Comment faire des simulations sociales en logique : formalisation du modèle de ségrégation dans une logique dynamique des affectations

Benoit Gaudou, Andreas Herzig, Emiliano Lorini, Christophe Sibertin-Blanc

Comment faire des simulations sociales en logique : formalisation du modèle de ségrégation dans une logique dynamique des affectations

B. Gaudou⋆† benoit.gaudou@univ-tlse1.fr
A. Herzig† herzig@irit.fr
E. Lorini† lorini@irit.fr
C. Sibertin-Blanc⋆† sibertin@univ-tlse1.fr

†UMR 5505 CNRS, Institut de Recherche en Informatique de Toulouse (IRIT)
Toulouse – FRANCE

⋆ Université de Toulouse
Toulouse – FRANCE

Résumé :
Le but de cet article est de montrer comment *faire* des simulations sociales en logique. Afin d'atteindre cet objectif, nous présentons une logique dynamique avec affectations et itérations bornées et non-bornées. Nous montrons que notre logique permet de représenter et de raisonner sur un exemple paradigmatique de la simulation sociale : le modèle de ségrégation de Schelling. Nous établissons également un lien entre simulation et planification. En particulier, nous montrons que le problème de la vérification qu'une certaine propriété *P* (comme la ségrégation) va émerger après *n* pas de simulation peut se ramener à un problème de planification à horizon *n*, c'est-à-dire le problème de la vérification de l'existence d'un plan de longueur au plus *n* assurant qu'un certain but va être atteint, problème qui a été très étudié en IA.

Mots-clés : Système Multi-Agents, logique, simulation, complexité

Abstract:
The aim of this paper is to show how to do social simulation in logic. In order to meet this objective we present a dynamic logic with assignments, tests, sequential and nondeterministic composition, and bounded and non-bounded iteration. We show that our logic allows to represent and reason about a paradigmatic example of social simulation : Schelling's segregation game. We also build a bridge between social simulation and planning. In particular, we show that the problem of checking whether a given property *P* (such as segregation) will emerge after *n* simulation moves is nothing but the planning problem with horizon *n*, which is widely studied in AI : the problem of verifying whether there exists a plan of length at most *n* ensuring that a given goal will be achieved.

Keywords: multiagent systems, logic, simulation, complexity

1 Introduction

Dans un débat récent, Bruce Edmonds [7] a attaqué ce qu'il appelle des "articles de logique formelle vides et sans aucun résultat" qui sont souvent proposés dans le domaine des systèmes multi-agents (SMA). Il les oppose aux travaux décrivant des simulations sociales : selon Edmonds, ces derniers présentent des résultats expérimentaux utiles pour mieux comprendre les phénomènes sociaux, au contraire des articles de logique qui ont seulement pour but d'étudier des concepts pertinents pour des phénomènes sociaux et leurs propriétés mathématiques (axiomatisation, décidabilité...), sans améliorer notre compréhension des phénomènes sociaux eux-mêmes. En réponse à cette attaque, plusieurs chercheurs ont défendu l'intérêt de la logique pour les SMA en général et pour la simulation sociale à base d'agents (SSBA) en particulier [8, 4, 5]. Par exemple, Conte et Paolucci, dans [4], ont soutenu l'idée que la logique pourrait permettre de construire une théorie sociale formelle qui pourrait s'avérer très utile à ce domaine. Dans [8], Fasli soutient la thèse selon laquelle la logique peut être utile dans les SSBA car un modèle logique basé sur (a) une théorie philosophique ou sociologique, (b) des observations et des données sur un phénomène social particulier et (c) des intuitions (ou un mélange des trois) peut fournir les pré-requis et la spécification d'un simulateur à base d'agents et plus généralement d'un SMA. De plus, un système logique peut aider à vérifier la validité d'un modèle de simuler et à l'ajuster afin d'obtenir une meilleure compréhension du phénomène modélisé. Toutes ces recherches considèrent que la logique et les SSBA sont non seulement des outils compatibles mais aussi complémentaires.

L'idée que nous défendons dans cet article est plus radicale. Notre but est de montrer que les SSBA peuvent être directement faites en logique et que la spécification logique d'un phénomène social peut être vue comme un modèle de simulation de ce phénomène. Nous pensons que l'utilisation de démonstrateurs automatiques adéquats permettra d'obtenir des résultats qui vont au-delà des possibilités des simulateurs existants. Pour cela, nous présentons dans cet article une logique simple appelée $\mathsf{DL}^{\mathsf{PA}}$ (Logique Dynamique des Affectations). $\mathsf{DL}^{\mathsf{PA}}$ est une extension de la logique propositionnelle avec des opérateurs dynamiques. Ces opérateurs permettent

de raisonner sur les *affectations* $p{\leftarrow}\top$ et $p{\leftarrow}\bot$, qui affectent à la variable propositionnelle p la valeur 'vrai' ou 'faux', et sur les *tests* $\Phi?$ de la valeur d'une formule booléenne Φ. Plus généralement, $\mathsf{DL}^{\mathsf{PA}}$ permet de raisonner sur les faits qui vont être vrais après des événements complexes ϵ, qui sont construits à partir d'affectations et de tests au moyen d'opérateurs de composition séquentielle $(\epsilon_1;\epsilon_2)$, de composition non-déterministe $(\epsilon_1 \cup \epsilon_2)$, d'itération bornée $(\epsilon^{\leq n})$ et non-bornée $(\epsilon^{<\infty})$.

Afin d'illustrer le pouvoir de notre logique, nous montrons qu'un exemple paradigmatique de la SSBA, le modèle de ségrégation de Schelling [16], peut être représenté et simulé dans notre logique. Le problème de vérifier si, sous certaines conditions initiales, une propriété donnée P, telle que la ségrégation, *peut émerger* après n pas de simulation est réduit à un problème de vérification dans notre logique que les conditions initiales impliquent que la formule φ, codant P, sera vraie à la fin d'au moins une séquence d'évènements ϵ de taille au plus n. De manière similaire, le problème de vérifier si P *va nécessairement émerger* après n pas de simulation est réduit au problème de la vérification que les conditions initiales impliquent que φ sera vraie à la fin de toute séquence d'événements ϵ de taille n. Ce dernier problème est en fait un problème de planification à horizon n, qui a été beaucoup étudié en IA et qui est par exemple à la base de planificateurs tels que SatPlan [14] : il s'agit de vérifier s'il existe un plan de longueur au plus n assurant qu'un but φ sera atteint. Dans le cas général, ce problème est connu pour être PSPACE, c'est-à-dire décidable dans un espace polynomial [3]. Nous montrons que notre logique reste dans ce domaine.

Dans le passé, ces problèmes de décisions PSPACE étaient considérés comme hors de portée des démonstrateurs automatiques de théorèmes. Cependant, dans les 20 dernières années, de grands progrès ont été faits dans ce domaine : des démonstrateurs pour des problèmes PSPACE complets sont apparus et ont montré qu'ils pouvaient avoir un intérêt pratique dans des applications de web sémantique, même pour des problèmes réalistes comprenant des milliers de clauses [13].

Nous pouvons également aller au-delà des simples quantificateurs existentiels et universels mentionnés ci-dessus. Pour cela, nous pouvons introduire des opérateurs modaux avec dénombrement (provenant des logiques modales valuées [9, 20] et des logiques de descriptions [1]). Nous discuterons brièvement ce point et montrerons que la complexité reste PSPACE.

La suite de l'article est organisée comme suit. En section 2, nous décrivons le modèle de ségrégation de Schelling. Nous définissons notre logique en section 3 et montrons en section 4 comment elle permet de traiter ce modèle. Enfin nous esquissons un certain nombre d'extensions (section 5) et concluons (section 6).

2 Le modèle de ségrégation

Dans cette section nous donnons une description informelle du modèle de ségrégation de Schelling [16]. La formalisation de cet exemple est présentée en section 4.

2.1 Le modèle original

Thomas C. Schelling a étudié dans [16] le phénomène de ségrégation en lien avec des caractéristiques visibles des individus tels que le sexe, l'âge, la couleur de peau etc., en mettant en évidence les conditions de son apparition du fait des "choix discriminatoires individuels". L'exemple le plus connu est la formation de zones résidentielles en fonction de la couleur de peau, sous l'influence des préférences individuelles d'être entouré par au plus un certain nombre de voisins ayant une couleur de peau différente : au-dessus de ce seuil, les habitants ne sont pas heureux et vont déménager.

Un des principaux résultats du travail de Schelling a été de montrer que le phénomène de ségrégation apparaît même avec un seuil de tolérance élevé. Par exemple, même si chaque habitant accepte que plus de la moitié de ses visions aient une couleur de peau différente de la sienne, des groupe d'habitants de même couleur vont avoir tendance à se former.

2.2 Le modèle implémenté

Le modèle de ségrégation a été implémenté dans de nombreux langages et formalismes, en particulier dans presque toutes les plates-formes de simulation multi-agents (par exemple NetLogo [23, 22] et GAMA [18]). Chaque habitant (ou famille) est représenté par un agent parmi l'ensemble des agents $\mathbb{A} = \{1, \ldots, |\mathbb{A}|\}$. Les éléments de $\mathbb{A}$ seront typiquement notés i, j... Chaque agent peut se déplacer sur une grille

(sorte d'échiquier) de taille $N \times N$, avec N tel que $|\mathbb{A}| < N^2$. Chaque agent est caractérisé par sa couleur (par exemple rouge ou bleue) et sa position sur la grille, décrite par un couple d'entiers $(k, l) \in [1..N] \times [1..N]$, avec $1 \leq k, l \leq N$. Les deux paramètres principaux de la simulation sont :
– le nombre d'habitants $|\mathbb{A}|$,
– le seuil de tolérance : le nombre d'agents différents au-delà duquel un agent est malheureux (supposé identique pour chaque agent).
(Un autre jeu de paramètres pourrait être la densité d'habitants. On peut remarquer également que beaucoup de modèles utilisent plutôt l'inverse du seuil de tolérance, appelé seuil de similarité, ou bien le pourcentage d'agents différents, sans que cela n'influence la simulation.)

Les simulateurs ou les plates-formes de simulation possèdent un ordonnanceur (ou *scheduler*) qui génère, à chaque pas de simulation, un ordonnancement, le plus souvent aléatoire, de l'ensemble des agents et les activent dans cet ordre. A chaque activation, l'agent vérifie son niveau de bonheur : un agent est heureux si et seulement si le pourcentage d'agents voisins différents (ayant une couleur différente) est sous son seuil de tolérance. Si l'agent n'est pas heureux, il va se déplacer sur une case libre de la grille choisie au hasard.

La simulation s'arrête lorsqu'un état stable est atteint, c'est-à-dire quand tous les agents sont heureux.

La simulation peut aboutir à trois résultats différents :
– la simulation ne s'arrête pas car elle ne peut atteindre un état stable où tous les agents sont heureux (typiquement quand la densité d'agents sur la grille est grande et que le seuil de tolérance est bas, c'est-à-dire quand les agents sont très intolérants) ;
– la simulation s'arrête mais on n'observe pas de ségrégation (c'est la cas quand le seuil de similarité est très bas (tolérance forte) et/ou quand la densité est très faible) ;
– des groupes d'habitants de même couleur émergent.

3 Logique dynamique avec affectations

Cette section introduit la syntaxe et la sémantique de la logique $\mathsf{DL}^{\mathsf{PA}}$. Cette logique peut être vue comme une instanciation de la logique dynamique propositionnelle PDL (*Propositional Dynamic Logic*) [11] avec des programmes concrets affectant les valeurs 'vrai' ou 'faux' à des variables propositionnelles.

3.1 Langage

On considère un ensemble dénombrable de variables propositionnelles $\mathbb{P}$ dont les éléments seront notés $p, q, \ldots$ L'ensemble des formules booléennes est construit à partir de $\mathbb{P}$ en utilisant les opérateurs booléens de négation et de disjonction. Les autre connecteurs sont définis comme des abréviations. Dans notre exemple, les formules booléennes pourront représenter des choses telles que : $\bigwedge_{(k,l)\neq(k',l')} \neg(\mathsf{At}(i,k,l) \wedge \mathsf{At}(i,k',l'))$, exprimant le fait qu'un agent i ne peut pas être à la fois sur la case (k, l) et sur la case (k', l').

L'ensemble des *événements* $\mathcal{E}$ est défini par la BNF suivante :

$$\epsilon \quad ::= \quad p{\leftarrow}\top \mid p{\leftarrow}\bot \mid \Phi? \mid \epsilon;\epsilon \mid \epsilon \cup \epsilon \mid \epsilon^{=n} \mid \epsilon^{\leq n} \mid \epsilon^{<\infty}$$

avec p dans $\mathbb{P}$, Φ dans l'ensemble des formules booléennes et n dans l'ensemble des entiers naturels $\mathbb{N}$.

Les événements $p{\leftarrow}\top$ et $p{\leftarrow}\bot$ sont des affectations modifiant la valeur de vérité de la variable propositionnelle p : l'événement $p{\leftarrow}\top$ met p à vrai tandis que l'événement $p{\leftarrow}\bot$ met p à faux. $\Phi?$ est le test de Φ, qui échoue dans le cas où Φ est faux. $\epsilon_1;\epsilon_2$ représente une séquence d'événements. $\epsilon^{=n}$ représente l'itération de ϵ exactement n fois, tandis que $\epsilon^{\leq n}$ représente l'itération de ϵ au plus n fois et $\epsilon^{<\infty}$ représente l'itération de ϵ un nombre arbitraire de fois. Les affectations et les tests sont des événements atomiques tandis que les autres événements sont appelés complexes. Un exemple d'événement complexe est le déplacement de l'agent i d'une position (k, l) à une position (k', l'), représenté par : $\mathsf{At}(i,k,l){\leftarrow}\bot; \mathsf{At}(i,k',l'){\leftarrow}\top$.

L'ensemble des *formules* $\mathcal{F}$ est défini par la BNF suivante :

$$\varphi \quad ::= \quad q \mid \top \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists\epsilon.\varphi$$

avec q dans $\mathbb{P}$ et ϵ dans l'ensemble des évènements $\mathcal{E}$. (On peut noter que l'on utilise Φ pour les formules booléennes et φ pour les formules de $\mathsf{DL}^{\mathsf{PA}}$.) La formule $\exists\epsilon.\varphi$ se lit "il existe une

exécution de l'événement ϵ après lequel φ est vraie". Donc $\exists\epsilon.\top$ doit être lue "ϵ peut se produire".

Les opérateurs '?', ';', '$\cup$' et '$^{<\infty}$' sont les opérateurs habituels de la logique dynamique propositionnelle PDL. (On pourrait aussi bien utiliser la fermeture de Kleene et écrire ϵ^* au lieu de $\epsilon^{<\infty}$, comme d'habitude dans PDL.) Dans la logique PDL non-interprétée, ces opérateurs combinent des programmes atomiques abstraits, tandis que dans PDL interprétée, ils combinent des affectations de variables objet à des valeurs d'un certain domaine. En revanche, les programmes atomiques sont ici des affectations de valeur de vérité à des variables propositionnelles, comme cela a été étudié précédemment dans la littérature sur la logique épistémique dynamique [21, 6, 2].

La formule $\exists(q{\leftarrow}\top \cup q{\leftarrow}\bot).\varphi$ a la même interprétation que la formule booléenne quantifiée (QBF pour *Quantified Boolean Formula*) $\exists q.\varphi$ [10]. Le langage de $\mathsf{DL}^{\mathsf{PA}}$ peut donc être vu comme une généralisation de la quantification sur les variables booléennes à des "programmes de quantification" complexes.

La *longueur* de la formule φ, notée $|\varphi|$, est le nombre de symboles utilisés pour écrire φ ; les symboles '(', ')', les points et les parenthèses ne sont pas comptés. On considère que les entiers ont une longueur de 1 [1]. Par exemple, $|\exists(q{\leftarrow}\top)^{\leq 3}.(q \vee r)| = 1+|q{\leftarrow}\top|+1+1+|q \vee r| = 1+3+2+3 = 9$.

Les opérateurs logiques $\wedge$ et $\rightarrow$ sont définis comme des abréviations ; par exemple, $\varphi \rightarrow \psi$ est l'abréviation de $\neg\varphi \vee \psi$. De plus, $\forall\epsilon.\varphi$ est l'abréviation de $\neg\exists\epsilon.\neg\varphi$. Ainsi $\forall\epsilon.\bot$ doit donc se lire "ϵ ne peut pas se produire".

Remarque 1 *Notons que l'on aurait pu définir les programmes $\epsilon^{=k}$ comme des abréviations de la composition séquentielle de ϵ k fois* [2] *et donc définir $\epsilon^{\leq n}$ comme une abréviation de $\bigcup_{0\leq k\leq n}\epsilon^{=k}$. Cependant le développement de ces abréviations augmenterait de manière exponentielle la longueur des formules. Pour la même raison, nous n'avons pas introduit '$\leftrightarrow$' comme une abréviation.*

1. Plus précisément la longueur d'un entier n devrait être $\log n$. Notre hypothèse est néanmoins sans dommage.

2. La définition précise est une définition par induction : $\epsilon^0 = \mathsf{skip}$ et $\epsilon^{=k+1} = \epsilon^{=k};\epsilon$.

Nous utilisons τ pour représenter soit $\top$, soit $\bot$ et on écrit $q{\leftarrow}\tau$ pour désigner soit $q{\leftarrow}\top$ soit $q{\leftarrow}\bot$ de manière succincte.

3.2 Sémantique

Une *valuation* dans notre logique n'est rien d'autre qu'un modèle de la logique propositionnelle classique, à savoir un sous-ensemble de l'ensemble des variables propositionnelles $\mathbb{P}$.

Les conditions de vérité sont celles habituelles pour $\top$, $\bot$, la négation et la disjonction, complétées par :

$$\begin{array}{lll} V \models p & \text{ssi} & p \in V \\ V \models \exists\epsilon.\varphi & \text{ssi} & \text{il existe } V' \text{ tel que } VR_\epsilon V' \text{ et } V' \models \varphi \end{array}$$

où R_ϵ est une relation binaire de valuation définie par :

$$\begin{array}{lcl} R_{p\leftarrow\top} & = & \{(V,V') \ : \ V' = V \cup \{p\}\} \\ R_{p\leftarrow\bot} & = & \{(V,V') \ : \ V' = V \setminus \{p\}\} \\ R_{\epsilon_1;\epsilon_2} & = & R_{\epsilon_1} \circ R_{\epsilon_2} \\ R_{\epsilon_1\cup\epsilon_2} & = & R_{\epsilon_1} \cup R_{\epsilon_2} \\ R_{\varphi?} & = & \{(V,V) \ : \ V \models \varphi\} \\ R_{\epsilon^{=n}} & = & (R_\epsilon)^n \\ R_{\epsilon^{\leq n}} & = & \bigcup_{0\leq m\leq n}(R_\epsilon)^m \\ R_{\epsilon^{<\infty}} & = & \bigcup_{0\leq m}(R_\epsilon)^m \end{array}$$

Les valuations V' telles que $(V,V') \in R_\epsilon$ sont appelées des *mises à jour possibles de V par ϵ*.

La validité et la satisfiabilité sont définies comme d'habitude.

3.3 Axiomes de réduction pour le fragment sans étoile

Le fragment de $\mathsf{DL}^{\mathsf{PA}}$ sans itération non bornée (appelé "fragment sans étoile" dans PDL) peut être axiomatisé au moyen des axiomes de réduction. Ces axiomes permettent d'éliminer tous les opérateurs dynamiques des formules. Cependant, cette élimination peut provoquer une explosion combinatoire à cause de l'opérateur de composition non-déterministe $\cup$. Nous caractériserons donc par la suite la complexité de la vérification de la validité par d'autres moyens.

Proposition 1 *Les équivalences suivantes sont valides dans $\mathsf{DL}^{\mathsf{PA}}$.*

$$
\begin{array}{lcl}
\exists\epsilon^{=n}.\varphi & \leftrightarrow & \begin{cases}\varphi & \text{if } n=0\\ \exists\epsilon^{=n-1}.\exists\epsilon.\varphi & \text{if } n>0\end{cases}\\
\exists\epsilon^{\leq n}.\varphi & \leftrightarrow & \begin{cases}\varphi & \text{if } n=0\\ \exists\epsilon^{\leq n-1}.(\varphi\vee\exists\epsilon.\varphi) & \text{if } n>0\end{cases}\\
\exists\varphi?.\psi & \leftrightarrow & \varphi\wedge\psi\\
\exists\epsilon_1\cup\epsilon_2.\varphi & \leftrightarrow & \exists\epsilon_1.\varphi\vee\exists\epsilon_2.\varphi\\
\exists\epsilon_1;\epsilon_2.\varphi & \leftrightarrow & \exists\epsilon_1.\exists\epsilon_2.\varphi\\
\exists p{\leftarrow}\tau.\neg\varphi & \leftrightarrow & \neg\exists p{\leftarrow}\tau.\varphi\\
\exists p{\leftarrow}\tau.(\varphi_1\vee\varphi_2) & \leftrightarrow & \exists p{\leftarrow}\tau.\varphi_1\vee\exists p{\leftarrow}\tau.\varphi_2\\
\exists p{\leftarrow}\tau.\top & \leftrightarrow & \top\\
\exists p{\leftarrow}\tau.\bot & \leftrightarrow & \bot\\
\exists p{\leftarrow}\tau.q & \leftrightarrow & \begin{cases}\tau & \text{if } q=p\\ q & \text{if } q\neq p\end{cases}
\end{array}
$$

Ces équivalences fournissent un ensemble complet d'axiomes de réduction pour les opérateurs dynamiques $\exists\epsilon.$ sans $^{<\infty}$. Nous appelons red l'application sur $\mathsf{DL}^{\mathsf{PA}}$ qui applique itérativement les équivalences ci-dessus de la gauche vers la droite, à partir d'un des opérateurs les plus internes. Elle permet d'éliminer dans un premier temps les événements complexes, puis pousse les opérateurs dynamiques à l'intérieur de la formule et enfin les élimine lorsqu'elle rencontre une formule atomique.

Proposition 2 *Soit φ une formule dans le langage $\mathsf{DL}^{\mathsf{PA}}$ privé de l'opérateur $\epsilon^{<\infty}$. Alors*

1. $\mathsf{red}(\varphi)$ *ne contient pas d'opérateurs modaux ;*
2. $\mathsf{red}(\varphi)\leftrightarrow\varphi$ *est valide dans $\mathsf{DL}^{\mathsf{PA}}$;*
3. $\mathsf{red}(\varphi)$ *est valide dans $\mathsf{DL}^{\mathsf{PA}}$ ssi* $\mathsf{red}(\varphi)$ *est valide dans la logique propositionnelle classique.*

La proposition 2 établit une relation forte entre $\mathsf{DL}^{\mathsf{PA}}$ sans étoile et la logique propositionnelle. L'avantage de notre formalisme est de proposer une théorie de modèles et un langage plus intuitif et succinct. Mais les deux sont intéressants du point de vue de la modélisation.

3.4 Complexité

La proposition 2 nous apprend que $\mathsf{DL}^{\mathsf{PA}}$ n'est pas plus expressive que la logique propositionnelle classique. Cependant la réduction peut augmenter exponentiellement la longueur de la formule à cause des opérateurs entre événements $\cup$ et $\epsilon^{\leq n}$. Mais cette procédure est sous-optimale comme nous allons le montrer dans cette section.

Nous commençons par donner le résultat de complexité du problème de model checking. Les entrées de ce problème sont une valuation V et une formule φ et le problème consiste à déterminer si $V\models\varphi$.

Théorème 1 *Le problème du model checking dans $\mathsf{DL}^{\mathsf{PA}}$ est PSPACE-complet.*

Proof. Nous commençons par établir la *difficulté* en réduisant le problème de validité des formules booléennes quantifiées au model checking dans $\mathsf{DL}^{\mathsf{PA}}$. Soit une formule booléenne complètement quantifiée

$$\Phi=\exists q_1\forall q_2\exists q_3\ldots\exists q_{m-1}\forall q_m.\varphi$$

où $m\geq 0$ est pair et $\varphi(q_1,\ldots,q_m)$ est une formule propositionnelle ne contenant pas d'autres variables que $q_1,\ldots,q_m$. (L'hypothèse que le nombre de quantificateurs est pair n'est pas contraignant : il suffit d'ajouter une variable fictive q_m qui n'apparaît pas dans φ.) Nous définissons :

$$\Phi^{\mathsf{DL}^{\mathsf{PA}}}=\exists\epsilon_1.\forall\epsilon_2.\ldots\exists\epsilon_{m-1}.\forall\epsilon_m.\varphi$$

où $\epsilon_k=q_k{\leftarrow}\top\cup q_k{\leftarrow}\bot$, pour $1\leq k\leq m$. Considérons la valuation V sur l'ensemble des variables propositionnelles $\mathbb{P}=\{q_1,\ldots,q_m\}$ telle que $V(q_i)=\mathtt{ff}$ pour tout q_i. Il est facile de vérifier que Φ est valide dans la logique des formules booléennes quantifiées ssi $\Phi^{\mathsf{DL}^{\mathsf{PA}}}$ est vraie dans V. Comme à la fois la taille de $\Phi^{\mathsf{DL}^{\mathsf{PA}}}$ et la taille du modèle sont linéaires en la taille de Φ, nous pouvons conclure que le model checking dans $\mathsf{DL}^{\mathsf{PA}}$ est PSPACE difficile.

Le preuve de l'*appartenance* est semblable à la preuve pour $\mathsf{DL}^{\mathsf{PC}}$ [12]. Elle nécessite néanmoins la définition récursive de l'ensemble des événements atomiques *admis* par un événement complexe ϵ.

$$
\begin{array}{lcl}
adm(p{\leftarrow}\tau) & = & \{p{\leftarrow}\tau\}\\
adm(\Phi?) & = & \{\Phi?\}\\
adm(\epsilon_1;\epsilon_2) & = & \\
\multicolumn{3}{l}{\quad\{\alpha_1;\alpha_2\ :\ \alpha_1\in adm(\epsilon_1)\text{ and }\alpha_2\in adm(\epsilon_2)\}}\\
adm(\epsilon_1\cup\epsilon_2) & = & adm(\epsilon_1)\cup adm(\epsilon_2)\\
adm(\epsilon^{=0}) & = & \{\top?\}\\
adm(\epsilon^{=n+1}) & = & \\
\multicolumn{3}{l}{\quad\{\alpha_1;\alpha_2\ :\ \alpha_1\in adm(\epsilon^{=n})\text{ and }\alpha_2\in adm(\epsilon)\}}\\
adm(\epsilon^{\leq n}) & = & \bigcup_{m\leq n} adm(\epsilon^{=m})\\
adm(\epsilon^{<\infty}) & = & \bigcup_{n} adm(\epsilon^{\leq n})
\end{array}
$$

(Il est clair que l'ensemble $adm(\epsilon)$ est infini ssi ϵ contient $^{<\infty}$). Le point principal de cette preuve est que chaque mise à jour de la valuation V par un événement complexe ϵ peut également être effectuée par une séquence d'événements atomiques qui sont admis par ϵ et qui est de longueur exponentielle en la longueur de ϵ. Ceci est vrai malgré la présence de l'opérateur $^{<\infty}$, du fait que tout $\epsilon^{<\infty}$ ne peut apporter qu'un nombre fini de mises à jour de la valuation. A partir de là, on peut en déduire que l'appartenance d'un couple de valuations (V, V') à R_ϵ peut être évaluée en espace polynomial. Finalement on peut prouver qu'une formule φ peut être évaluée dans un espace polynomial en la taille de φ. En particulier, quand on évalue $\exists\epsilon^{<\infty}.\,\varphi$, on peut décider en espace polynomial si l'une des (exponentiellement nombreuses) valuations V' est accessible depuis V. ■

Théorème 2 *Le problème de vérification de la validité dans* $\mathsf{DL}^{\mathsf{PA}}$ *est PSPACE-complet.*

Proof. La *difficulté* peut être prouvée en transposant les formules QBF dans $\mathsf{DL}^{\mathsf{PA}}$ de la même manière que dans le théorème 1. L'*adhésion* pour le problème de vérification de la satisfiabilité dans $\mathsf{DL}^{\mathsf{PA}}$ peut être prouvée comme suit. Etant donné φ, on considère un modèle V. (V peut être supposé de taille polynomiale car on peut se concentrer sur les variables propositionnelles apparaissant dans φ et négliger les autres.) On vérifie ensuite si $V \models \varphi$ est vraie, ce qui peut être fait en espace polynomial en reproduisant les conditions de vérité. Ceci montre que la satisfiabilité dans $\mathsf{DL}^{\mathsf{PA}}$ peut être vérifiée en NPSPACE. D'après le théorème de Savitch, NPSPACE = PSPACE. On peut donc en conclure que la satisfiabilité dans $\mathsf{DL}^{\mathsf{PA}}$ peut être vérifiée en espace polynomial. On en déduit que le problème complémentaire de la validité dans $\mathsf{DL}^{\mathsf{PA}}$ peut également être vérifié en espace polynomial. ■

Le résultat ci-dessus montre que $\mathsf{DL}^{\mathsf{PA}}$ est plus concise que la logique propositionnelle : il existe des formules de $\mathsf{DL}^{\mathsf{PA}}$ (et même des formules de $\mathsf{DL}^{\mathsf{PA}}$ sans étoile) telles que toute formule propositionnelle équivalente est de taille exponentielle.

4 Le modèle de ségrégation dans $\mathsf{DL}^{\mathsf{PA}}$

En section 2, nous avons introduit le modèle de ségrégation de Schelling de manière informelle. Nous allons maintenant le formaliser dans notre logique.

4.1 Les variables propositionnelles

Nous définissons trois types de variables propositionnelles :

$\mathsf{At}(i,k,l)$	"l'agent i est à la position (k,l)"
$\mathsf{R}(i)$	"l'agent i est rouge"
$\mathsf{Done}(i)$	"l'agent i a déjà été activé dans ce pas de simulation"

avec i un agent de $\mathbb{A} = \{1, \ldots, |\mathbb{A}|\}$ et (k,l) une position définie dans $[1..N] \times [1..N]$.

Nous définissons les abréviations suivantes pour pouvoir décrire de manière plus intuitive le modèle :

$$
\begin{array}{ll}
\mathsf{B}(i) & \stackrel{\mathsf{def}}{=} \neg\mathsf{R}(i) \\
\mathsf{NB}^{<1}(k,l) & \stackrel{\mathsf{def}}{=} \\
 & \bigwedge_i \bigwedge_{k',l' \le N \,:\, |k'-k|,|l'-l| \le 1} \neg\left(\mathsf{At}(i,k',l') \wedge \mathsf{B}(i)\right) \\
\mathsf{NR}^{<1}(k,l) & \stackrel{\mathsf{def}}{=} \\
 & \bigwedge_i \bigwedge_{k',l' \le N \,:\, |k'-k|,|l'-l| \le 1} \neg\left(\mathsf{At}(i,k',l') \wedge \mathsf{R}(i)\right) \\
\mathsf{Hpy}^{<1}(i,k,l) & \stackrel{\mathsf{def}}{=} \\
 & \left(\mathsf{R}(i) \wedge \mathsf{NB}^{<1}(k,l)\right) \vee \left(\mathsf{B}(i) \wedge \mathsf{NR}^{<1}(k,l)\right) \\
\mathsf{Free}(k,l) & \stackrel{\mathsf{def}}{=} \bigwedge_i \neg\mathsf{At}(i,k,l) \\
\mathsf{Segreg}^{<1} & \stackrel{\mathsf{def}}{=} \bigwedge_i \bigvee_{k,l} \left(\mathsf{At}(i,k,l) \wedge \mathsf{Hpy}^{<1}(i,k,l)\right)
\end{array}
$$

La formule $\mathsf{NB}^{<1}(k,l)$ se lit "la case (k,l) n'a aucun voisin bleu", c'est-à-dire qu'il n'y a pas d'agent bleu sur une case adjacente de (k,l) ; la formule $\mathsf{NR}^{<1}(k,l)$ se lit de manière similaire : "la case (k,l) n'a aucun voisin rouge". De plus, l'agent i est heureux sur une case (k,l), noté $\mathsf{Hpy}^{<1}(i,k,l)$, si et seulement si "en (k,l), l'agent i n'a pas de voisin de couleur différente de la sienne". $\mathsf{Free}(k,l)$ se lit "la case (k,l) est libre", ce qui signifie qu'il n'y a pas d'agent sur cette case. Enfin, la propriété de ségrégation, notée $\mathsf{Segreg}^{<1}$, est vérifiée si et seulement si "tous les agents sont heureux dans la case sur laquelle ils sont positionnés".

Nous évaluons maintenant la longueur de ces formules, afin de déterminer la complexité de la vérification de leur validité. La longueur de $\mathsf{NB}^{<1}(k,l)$ et de $\mathsf{NR}^{<1}(k,l)$ est en $O(|\mathbb{A}|)$ (on ne tient pas compte des quantificateurs sur les positions (k',l') dans cette évaluation, car ils déterminent exactement 9 positions en comptant (k,l) et ses voisins) et donc en $O(N^2)$ car $|\mathbb{A}| < N^2$. On obtient le même résultat pour la longueur de $\mathsf{Hpy}^{<1}(i,k,l)$. On en déduit donc que la taille de $\mathsf{Segreg}^{<1}$ est en $O(|\mathbb{A}| \times N^2 \times N^2)$, c'est-à-dire en $O(N^6)$.

4.2 Description du déplacement des agents

Chaque déplacement d'un agent i de la position (k,l) à la position (k',l') peut être décrit par un événement complexe dans notre langage.

$$\begin{array}{lll}\mathsf{move}(i,k,l,k',l') & \stackrel{\mathtt{def}}{=} & \mathsf{At}(i,k,l)?\ ;\ \mathsf{Free}(k',l')?\ ;\\ & & \mathsf{At}(i,k,l)\leftarrow\bot\ ;\ \mathsf{At}(i,k',l')\leftarrow\top\end{array}$$

Le déplacement des agents à chaque pas de la simulation est donc défini comme la composition non-déterministe du comportement de chaque agent non satisfait (auxquels s'ajoutent des formules de gestion du tour).

$$\begin{array}{ll}\mathsf{move} & \stackrel{\mathtt{def}}{=}\\ & \bigcup_{i,k,l}(\neg\mathsf{Hpy}^{<1}(i,k,l)?;\neg\mathsf{Done}(i)?;\\ & \qquad \bigcup_{k',l'}\mathsf{move}(i,k,l,k',l');\mathsf{Done}(i)\leftarrow\top)\ ;\\ & (\top?\cup\neg\bigvee_{i,k,l}(\neg\mathsf{Hpy}^{<1}(i,k,l)\wedge\neg\mathsf{Done}(i))?\ ;\\ & \quad \mathsf{Done}(1)\leftarrow\bot;\ldots;\mathsf{Done}(|\mathbb{A}|)\leftarrow\bot)\end{array}$$

Le choix non-déterministe correspond à l'ordonnanceur présent dans les simulations de ce modèle. Un pas de simulation est composé de $|\mathbb{A}|$ exécutions de move.

Comme pour $\mathsf{Hpy}^{<1}(i,k,l)$, la longueur de la formule $\mathsf{move}(i,k,l,k',l')$ est en $O(N^2)$. La longueur de move est donc en $O(|\mathbb{A}| \times N^4 \times N^2) = O(N^8)$.

4.3 Description du domaine et état initial

Afin de modéliser correctement le modèle de ségrégation, nous devons d'abord décrire les *lois du domaines* exprimant les relations entre les variables propositionnelles $\mathsf{At}(i,k,l)$, $\mathsf{R}(i)$ et $\mathsf{Done}(i)$. Elles spécifient qu'il ne peut pas y avoir deux agents en même temps sur une case (χ_1), qu'un agent ne peut pas être sur deux cases en même temps (χ_2) et qu'un agent est au moins sur une case (χ_3) :

$$\begin{array}{lll}\chi_1 & = & \bigwedge_{i_1\neq i_2,k,l}\neg(\mathsf{At}(i_1,k,l)\wedge\mathsf{At}(i_2,k,l))\\ \chi_2 & = & \bigwedge_{i,(k,l)\neq(k',l')}\neg(\mathsf{At}(i,k,l)\wedge\mathsf{At}(i,k',l'))\\ \chi_3 & = & \bigwedge_i\bigvee_{k,l}\mathsf{At}(i,k,l)\end{array}$$

(Afin de ne pas surcharger les notations, nous laissons implicites que $1 \le k,l \le N$.) Ces trois formules composent les lois du domaine :

$$\mathsf{Laws} = \chi_1 \wedge \chi_2 \wedge \chi_3$$

La longueur de Laws est déterminée par celle de χ_1, qui est en $O(|\mathbb{A}|^2 \times N^2) = O(N^6)$.

L'*état initial* du système est une valuation satisfaisant la formule :

$$\begin{array}{lll}\mathsf{Init} & = & (\bigwedge_{i\in J}\mathsf{R}(i))\wedge(\bigwedge_{i\notin J}\neg\mathsf{R}(i))\\ & & \wedge(\bigwedge_{i\in\mathbb{A}}\neg\mathsf{Done}(i)\wedge\mathsf{At}(i,k_i,l_i))\end{array}$$

avec $J \subseteq \mathbb{A}$ l'ensemble des agents rouges et (k_i,l_i) la position initiale de l'agent i. Sa longueur est en $O(|\mathbb{A}|) = O(N^2)$, et on se convaincra sans peine que $\mathsf{Laws} \wedge \mathsf{Init}$ est satisfiable pour toute répartition des agents sur des cases différentes.

4.4 Exemples de propriétés du modèle pouvant être prouvées

Dans notre langage, on peut exprimer de nombreuses propriétés intéressantes comme "la ségrégation va toujours apparaître après n pas de simulation" (φ_1), "le ségrégation peut apparaître en n pas de simulation" (φ_2), ou "après l'apparition de la ségrégation, aucun agent ne va se déplacer" (φ_3) :

$$\begin{array}{lll}\varphi_1 & = & \forall\mathsf{move}^{=n}.\,\mathsf{Segreg}^{<1}\\ \varphi_2 & = & \exists\mathsf{move}^{\le n}.\,\mathsf{Segreg}^{<1}\\ \varphi_3 & = & \mathsf{Segreg}^{<1}\rightarrow\forall\mathsf{move}.\,\bot\end{array}$$

D'autres propriétés plus générales seront discutées en section 5.

Étant donné une propriété décrite par la formule φ, ce qui nous intéresse est de vérifier si la formule

$$(\mathsf{Laws}\wedge\mathsf{Init})\rightarrow\varphi$$

est valide dans $\mathsf{DL}^{\mathsf{PA}}$ (avec φ une des propriétés précédentes, ou plus généralement une propriété du modèle ou de l'un de ses états que l'on veut démontrer).

La différence entre Init et Laws est que Init ne doit être vraie que dans l'état initial, tandis que Laws doit être vérifiée tout au long de la simulation et donc après chaque mise à jour de l'état du système. Afin d'assurer que notre modélisation est correcte, la première chose à faire est donc de vérifier que les lois du domaine restent vraies après l'exécution de tout événement move. Pour cela, il suffit de prouver que

$$\mathsf{Laws} \rightarrow [\forall \mathsf{move}.\, \mathsf{Laws}]\mathsf{Laws}$$

est valide dans $\mathsf{DL}^{\mathsf{PA}}$ [3].

Il est important de noter que les tailles des formules Laws, $\mathsf{Segreg}^{<1}$, $\mathsf{Hpy}^{<1}(i,k,l)$, Init et de l'événement complexe move sont polynomiales en N. La longueur des formules $\mathsf{Laws} \rightarrow [\forall \mathsf{move}.\, \mathsf{Laws}]\mathsf{Laws}$ et $(\mathsf{Laws} \wedge \mathsf{Init}) \rightarrow \varphi_k$ est donc polynomiale en N et, plus précisément, leur taille est en $O(N^8)$. Le problème de la vérification de la validité dans $\mathsf{DL}^{\mathsf{PA}}$ étant PSPACE, on obtient les résultats suivants.

Proposition 3 *La validité de* $\mathsf{Laws} \rightarrow [\forall \mathsf{move}.\, \mathsf{Laws}]\mathsf{Laws}$ *et de* $(\mathsf{Laws} \wedge \mathsf{Init}) \rightarrow \varphi_k$, *pour* $\varphi_k \in \{\varphi_1, \varphi_2, \varphi_3\}$, *peut être vérifiée en espace polynomial en* N.

Tous nos problèmes de décision étant PSPACE, on peut envisager d'utiliser des démonstrateurs de théorème existant pour des problèmes PSPACE afin de vérifier les propriétés précédentes, tels que les démonstrateurs pour la logique K, pour la logique des descriptions ALC ou pour le logique des formules booléennes quantifiées (QBF). La vérification de ces formules demande leur transformation polynomiale dans la logique du démonstrateur choisi [4].

4.5 Faire varier la tolérance des agents

Les agents décrits jusqu'à présent sont extrêmement intolérants. Des agents plus tolérants peuvent être décrits comme suit :

$$\mathsf{NB}^{<2}(k,l) \stackrel{\text{def}}{=} \bigwedge_{i_1,i_2 \,:\, i_1 \neq i_2} \bigwedge_{(k_1,l_1),(k_2,l_2) \,:\, |k_1-k|,|l_1-l|,|k_2-k|,|l_2-l| \leq 1} \neg(\mathsf{At}(i_1,k_1,l_1) \wedge \mathsf{At}(i_2,k_2,l_2) \wedge \mathsf{B}(i_1) \wedge \mathsf{B}(i_2))$$

$$\mathsf{NB}^{<p}(k,l) \stackrel{\text{def}}{=} \bigwedge_{i_1,\ldots,i_p \,:\, i_m \neq i_n \text{ if } m \neq n} \bigwedge_{(k_1,l_1),\ldots,(k_p,l_p) \,:\, |k_m-k|,|l_m-l| \leq 1} \neg(\mathsf{At}(i_1,k_1,l_1) \wedge \ldots \wedge \mathsf{At}(i_p,k_p,l_p) \wedge \mathsf{B}(i_1) \wedge \ldots \wedge \mathsf{B}(i_p))$$

La formule $\mathsf{NB}^{<p}(k,l)$ (resp. $\mathsf{NR}^{<p}(k,l)$) se lit "la position (k,l) a moins de p voisins bleus (resp. rouges)". $\mathsf{Hpy}^{<1}(i,k,l)$ et $\mathsf{Segreg}^{<1}$ peuvent être généralisés à $\mathsf{Hpy}^{<p}(i,k,l)$ et $\mathsf{Segreg}^{<p}$ de manière évidente.

On pourrait également introduire un pourcentage à la place du nombre d'agents voisins pour caractériser qu'un agent est heureux : un agent serait heureux si le pourcentage d'agents d'une couleur différente dans son voisinage est sous son seuil de tolérance.

La longueur de la formule $\mathsf{NB}^{<2}$ est en $O(|\mathbb{A}|^2)$ et donc en $O(N^4)$. On en déduit donc que la longueur de $\mathsf{Hpy}^{<2}(i,k,l)$ est aussi en $O(N^4)$, que celle de $\mathsf{Segreg}^{<2}$ est en $O(N^8)$ et que celle de move est en $O(N^{10})$. De manière générale, la longueur de $\mathsf{NB}^{<p}(k,l)$ est en $O(|\mathbb{A}|^p)$; le paramètre p valant au plus 8, on en déduit que la longueur de $\mathsf{NB}^{<p}(k,l)$ est en $O(N^{16})$, celle de $\mathsf{Hpy}^{<p}(i,k,l)$ en $O(N^{16})$ également, celle de $\mathsf{Segreg}^{<p}$ en $O(N^{20})$ et celle de move en $O(N^{22})$. Donc, toutes ces propriétés peuvent être vérifiées dans un espace polynomial, tout comme celles de la section 4.4.

5 Langage plus expressif

Dans cette section, nous généralisons notre logique afin de d'exprimer plus naturellement des propriétés supplémentaires que nous souhaitons montrer sur les simulations.

Nous introduisons deux opérateurs modaux supplémentaires $\geq k\ \epsilon$ et $>\frac{1}{2}\ \epsilon.$, où $\geq k\ \epsilon.\varphi$ se lit "φ est vraie dans au moins k des mises à jour possibles par ϵ" et $\geq \frac{1}{2}\ \epsilon.\varphi$ "φ est vraie dans la plupart des états après ϵ". La formule $\exists \epsilon.\varphi$ présentée en section 3 se ramène donc à $\geq 1\ \epsilon.\varphi$.

$V \models \geq k\ \epsilon.\varphi$ ssi
$$|\{V' \,:\, (V,V') \in R_\epsilon \text{ et } V' \models \varphi\}| \geq k$$
$V \models >\frac{1}{2}\ \epsilon.\varphi$ ssi
$$|\{V' \,:\, VR_\epsilon V' \ \&\ V' \models \varphi\}| > |\{V' \,:\, VR_\epsilon V' \ \&\ V' \not\models \varphi\}|$$

3. Si cette formule est valide, on déduit par les principes standards de la logique modale que $\mathsf{Laws} \rightarrow [\forall \mathsf{move}^n.\, \mathsf{Laws}]\mathsf{Laws}$ et $\mathsf{Laws} \rightarrow [\forall \mathsf{move}^{\leq} n.\, \mathsf{Laws}]\mathsf{Laws}$ sont également valides dans $\mathsf{DL}^{\mathsf{PA}}$.

4. Nous savons qu'une telle transformation existe car tous ces problèmes sont dans la même classe de complexité, il nous reste cependant à trouver une transformation élégante.

Ces formules permettent d'exprimer des propriétés intéressantes du modèle de ségrégation telles que :

– "la ségrégation aura lieu en au plus n pas *au moins* k fois" (ψ_1) ;
– "la ségrégation aura lieu au cours de la simulation *au moins* k fois" (ψ_2) ;
– "la ségrégation aura lieu en n pas de simulation *exactement* k fois" (ψ_3) ;
– "la ségrégation aura lieu au cours de la simulation *exactement* k fois" (ψ_4) ;
– "la ségrégation aura lieu après n pas de simulation dans la plupart des cas" (ψ_5).

$$
\begin{array}{lcl}
\psi_1 & = & \geq k\ \mathsf{move}^{\leq n}.\ \mathsf{Segreg}^{<p} \\
\psi_2 & = & \geq k\ \mathsf{move}^{<\infty}.\ \mathsf{Segreg}^{<p} \\
\psi_3 & = & \geq k\ \mathsf{move}^{\leq n}.\ \mathsf{Segreg}^{<p} \\
 & & \wedge \neg \geq k{+}1\ \mathsf{move}^{\leq n}.\ \mathsf{Segreg}^{<p} \\
\psi_4 & = & \geq k\ \mathsf{move}^{<\infty}.\ \mathsf{Segreg}^{<p} \\
 & & \wedge \neg \geq k{+}1\ \mathsf{move}^{<\infty}.\ \mathsf{Segreg}^{<p} \\
\psi_5 & = & > \frac{1}{2}\ \mathsf{move}^{=n}.\ \mathsf{Segreg}^{<p}
\end{array}
$$

La procédure de vérification de modèle (model checking) nécessite de compter le nombre de valuations, mais ceci peut toujours être fait en espace polynomial. Nous en déduisons que le problème de vérification de la validité est PSPACE complet en utilisant le théorème de Savitch de la même manière que nous l'avions fait dans la preuve du théorème 2.

6 Conclusion

Dans cet article, nous avons montré comment *faire* de la simulation sociale dans une logique dynamique avec affectation, test, composition séquentielle et non déterministe et itération bornée et non bornée. Dans la section 5, nous avons montré que notre logique permet d'étudier des propriétés intéressantes du modèle de ségrégation de Schelling. Par exemple, il permet de tester si, étant donné une certaine répartition initiale des agents dans l'espace, la ségrégation apparaîtra à un moment donné dans l'avenir au moins k fois ou exactement k fois. Afin de tester ces propriétés par les méthodes de simulation standard, il serait nécessaire d'effectuer de nombreuses simulations informatiques et de faire ensuite une analyse statistique des résultats observés pour obtenir une estimation de la fréquence à laquelle la ségrégation va émerger à un moment donné de simulations. En ce sens, l'approche proposée dans ce travail offre une nouvelle méthode pour l'étude et l'analyse des résultats de simulations multi-agents.

Nous sommes conscients que, même si le problème de la vérification de la validité des formules telles que ψ_2 ou ψ_4 est PSPACE complet, il pourrait être un problème trop difficile pour des applications réelles. En effet, afin de vérifier si une formule telle que ψ_2 ou ψ_4 est valide, c'est l'arbre entier de toutes les simulations possibles à partir d'un état initial donné qui doit être exploré. Dans de futurs travaux, nous avons l'intention de fournir une version simplifiée (dans le sens où elle nécessite des formules de taille moindre) de notre modélisation formelle du modèle de ségrégation en le représentant sous la forme d'un automate cellulaire : il n'y aura alors plus d'agents et seules les cellules de la grille seront représentées. Dans cette version simplifiée, une cellule de la grille sera caractérisée par une couleur (par exemple bleu, rouge ou grise, pour représenter les cellules non-occupées) et changera de couleur lorsque le pourcentage de cellules qui l'entourent ayant une couleur différente est au-dessus de son seuil de tolérance. Cette cellule deviendra donc grise et une autre cellule grise tirée aléatoirement deviendra de la couleur de la cellule considérée. La simulation s'arrêtera lorsque, pour chaque cellule de la grille, le pourcentage de cellules qui l'entourent et ayant une couleur différente sera en dessous de son seuil de tolérance. Cela permettra de réduire considérablement la taille des formules qui décrivent le modèle de ségrégation et, par conséquent, la complexité du problème de vérifier si la ségrégation apparaîtra après un nombre de pas de simulation donné au moins k fois (ou exactement k fois). Cet exemple met en évidence le fait bien connu qu'une modélisation peut être plus ou moins économique.

A la place de notre logique nous aurions pu également utiliser d'autres approches logiques pour raisonner sur les actions telles que le calcul des situations (*Situation Calculus*) [15], le calcul sur les fluents (*Fluent Calculus*) [19], ou le calcul des événements (*Event Calculus*) [17]. Cependant, alors que tous ces formalismes permettent de représenter plus ou moins les mêmes choses, leur analyse mathématique est moins développée : s'il existe quelques résultats de décidabilité, il n'y a pas de résultat de complexité qui pourrait être comparé à ceux concernant notre logique (PSPACE complétude).

Comme nous l'avons dit dans l'introduction, les propriétés que nous voulons prouver peuvent

être considérées comme des problèmes de planification à horizon fini. Nous aurions donc pu utiliser des planificateurs à horizon fini existants afin de prouver des propriétés des simulations. On doit cependant remarquer que les planificateurs construisent généralement des plans, tandis que nous nous intéressons uniquement à prouver leur existence. Toutefois, il s'agit d'une voie de recherche intéressante à exploiter pour envisager une éventuelle convergence des domaines de la simulation et de la planification.

Remerciements

Ce travail a été financé par le projet RTRA MAELIA. Les auteurs remercient les relecteurs anonymes pour leurs précieux conseils, ainsi que Charles Cultien pour sa relecture et les erreurs qu'il a relevé.

Références

[1] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *Description Logic Handbook*. Cambridge University Press, 2003.

[2] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(204) :1620–1662, 2006.

[3] T. Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69 :165–204, 1994.

[4] R. Conte and M. Paolucci. Responsibility for societies of agents. *J. of Artificial Societies and Social Simulation*, 7(4), 2004.

[5] F. Dignum, B. Edmonds, and L. Sonenberg. Editorial : The use of logic in agent-based social simulation. *J. of Artificial Societies and Social Simulation*, 7(4), 2004.

[6] H. P. van Ditmarsch, W. van der Hoek, and B. Kooi. Dynamic epistemic logic with assignment. In *Proc. of AAMAS'05*, pages 141–148. ACM Press, 2005.

[7] B. Edmonds. How formal logic can fail to be useful for modelling or designing MAS. In G. Lindeman, editor, *Proc. of the International Workshop on Regulated Agent-Based Social Systems : Theories and Applications (RASTA'02)*, LNAI, pages 1–15. Springer-Verlag, 2004.

[8] M. Fasli. Formal systems and agent-based social simulation equals null ? *J. of Artificial Societies and Social Simulation*, 7(4), 2004.

[9] M. Fattorosi-Barnaba and F. de Caro. Graded modalities I. *Studia Logica*, 44 :197–221, 1985.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman Co., 1979.

[11] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, 2000.

[12] W. van der Hoek, D. Walther, and M. Wooldridge. On the logic of cooperation and the transfer of control. *JAIR*, 37 :437–477, 2010.

[13] I. Horrocks. Using an expressive description logic : Fact or fiction ? In *Proc. of KR'98*, pages 636–649, 1998.

[14] H. A. Kautz and B. Selman. Planning as satisfiability. In *Proc. of ECAI'92*, pages 359–363, 1992.

[15] R. Reiter. *Knowledge in Action : Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[16] T. C. Schelling. Dynamic Models of Segregation. *Journal of Mathematical Sociology*, 1 :143–186, 1971.

[17] M. Shanahan. *Solving the frame problem : a mathematical investigation of the common sense law of inertia*. MIT Press, 1997.

[18] P. Taillandier, A. Drogoul, D.A. Vo, and E. Amouroux. GAMA : a simulation platform that integrates geographical information data, agent-based modeling and multi-scale control. In *Proc. of PRIMA'10*, 2010.

[19] M. Thielscher. The logic of dynamic systems. In *Proc. 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, pages 1956–1962, Montreal, Canada, 1995.

[20] W. van der Hoek. On the semantics of graded modalities. *J. of Applied Non-Classical Logics*, 2(1), 1992.

[21] Jan van Eijck. Making things happen. *Studia Logica*, 66(1) :41–58, 2000.

[22] U. Wilensky. Netlogo segregation model. Technical report, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1997.

[23] U. Wilensky. Netlogo. Technical report, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999.