!\!\!\gg Y$ représente l'ensemble des bijections de X sur Y. Ainsi, informellement, cette définition signifie que, dans certains cas caractérisés par un prédicat donné $p \in \mathcal{P}$, les règles morales MR sont ordonnées. Par exemple, si $p \in \mathcal{P}$ est un prédicat, $er(p)(1)$ définit la règle morale la plus prioritaire lorsque p est vrai, $er(p)(2)$ définit la seconde la plus prioritaire, et ainsi de suite.

Considérons l'exemple suivant : soit un agent A_1qui doit décider la couleur d'un feu de circulation $tl1$ affecté à une route $r1$ à un croisement avec une route $r2$. Dans le système dans lequel cet agent agit, deux règles morales s'appliquent : la première, $mr1$, exprime que, pour éviter les accidents, lorsque la couleur du feu de la route $r2$ est $vert$ ou $orange$ alors la couleur de $tl1$ ne peut pas être $vert$. La règle $mr1$ peut alors être formalisée ainsi :

$$\{(tl2 \in \{green, orange\}, \{tl1, \{orange, red\}\})\}$$

La seconde règle morale, $mr2$, exprime que la route $r1$ est une route très prioritaire et qu'ainsi, la couleur de $tl1$ doit être toujours $vert$. Cela peut être formalisé ainsi :

$$\{(true, \{tl1, \{green\}\})\}$$

Bien évidemment, ces deux règles ne peuvent pas toujours être toutes les deux vérifiées. C'est notamment le cas lorsque le deuxième feu est $vert$: d'après $mr1$, la couleur de $tl1$ doit être $orange$ ou $rouge$ alors que selon $mr2$, la couleur de $tl1$ doit être $vert$.

Supposons maintenant que dans le système considéré, une règle éthique er donne des priorités aux règles morales. Par exemple, supposons que er précise que la route $r1$ est une règle prioritaire sauf si $tl2$ est $vert$ ou $orange$. En d'autres termes, cela signifie que $mr1$ a toujours une priorité plus élevée que $mr2$. Formellement :

$$\{(true, \{(1, mr1), (2, mr2)\})\}$$

4.2 Solution proposée

Dans le cadre de nos travaux, nous souhaitons vérifier qu'un agent vérifie bien une règle éthique définie sur un certain nombre de règles morales. Or un tel agent ne peut pas vérifier l'ensemble des règles morales qui le concernent puisque, comme nous l'avons expliqué précédemment, il arrive que ces règles soient contradictoires et qu'un état du système accessible par l'agent ne puisse vérifier toutes les règles morales. Aussi, afin de garantir qu'un agent respecte une règle éthique donnée, nous proposons un *système de transformation de prédicats* qui transforme les prédicats associés aux règles morales en d'autres prédicats, prouvables de façon à tenir compte de la règle éthique qui s'applique. Dans l'étude présentée ici, nous ne considérons qu'une situation avec deux règles morales mais le principe proposé pourrait être appliqué à un système avec plus de règles morales. Les règles morales et la règle éthique vont être transformées en un ensemble de propriétés d'invariance. Dans la suite, la transformation est montrée dans le cas où une seule variable est affectée par les règles morales. Dans le cas général, il faut générer les mêmes formules pour chaque variable apparaissant dans l'ensemble des règles morales (si une variable n'apparaît que dans certaines règles morales, elle est ajoutée dans les autres avec comme unique "contrainte" l'appartenance à son domaine de définition).

Considérons une telle variable V. Supposons également que la règle morale $mr1$ fournit les contraintes suivantes sur V :

$$mr1 = \left\{ \begin{array}{l} (when_{mr1_1}, (V, set_{mr1_1})) \\ (when_{mr1_2}, (V, set_{mr1_2})) \end{array} \right\}$$

Supposons qu'une seconde règle morale $mr2$ fournit les contraintes suivantes sur V :

$$mr2 = \left\{ \begin{array}{l} (when_{mr2_1}, (V, set_{mr2_1})) \\ (when_{mr2_2}, (V, set_{mr2_2})) \\ (when_{mr2_3}, (V, set_{mr2_3})) \end{array} \right\}$$

Enfin, considérons une règle éthique spécifiant que sous la condition $cond_1$, la règle $mr1$ est prioritaire par rapport à $mr2$, tandis que c'est le contraire sous la condition $cond_2$. Cette règle éthique er est donc définie ainsi :

$$er = \left\{ \begin{array}{l} (cond_1, \{(1, mr1), (2, mr2)\}) \\ (cond_2, \{(1, mr2), (2, mr1)\}) \end{array} \right\}$$

Nous générons alors un ensemble de propriétés d'invariance prouvables. Tout d'abord, sous

$cond_1$, $mr1$ est une règle prioritaire :

$$cond_1 \rightarrow (when_{mr1_1} \rightarrow V \in set_{mr1_1})$$
$$cond_1 \rightarrow (when_{mr1_2} \rightarrow V \in set_{mr1_2})$$

Puis, sous $cond_1$, quand $mr1$ ne s'applique pas, $mr2$ s'applique :

$$cond_1 \rightarrow \left(\begin{array}{l} (\neg when_{mr1_1} \wedge \neg when_{mr1_2}) \\ \rightarrow \\ (when_{mr2_1} \rightarrow V \in set_{mr2_1}) \end{array} \right)$$
$$cond_1 \rightarrow \left(\begin{array}{l} (\neg when_{mr1_1} \wedge \neg when_{mr1_2}) \\ \rightarrow \\ (when_{mr2_2} \rightarrow V \in set_{mr2_2}) \end{array} \right)$$
$$cond_1 \rightarrow \left(\begin{array}{l} (\neg when_{mr1_1} \wedge \neg when_{mr1_2}) \\ \rightarrow \\ (when_{mr2_3} \rightarrow V \in set_{mr2_3}) \end{array} \right)$$

Enfin, lorsque $cond_1$ est vraie, quand mr_1 et mr_2 s'appliquent, si c'est possible, une valeur satisfaisant les 2 règles morales doit être choisie :

$$\left(\begin{array}{l} (cond_1 \wedge when_{mr1_1} \wedge when_{mr2_1}) \\ \rightarrow \\ (set_{mr1_1} \cap set_{mr2_1} \neq \emptyset \rightarrow V \in set_{mr1_1} \cap set_{mr2_1}) \end{array} \right)$$
$$\left(\begin{array}{l} (cond_1 \wedge when_{mr1_1} \wedge when_{mr2_2}) \\ \rightarrow \\ (set_{mr1_1} \cap set_{mr2_2} \neq \emptyset \rightarrow V \in set_{mr1_1} \cap set_{mr2_2}) \end{array} \right)$$
$$\left(\begin{array}{l} (cond_1 \wedge when_{mr1_1} \wedge when_{mr2_3}) \\ \rightarrow \\ (set_{mr1_1} \cap set_{mr2_3} \neq \emptyset \rightarrow V \in set_{mr1_1} \cap set_{mr2_3}) \end{array} \right)$$
$$\left(\begin{array}{l} (cond_1 \wedge when_{mr1_2} \wedge when_{mr2_1}) \\ \rightarrow \\ (set_{mr1_2} \cap set_{mr2_1} \neq \emptyset \rightarrow V \in set_{mr1_2} \cap set_{mr2_1}) \end{array} \right)$$
$$\left(\begin{array}{l} (cond_1 \wedge when_{mr1_2} \wedge when_{mr2_2}) \\ \rightarrow \\ (set_{mr1_2} \cap set_{mr2_2} \neq \emptyset \rightarrow V \in set_{mr1_2} \cap set_{mr2_2}) \end{array} \right)$$
$$\left(\begin{array}{l} (cond_1 \wedge when_{mr1_2} \wedge when_{mr2_3}) \\ \rightarrow \\ (set_{mr1_2} \cap set_{mr2_3} \neq \emptyset \rightarrow V \in set_{mr1_2} \cap set_{mr2_3}) \end{array} \right)$$

Des propriétés d'invariance similaires doivent aussi être générées lorsque $cond_2$ est vraie, mais cette fois-ci avec $mr2$ comme règle prioritaire.

Appliquons ce mécanisme sur l'exemple présenté précédemment. Comme $cond_1$ vaut $vrai$, les formules peuvent être simplifiées. De plus, comme il n'y a qu'un cas (un seul $when$) pour $mr1$ et $mr2$, les formules précédentes peuvent être simplifiées comme suit. Sous $cond_1$, $mr1$ est une règle prioritaire :

$$when_{mr1_1} \rightarrow V \in set_{mr1_1}$$

Sous $cond_1$, quand $mr1$ ne s'applique pas, $mr2$ s'applique :

$$(\neg when_{mr1_1}) \rightarrow (when_{mr2_1} \rightarrow V \in set_{mr2_1})$$

Sous $cond_1$, quand mr_1 et mr_2 s'appliquent, si possible, une valeur satisfiant les deux règles morales doit être choisie :

$$(when_{mr1_1} \wedge when_{mr2_1}) \rightarrow \left(\begin{array}{l} set_{mr1_1} \cap set_{mr2_1} \neq \emptyset \\ \rightarrow \\ V \in set_{mr1_1} \cap set_{mr2_1} \end{array} \right)$$

De plus, nous avons :

$$\left\{ \begin{array}{l} V \equiv TL1 \\ when_{mr1_1} \equiv (TL2 \in \{green, orange\}) \\ set_{mr1_1} \equiv (\{orange, red\}) \\ when_{mr2_1} \equiv (true) \\ set_{mr2_1} \equiv (\{green\}) \end{array} \right.$$

Nous obtenons ainsi l'invariant suivant, qui doit être vérifié pour garantir que l'agent $a1$ respecte la règle éthique qui spécifie que la route $r1$ est une route prioritaire, sauf si le feu $tl2$ est $vert$ ou $orange$:

$$\begin{array}{l} TL2 \in \{green, orange\} \rightarrow TL1 \in \{orange, red\} \\ TL2 \notin \{green, orange\}) \rightarrow TL1 \in \{green\} \\ (TL2 \in \{green, orange\}) \rightarrow \left(\begin{array}{l} \{orange, red\} \cap \{green\} \neq \emptyset \\ \rightarrow \\ TL1 \in \{orange, red\} \cap \{green\} \end{array} \right) \end{array}$$

Comme $\{orange, red\} \cap \{green\} = \emptyset$, cette propriété d'invariance peut être simplifiée en :

$$TL2 \in \{green, orange\} \rightarrow TL1 \in \{orange, red\}$$
$$TL2 \notin \{green, orange\}) \rightarrow TL1 \in \{green\}$$

Ainsi, grâce au système de transformation de prédicat proposé, nous obtenons un nouvel invariant qui sera respecté par un agent qui respecte les règles morales données en suivant la règle éthique du système. Par la même occasion, la spécification formelle d'un tel agent pourra être prouvée.

4.3 Étude de cas

Dans cette section, nous montrons l'application des propositions présentées précédemment sur un cas dans lequel une question éthique plus traditionnelle se pose. On se place dans le cadre de 3 agents A, B et C qui doivent trouver une date de réunion. Un agent peut proposer une date et les deux autres agents doivent indiquer à tous si la date leur convient ou pas. Par exemple, l'agent A peut proposer une date de réunion à B et C. Si B ou C n'acceptent pas la date, ils doivent donner la raison de leur refus aux autres agents. Soit d une date proposée par A. L'agent C doit respecter les deux règles morales suivantes :

– $mr1 : C$ ne souhaite pas faire de mal ;
– $mr2 : C$ doit dire à A et B pourquoi la date d ne lui convient pas.

Cependant, si la vraie raison qui amène C à refuser la date d fait du mal à A (par exemple avoir un rendez-vous galant avec la femme de A), il y a un conflit entre les deux règles morales $mr1$ et $mr2$. Pour résoudre ce conflit, l'agent C dispose d'une règle éthique er stipulant que dans tous les cas, il est préférable de ne pas faire de mal plutôt que de dire la vérité.

Afin de formaliser le problème, nous introduisons quelques notations. Appelons $E_{RP} = \{r1, r2, r3, r4\}$ l'ensemble des réponses que C peut donner à A ou B et V_{RC} la variable contenant la vraie raison du refus de C. Pour concrétiser le problème, nous pouvons donner le sens suivant aux différentes réponses possibles :
– $r1$: J'ai un rendez-vous galant avec la femme de A ;
– $r2$: Je suis malade ;
– $r3$: A organise mal les réunions ;
– $r4$: J'ai eu un accident avec la voiture que B m'a prêtée.

Nous représentons également l'ensemble des réponses douloureuses pour chacun des agents par une fonction $F_{RD} \in agents \rightarrow \mathcal{P}(E_{RP})$. Pour l'exemple, nous prendrons $F_{RD} = \{(A, \{r1, r3\}), (B, \{r4\})\}$, ce qui signifie que $r1$ et $r3$ sont des réponses douloureuses pour l'agent A, tandis que $r4$ est une réponse douloureuse pour l'agent B. Nous appelons également V_{RFA} la variable contenant la réponse faite à l'agent A et V_{RFB} la variable contenant la réponse faite à l'agent B.

Nous avons donc identifié deux règles morales :
– $mr1 : C$ ne souhaite pas faire de mal à A et à B, donc ses réponses doivent être réponses possibles non douloureuses ;
– $mr2 : C$ ne veut pas mentir, donc la réponse doit être la vraie raison.

Ces règles peuvent être formalisées ainsi :

$$mr1 : \left\{true, \left\{ \begin{array}{l} (V_{RFA}, E_{RP} - F_{RD}(A)) \\ (v - RFB, E_{RP} - F_{RD}(B)) \end{array} \right\}\right\}$$

$$mr2 : \{true, \{(V_{RFA}, \{V_{RC}\}), (V_{RFB}, \{V_{RC}\})\}\}$$

Enfin, nous avons une règle éthique qui s'applique, er, qui précise que dans tous les cas, la rè gle $mr1$ est prioritaire sur la rè gle $mr2$. Autrement dit, nous avons :

$$er = \{(true, \{(1, mr1), (2, mr2)\})\}$$

En appliquant les résultats de la section précédente, nous rajoutons à l'invariant de C les formules suivantes (nous ne montrons ici que les formules générées pour V_{RFA} ; il faut normalement également rajouter des formules similaires pour V_{RFB}.

Sous $cond_1$, $mr1$ est une règle prioritaire :

$$true \rightarrow (true \rightarrow V_{RFA} \in E_{RP} - F_{RD}(A))$$

Sous $cond_1$, quand $mr1$ ne s'applique pas, $mr2$ s'applique :

$$true \rightarrow ((\neg true \wedge \neg true) \rightarrow (true \rightarrow V_{RFA} \in \{V_{RC}\}))$$

Sous $cond_1$, quand $mr1$ et $mr2$ s'appliquent, si possible, une valeur satisfiant les deux règles morales doit être choisie :

$$\begin{array}{l} (true \wedge true \wedge true \rightarrow \\ \quad ((E_{RP} - F_R D(A)) \cap \{V_{RC}\} \neq \emptyset \rightarrow \\ \quad\quad V_{RFA} \in (E_{RP} - F_{RD}(A)) \cap \{V_{RC}\})) \end{array}$$

Ceci peut alors se simplifier en :

$$\begin{array}{l} V_{RFA} \in E_{RP} - F_{RD}(A) \\ ((E_{RP} - F_{RD}(A)) \cap \{V_{RC}\} \neq \emptyset \rightarrow \\ \quad V_{RFA} \in (E_{RP} - F_{RD}(A)) \cap \{V_{RC}\})) \end{array}$$

Ces propriétés, ajoutées à l'invariant de l'agent, permettent, si elles sont prouvées pour l'agent C de garantir que le comportement de cet agent respecte la règle éthique évoquée. Ainsi, dans le cas présenté, et pour le cas où la vraie raison du refus de C serait $r1$, un agent dont le comportement respecte la règle éthique présentée précédemment devrait simplement vérifier :

$$V_{RFA} \in \{r1, r2, r3, r4\} - \{r1, r3\}$$

Soit :

$$V_{RFA} \in \{r2, r4\}$$

Par contre, si la réponse correcte est $r2$, le comportement de l'agent C devrait vérifier les deux propriétés suivantes :

$$\begin{array}{l} V_{RFA} \in \{r1, r2, r3, r4\} - \{r1, r3\} \\ V_{RFA} \in (\{r1, r2, r3, r4\} - \{r1, r3\}) \cap \{r2\} \end{array}$$

Ce qui revient à n'autoriser que l'unique solution : $V_{RFA} = r2$. En poursuivant sur chaque raison, nous arrivons au tableau suivant :

V_{RC}	$r1$	$r2$	$r3$	$r4$
V_{RFA}	$r2, r4$	$r2$	$r2, r4$	$r4$

Nous avons ainsi des propriétés qui seront vérifiées par un agent qui préfère mentir à blesser, mais qui, dès qu'il le peut, dit la vérité. En effet, lorsque la vraie raison ne blesse pas A ($r2$ ou $r4$), un agent dont le comportement a été prouvé devra donner à A cette vraie raison. Par contre, lorque la vraie raison blesse A ($r1$ ou $r3$), un agent correct mentira en donnant un autre justificatif ne blessant pas A (ici, $r2$ ou $r4$).

5 Conclusion et perspectives

Dans cet article, nous avons montré qu'il est possible de prouver formellement qu'un agent peut respecter des règles morales potentiellement contradictoires à partir du moment où une règle éthique permet d'établir, dans les cas où des contradictions apparaissent, quelles sont les priorités qui doivent exister entre les différentes règles morales. Pour ce faire, nous avons introduit des transformeurs de prédicats permettant de générer un ensemble de prédicats cohérents à partir de règles morales pourtant contradictoires. Après un exemple simple ayant servi à introduire les concepts, nous avons montré sur un cas concret que le système proposé pouvait être appliqué sur des cas réels.

D'autres études de cas seront cependant nécessaires pour continuer à valider la portée du système proposé. Notamment, nous nous sommes restreints à des règles morales qui pouvaient être exprimées sous la forme de contraintes disjointes d'affectation de valeurs à des variables. Il nous apparait important de vérifier la portée de cette restriction. Pour les cas où cette restriction rendrait invalide notre méthode, il faudra alors étudier comment l'étendre à des affectations de valeurs liées. Par exemple, on pourrait imaginer que la règle de prudence caractérisant la conduite en cas de verglas pourrait relier la vitesse maximum à l'angle que fait la direction de la voiture avec la ligne droite ainsi : $temps = Verglas \rightarrow vitesse + angle/2 \leq 40$. En effet, plus la voiture prend un virage serré, plus la vitesse doit être réduite pour éviter que la voiture ne dérape.

Enfin, d'un point de vue philosophique, notre système doit être étendu pour capturer de manière plus exacte la morale et l'éthique, en particulier en considérant la notion de valeur. En effet, ce sont des valeurs, comme la générosité, l'égalité, l'amour de la vérité, qui sous-tendent les règles de morales et, dans un contexte donné, le jugement éthique hiérarchise ces valeurs. Modéliser clairement la notion de valeur est donc la prochaine étape de notre travail.

Références

[1] D. Abramson and L. Pike. When formal Systems Kill : Computer Ethics and Formal Methods. *APA Newsletter on Philosophy and Computers*, 11(1), 2011.

[2] J.-R. Abrial. *The B-Book*. Cambridge Univ. Press, 1996.

[3] N. Alechina, B. Logan, and M. Whitsey. A complete and decidable logic for resource-bounded agents. In *3rd AAMAS*, 2004.

[4] R. Arkin. *Governing Lethal Behavior in Autonomous Robots*. Chapman and Hall, 2009.

[5] M. Baldoni, C. Baroglio, I. Gungui, A. Martelli, M. Martelli, V. Mascardi nd V. Patti, and C. Schifanella. Reasoning About Agents' Interaction Protocols Inside DCaseLP. In *LNCS*, volume 3476, pages 112–131, 2005.

[6] A.F. Beavers. Moral machines and the threat of ethical nihilism. In *Robot ethics : the ethical and social implication of robotics*, pages 333–386. MIT Press, 2011.

[7] F. Berreby, G. Bourgne, and J.-G. Ganascia. Modelling moral reasoning and ethical responsibility with logic programming. In *20th LPAR*, pages 532–548, 2015.

[8] R.H. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifiable multi-agent programs. In *1st ProMAS*, 2003.

[9] F.M.T. Brazier, P.A.T. van Eck, and J. Treur. *Simulating Social Phenomena*, volume 456, chapter Modelling a Society of Simple Agents : from Conceptual Specification to Experimentation, pages pp 103–109. LNEMS, 1997.

[10] J. Broersen, M. Dastani, Z. Huang, J. Hulstijn, and L. Van der Torre. The BOID architecture : conflicts between beliefs, obligations, intentions and desires. In *5th AA*, pages 9–16, 2001.

[11] H. Coelho and A.C. da Rocha Costa. On the intelligence of moral agency, 2009.

[12] A. Comte-Sponville. *La philosophie*. PUF, 2012.

[13] M. Cossentino and C. Potts. A CASE tool supported methodology for the design of multi-agent systems. In *SERP*, 2002.

[14] A. Damasio. *Descartes' error : Emotion, reason and the human brain*. Random House, 2008.

[15] M. Dastani. 2APL : a practical agent programming language. *JAAMAS*, 16 :214–248, 2008.

[16] G. de Giacomo, Y. Lesperance, and H. J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1-2) :109–169, 2000.

[17] Commission de réflexion sur l'Éthique de la Recherche en science et technologies du Numérique d'Allistene. éthique de la recherche en robotique. Technical report, CERNA, 2014.

[18] L.A. Dennis, M. Fisher, and A.F.T. Winfield. Towards Verifiably Ethical Robot Behaviour. In *Artifial Intelligence and Ethics AAAI Workshop*, 2015.

[19] M. Fisher. A survey of concurrent METATEM – the language and its applications. In *1st ICTL*, pages 480–505, 1994.

[20] J.-G. Ganascia. Ethical system formalization using non-monotonic logics. In *29th ACCSS*, pages 1013–1018, 2007.

[21] J.G. Ganascia. Modeling ethical rules of lying with answer set programming. *Ethics and Information Technologyn*, 9 :39–47, 2007.

[22] J. Greene and J. Haidt. How (and where) does moral judgment work ? *Trends in Cognitive Sciences*, 6(12) :517–523, 2002.

[23] J.F. Hubner, J.S. Sichman, and O. Boissier. Spécification structurelle, fonctionnelle et déontique d'organisations dans les SMA. In *JFIADSM*. Hermes, 2002.

[24] M. Kacprzak, A. Lomuscio, and W. Penczek. Verification of multiagent systems via unbounded model checking. In *3rd AAMAS*, 2004.

[25] M. Martelli, V. Mascardi, and F. Zini. CaseLP : a Complex Application Specification Environment base on Logic Programming. In *8th ICLP*.

[26] D. McDermott. Why ethics is a high hurdle for ai. In *North American Conference on Computers and Philosophy*, 2008.

[27] A. McIntyre. Doctrine of double effect. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter edition, 2014.

[28] B.M. McLaren. Computational models of ethical reasoning : challenges, initial steps, and future directions. *IEEE Intelligent Systems*, 21(4) :29–37, 2006.

[29] B. Mermet and G. Simon. Specifying recursive agents with GDTs. *JAAMAS*, 23(2) :273–301, 2011.

[30] B. Mermet and G. Simon. A new proof system to verify GDT agents. In *IDC*, volume 511 of *Studies in Computational Intelligence*, pages 181–187. Springer, 2013.

[31] J.H. Moor. The nature, importance, and difficulty of machine ethics. *IEEE Intelligent Systems*, 21(4) :29–37, 2006.

[32] Future of Life Institute. Research priorities for robust and beneficial artificial intelligence, 2015.

[33] S. Owre, N. Shankar, and J. Rushby. Pvs : A prototype verification system. In *11th CADE*, 1992.

[34] F. Raimondi and A. Lomuscio. Verification of multiagent systems via orderd binary decision diagrams : an algorithm and its implementation. In *3rd AAMAS*, 2004.

[35] A. Rao and M. Georgeff. BDI agents from theory to practice. In *Technical note 56*. AAII, 1995.

[36] P. Ricoeur. *Soi-même comme un autre*. Points Essais, 1990.

[37] A. Russo, R. Miller, B. Nuiseibeh, and J. Kramer. An abductive approach for analysing event-based requirements specifications. Technical report, Department of Computing, Imperial College, 2001.

[38] A. Sabas, S. Delisle, and M. Badri. A comparative analysis of multiagent system development methodologies : Towards a unified approach. In *Cybernetics and Systems*, pages 599–604. Austrian Society for Cybernetics Studies, 2002.

[39] A. Saptawijaya and L.M. Pereira. Towards modeling morality computationally with logic programming. In *16th ISPADL*, pages 104–119, 2014.

[40] S. Shapiro, Y. Lespérance, and H. J. Levesque. The Cognitive Agents Specification Language and Verification Environment for Multiagent Systems. In *2nd AAMAS*, pages 19–26, 2002.

[41] M. Timmons. *Moral theory : An introduction*. Rowman and Littlefiled, 2012.

[42] M. Tufis and J.-G. Ganascia. Normative rational agents : A BDI approach. In *1st Workshop on Rights and Duties of Autonomous Agents*, pages 37–43. CEUR Proceedings Vol. 885, 2012.

[43] A.F.T. Winfield, C. Blum, and W. Liu. Towards and Ethical Robot : Internal Models, Consequences and Ethical Action Selection. In *LNCS*, volume 8717, pages 85–96, 2014.