Proposition de sujet de thèse
Spécification d'un système multi-agents par l'analyse d'échecs de preuve

Mots-clefs : Systèmes multi-agents ; Vérification ; Preuve automatique ; Raisonnement ; Logique

La thématique générale de ce projet de thèse est la preuve formelle de systèmes multi-agents. Dans notre contexte, un système multi-agents (SMA) est un système composé d'entités artificielles autonomes en interaction, construit par un « concepteur ». Les objectifs de la conception d'un SMA peuvent être divers : modéliser un phénomène observé, pour comprendre comment il émerge des interactions d'un ensemble d'entités simples (exemple des nuées d'oiseaux [BN13]), résoudre un problème algorithmique complexe (exemple de l'optimisation par « colonies de fourmis » [DB05]), simuler un environnement afin d'anticiper les phénomènes (exemple de la simulation d'évacuation de foules [PHDL07]), etc.

Quel que soit l'objectif, la conception d'un tel système consiste essentiellement à spécifier le comportement de chacun des agents ou types d'agents. C'est toutefois un problème extrêmement difficile, en particulier du fait de la grande diversité d'exécutions possibles de tels systèmes, mais aussi des phénomènes qui émergent des interactions, qui ne sont pas directement spécifiés dans les comportements individuels. Pour cette raison, la communauté a proposé de nombreux langages de spécification [Rao96, 3APL, GOAL], accompagnés d'outils diversifiés [Jason, Jade, Gama].

Dans ce contexte, l'équipe MAD du laboratoire GREYC mène depuis plusieurs années des travaux visant à développer une approche complète de conception, allant de la spécification logique à la production et à l'exécution du code informatique [MSSZ07, MS13, MS17, MS19]. L'originalité de l'approche développée par l'équipe, nommée GDT4MAS (*Goal Decomposition Trees for MultiAgent Systems*) est de proposer un langage de spécification accompagné d'un système de preuve automatique, reposant sur un fragment de la logique du second ordre. Le langage de spécification intègre à la fois les aspects comportementaux des agents, et les propriétés attendues du système. Il permet donc à la fois d'exécuter ce système et, via le système de preuve associé, de vérifier (semi-)automatiquement que toutes les étapes de la conception respectent les propriétés attendues.

La thèse sera encadrée par deux membres de cette équipe, Bruno Zanuttini (PU) en tant que directeur, et Gaële Simon (MCF), en tant que co-encadrante.Les travaux menés dans le cadre de cette thèse viseront à proposer des outils théoriques, et une mise en œuvre de ces outils, pour accompagner le concepteur d'un SMA, potentiellement non spécialiste de la preuve de théorèmes, dans un processus de conception accompagné par la preuve. L'idée générale est de proposer un développement incrémental de la spécification, plutôt qu'une approche directe dans laquelle le SMA est complètement spécifié puis prouvé. En effet, la complexité du processus de conception implique que le premier jet d'une spécification est rarement correct vis-à-vis de l'objectif recherché ; la preuve échoue alors. Dans un développement incrémental de la spécification, nous cherchons à proposer des outils permettant au concepteur d'exploiter ces « échecs de preuve » pour corriger et/ou compléter les spécifications du SMA. En effet, si l'échec de preuve est lié à une spécification erronée, incomplète ou imprécise, il faut pouvoir identifier la ou les parties de la spécification en cause et déterminer comment la/les corriger pour résoudre le problème. Toutes ces tâches sont particulièrement difficiles si l'on n'est pas un spécialiste de la preuve et du prouveur utilisé. Exploiter un échec de preuve implique donc d'être capable de comprendre la raison de cet échec.

À notre connaissance, il n'existe pas de tels outils dans la littérature sur les SMA. En revanche, l'exploitation des échecs de preuve a été en partie étudiée dans la littérature sur la preuve de théorèmes et la vérification de programmes. En effet, la preuve de programmes est utilisée avec succès depuis plus de vingt ans dans l'industrie, pour la vérification de systèmes sécuritaires (systèmes dont dépendent des vies humaines, comme la gestion des feux de circulation sur les voies ferrées). Comme l'approche GDT4MAS, elle repose cependant sur des prouveurs automatiques qui ne peuvent en général pas prouver tous les théorèmes si l'on sort de la logique propositionnelle ; par ailleurs, l'utilisation de prouveurs interactifs comme PVS [ORS92] ou Isabelle [Pau94] nécessitent une grande expertise de la preuve et du prouveur.

C'est pourquoi, dans ce domaine, des travaux visent à aller plus loin en proposant des aides à l'analyse d'échecs de preuve. On trouve par exemple STADY, un outil permettant la génération de tests ciblés, suite à un échec de preuve d'une spécification prenant la forme d'un programme C annoté [PKBGJ16]. Dans le cadre de l'Atelier B [Abr96], un outil appelé ProB, reposant notamment sur un solveur de contraintes, permet de générer un contre-exemple à partir d'un échec de preuve [KBL15]. Cet outil a été intégré à l'Atelier B pour permettre notamment la détection d'inconsistances dans la spécification. Dans le cadre de l'environnement SPARK développé à Inria, il est proposé d'exploiter des contre-modèles générés par des *SMT-Solvers* (*via* l'outil Why3) dans le cadre de la preuve de spécifications de programmes ADA [HMM16]: ces contre-modèles sont transformés en contre-exemples proposés à l'utilisateur sous forme de commentaires insérés au bon endroit dans le code à vérifier.

Le point commun de ces différents types de travaux est la nécessité de proposer à l'utilisateur un ensemble d'informations lui permettant d'analyser des échecs de preuve, de la façon la plus intelligible possible et avec le moins de connaissances possible sur le prouveur. Cela implique de pouvoir exprimer les contre-exemples en utilisant notamment les variables du programme initial (et non celles éventuellement introduites par le processus de preuve). Cela demande également de pouvoir fournir des informations sur la localisation de l'erreur.

Dans certains cas, il peut être envisagé d'aller plus loin en proposant éventuellement des solutions de correction de la spécification. En effet, les techniques précédemment évoquées peuvent être utilisées pour détecter un faux théorème et déterminer les variables concernées par l'erreur. Mais il n'est pas forcément évident de corriger le théorème. Dans le cas où l'erreur dans le théorème est liée à un manque d'hypothèses, des techniques de type abduction peuvent être envisagées pour proposer les hypothèses à ajouter afin de corriger le théorème. Des travaux de ce type ont déjà été envisagés par Monroy et Dennis dans le cadre de preuves par induction [Mon03, DMN06]. D'autres travaux plus généraux, comme le système ABC sur la mise à jour des connaissances d'un agent représentées sous la forme d'une théorie logique [LBS18], ou encore le système EXPLAIN [DD13], permettant de mettre en œuvre un algorithme d'inférence abductive, peuvent être étudiés et adaptés à la correction de théorèmes de la spécification.

Les travaux menés dans cette thèse pourront ainsi s'appuyer sur ces différentes approches de la littérature, pour les exploiter dans le cadre de la conception de SMA. D'un point de vue technique, le caractère innovant, qui constitue le principal verrou scientifique, proviendra de l'articulation entre les spécifications d'un SMA et les théorèmes que l'on essaie de prouver : contrairement au cadre général de la preuve de théorèmes, les échecs de preuve devront permettre de revenir principalement sur la spécification à partir de laquelle ces théorèmes sont générés.

L'approche GDT4MAS, développée dans l'équipe et reposant directement sur la preuve de théorèmes, fournira un cadre particulièrement adapté à ces travaux. Toutefois, la vérification de spécifications formelles par la preuve peut être envisagée sur d'autres langages formels de spécifications [Rao96, 3APL]. Les outils proposés le seront donc, autant que possible, dans un contexte général, permettant leur application à tout système de conception de SMA aidé par la preuve (de théorèmes), déjà développé ou à venir.

Spécificités de l'approche GDT4MAS par rapports aux approches existantes de preuve de SMA

La quasi-totalité des travaux existant dans le domaine de la preuve de systèmes multi-agents reposent sur du *model-cheking* [Rai06]. Le principe général du *model-cheking* est de partir d'une spécification formelle exécutable (comme notre approche) du système à prouver, ainsi que des propriétés qui doivent être vérifiées par le système spécifié. Le *model-checker* va alors vérifier que toutes les traces d'exécution possibles du système vérifient bien les propriétés attendues. Le grand atout de cette approche est qu'elle est entièrement automatique, d'où son grand succès dans le domaine de la preuve de SMA depuis plusieurs années maintenant.

Cependant, cela nécessite d'avoir une spécification suffisamment complète et précise pour être exécutable. De plus, cela implique que le système spécifié puisse se ramener à un nombre fini et raisonnable d'états, ce qui représente l'un des principaux inconvénients de cette approche par rapport à la preuve de théorèmes. Dans le cas particulier des SMA, ces progrès restent également limités car la présence de plusieurs agents dans le système engendre un nombre gigantesque de traces possibles, ce qui ne permet pas d'utiliser ces approches sur des systèmes avec un grand nombre d'agents. Et le nombre d'états finis impose une contrainte forte, ne permettant pas de faire la preuve d'un SMA quel que soit le nombre d'agents présents dans le système. Pour un certain nombre de systèmes néanmoins, le nombre d'agents est connu et figé dès le départ, ce qui rend malgré tout ces techniques utilisables si le nombre d'agents n'est pas trop grand.

L'approche « preuve de théorèmes » utilisée dans le cadre de GDT4MAS repose sur la preuve logique de théorèmes, dont le but est de s'assurer que le comportement du système décrit dans la spécification fournie implique bien logiquement les propriétés attendues (également fournies dans la spécification). Bien sûr, en utilisant cette approche, on peut être confronté à deux problèmes :

- l'incomplétude de Gödel : un théorème vrai n'est pas nécessairement prouvable ; en effet, les obligations de preuve générées par le système de preuve de GDT4MAS sont des théorèmes de la logique du premier ordre dotée de l'arithmétique ; c'est le cadre dans lequel Gödel a montré qu'il existe des théorèmes vrais non prouvables ;
- la semi-décidabilité : il n'existe aucune stratégie automatique permettant de prouver tous les théorèmes vrais ; un théorème vrai peut donc ne pas être prouvé automatiquement par le prouveur...

Cela rend donc en théorie la preuve de théorèmes non automatisable. Néanmoins, dans les cas réels, peu de théorèmes appartiennent aux deux catégories précédentes. De plus, dans l'approche GDT4MAS, une attention particulière a été portée sur le fait de générer des théorèmes dont la preuve peut être prise en charge par des stratégies généralistes et automatiques des prouveurs.

En contrepartie de ces inconvénients, l'immense avantage de cette approche est de permettre une preuve indépendante du nombre d'états du système. Cela la rend très intéressante pour le cas particulier des systèmes multi-agents.

Dans le cadre de l'analyse d'échecs de preuve, l'avantage de l'approche « preuve de théorèmes » se fait encore plus sentir. En effet, avec un *model-checker*, un échec de preuve va correspondre au fait que le prouveur a détecté qu'une trace d'exécution (ou plusieurs) ne satisfait pas une ou plusieurs propriétés attendues. Dans ce cas, le prouveur le signale en fournissant les traces incriminées. Elle peuvent être très longues et il est donc très difficile de les analyser et d'en tirer des enseignements sur les causes de l'échec dans le cas général.

L'encadrement de la thèse permettra d'apporter au (à la) doctorant(e) l'expertise nécessaire sur deux aspects : Bruno Zanuttini, sur le processus abductif et la spécification logique d'actions [NZ08, SNZ21], et Gaële Simon, sur la spécification et la preuve de SMA [MSSZ07, MS13, MS19].

Bibliographie

[3APL] https://www.cs.uu.nl/3apl/
[Abr96] J.-R. Abrial. The B Book. Cambridge University Press, 1996.
[BN13] A. Barve, M. J. Nene. Survey of Flocking Algorithms in Multi-agent Systems, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 6, No 2, November 2013
[DB05] M. Dorigo, C Blum. Ant colony optimization theory: A survey. Theoretical Computer Science, Volume 344, Issues 2–3, 17 November 2005, pages 243-278
[DD13] I. Dillig, T. Dillig. Explain: A Tool for Performing Abductive Inference, 25th International Conference on Computer Aided Verification, 684-689, 2013.
[DMN06] L.A. Dennis, R Monroy, P Nogueira. Proof directed debugging and repair. Seventh Symposium on Trends in Functional Programming, 131-140, 2006.
[Gama] https://gama-platform.org/
[GOAL] https://goalapl.atlassian.net/wiki/spaces/GOAL/overview
[HMM16] D. Hauzar, C. Marché, Y. Moy. Counterexamples from proof failures in the SPARK program verifier. Research Report RR-8854, Inria, 2016.
[Jade] https://jade.tilab.com/
[Jason] https://jason.sourceforge.net/wp/

[KBL15] S. Krings, J. Bendisposto, M. Leuschel. From failure to proof: The PROB disprover for B and Event-B. Proc. Software Engineering and Formal Methods, Lecture Notes in Computer Science, vol. 9276, pp. 199-214, 2015.
[LBS18] X. Li, A. Bundy, A. Smaill. ABC Repair System for Datalog-like Theories, 15th International Conference on Knowledge Engineering and Ontology Development, 333-340, 2018.
[Mon03] R. Monroy. Predicate synthesis for correcting faulty conjectures: The proof planning paradigm. Automated Software Engineering 10 (3), 247-269, 2003.
[MS09] B. Mermet et G.simon. Modélisation, Implantation et Vérification d'agents avec les GDT, https://mermet.users.greyc.fr/Enseignement/CoursPDF/gdts.pdf, 2009.
[MS13] B. Mermet, G. Simon. A new proof system to verify GDT agents, International Symposium on Intelligent Distributed Computing, Prague, Czech Republic. pp.181-187, Sep 2013.
[MS17] B. Mermet, G. Simon. Vers une aide au débuggage des SMA par l'exploitation d'échecs de preuve, Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2017), Caen, France, Jul 2017.
[MS19] B. Mermet, G. Simon. Using proof failures to help debugging MAS. International Conference on Agents and Artificial Intelligence (ICAART 2019), Feb 2019, Prague, France. pp.523-530.
[MS22] B.Mermet et G.Simon. Comment les échecs de preuve peuvent aider à la correction de spécifications erronées de Systèmes Multi-Agents (poster). JFSMA 2022: 145
[MSSZ07] B. Mermet, G. Simon, Arnaud Saval, Bruno Zanuttini. Specifying, Verifying and Implementing a MAS : A Case study. 5th International Workshop on Programing Multi-Agent Systems (ProMAS 2007), 2007, pp.15.
[NZ08] G. Nordh, B. Zanuttini. What Makes Propositional Abduction Tractable. Artificial Intelligence, 2008, pp.1245-1284.
[ORS92] S. Owre, J.M. Rushby, N. Shankar. PVS: A prototype Verification System. 11th International Conference on Automated Deduction (CADE), Vol. 607 in Springer Verlag Lecture Notes in Artificial Intelligence; pp 748-752,1992.
[Pau94] L. C. Paulson . Isabelle: A Generic Theorem Prover. Lecture Notes in Computer Science, vol. 828, 1994.
[PHDL07] X. Pan, C. S. Han, K. Dauber, K. H. Law, « *A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations* », AI & Society, vol. 22, n°2, p. 113-132, 2007
[PKBGJ16] G. Petiot, N. Kosmatov, B. Botella, Alain Giorgetti, J. Julliand. Your proof fails? Testing helps to find the reason, Lecture Notes in Computer Science, Springer, pp.130-150, 2016.
[Rai06] F. Raimondi. Model checking multi-agent systems. Doctoral thesis, University of London. 2006.
[Rao96] A. S. Rao, AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language, Proceedings of Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-96).
[SNZ21] S. Scheck, A. Niveau, B. Zanuttini. Knowledge Compilation for Nondeterministic Action Languages. International Conference on Automated Planning and Scheduling, Aug 2021, Guangzhou, China.