Apprentissage Issu de la Communication pour des Agents Cognitifs

Clement Jonquet, Stefano A. Cerri

Apprentissage issu de la communication pour des agents cognitifs

Clément Jonquet — Stefano A. Cerri

LIRMM - Université Montpellier II
161, rue Ada
F-34392 Montpellier Cedex 5
{cerri, jonquet}@lirmm.fr

RÉSUMÉ. *La communication entre Agents cognitifs est un domaine de recherche en pleine effervescence. Nous proposons ici un modèle, basé sur le modèle STROBE, qui considère les Agents comme des interpréteurs Scheme. Ces Agents sont capables d'interpréter les messages d'une conversation dans un environnement donné, avec un interpréteur donné, tous les deux dédiés à la conversation courante. Ces interpréteurs peuvent en outre évoluer dynamiquement au fur et à mesure des conversations et représentent la connaissance de ces Agents au niveau méta. Nous proposons un mécanisme d'apprentissage à ce méta-niveau basé sur la communication. Ce court article a pour but de présenter brièvement notre modèle et ses intérêts pour des domaines tels que le Web, la génération de services, le GRID etc...*

ABSTRACT. *Cognitive Agent communication is a field of research in full development. We propose here a model, based on the STROBE model, which regards the Agents as Scheme interpreters. These Agents are able to interpret messages of a conversation in both a given environment and an interpreter dedicated to the current conversation. These interpreters can moreover evolve dynamically progressively with the conversations and thus they represent meta level Agent knowledge. We propose a learning-by-communicating device to that level. This short paper presents briefly our model and its interest for domains such as the Web, services generation, the Grid etc...*

MOTS-CLÉS : *communication agent, apprentissage par communication, interprétation dynamique, reifying procedures, STROBE, langage de communication agent.*

KEYWORDS: *agent communication, learning by communicating, dynamic interpretation, reifying procedures, STROBE model, ACL.*

RSTI/hors série. JFSMA 2003, pages 83 à 87

1. Introduction

La communication est la base des systèmes à Agents cognitifs. Définir et modéliser la communication a toujours été difficile. Aujourd'hui, il existe de nombreux modèles et langages de communication mais pouvons nous dire qu'ils sont adaptés au monde Agent ou à de nouvelles formes de communication comme celles que propose le Web ? Il ne s'agit pas de prendre les langages traditionnels de communication ou même les paradigmes actuels de programmation et de les adapter au Web et aux Agents. **Il s'agit de développer de nouvelles architectures et de nouveaux langages conçus pour le Web et les Agents**. Ainsi, nous allons expliciter ici notre modèle de communication Agent, basé sur le modèle STROBE [CER 99][CER 00], qui considère les Agents comme des interpréteurs Scheme. Ces Agents sont capables d'interpréter des messages dans un environnement donné incluant un interpréteur qui apprend par les conversations. Ces environnements sont dédiés à une conversation et les interpréteurs peuvent, en outre, évoluer dynamiquement au fur et à mesure des conversations. Ils représentent la connaissance de ces Agents au niveau méta. Nous proposons un mécanisme d'apprentissage[1] par modification dynamique de ces couples environnement/interpréteur.

Notre travail se base sur deux pré-requis : Si nous considérons le fait qu'une communication a des effets sur ses interlocuteurs (acte perlocutoire), alors il nous faut obligatoirement considérer que les Agents peuvent changer de but ou de point de vue au milieu de cette communication. **Ils doivent donc être autonomes et s'adapter pendant la communication** [CER 99]. Il faut aussi considérer que des Agents peuvent interagir entre eux ou avec des humains suivant les mêmes principes [CER 00]. Ce qui compte c'est **la représentation qu'un Agent se fait de son interlocuteur**. Dans le soucis du respect de ces deux pré-requis, nous expliquerons, dans un premier temps, comment considérer les Agents comme des interpréteurs permet de les voir comme des entités autonomes dynamiquement modifiables, et dans un second temps comment gérer leurs représentations des autres. Finalement, nous présenterons brièvement les travaux réalisés avec ce modèle ainsi que ses intérêts et extensions pour des domaines tels que le Web, la génération de services, le GRID etc...

2. Les Agents comme des interpréteurs Scheme

Le modèle que nous proposons a pour caractéristique principale le fait qu'il considère les Agents comme des interpréteurs. Cette idée est issue de STROBE qui s'inspire de la boucle classique d'évaluation et considère les Agents comme des interpréteurs de type REPL (Read, Eval, Print, Listen). Lors d'une communication, chaque Agent exécute une boucle REPL qui sont imbriquées les unes dans les autres. Ce principe est important car il permet de considérer les Agents comme des entités autonomes dont les interactions et le comportement sont dirigés par une procédure concrète, la procédure correspondant à l'interpréteur (fonction `evaluate`). En conséquent, les interpréteurs représentent la connaissance d'un Agent et son évolution dans le temps. Notre modèle

1. Nous pouvons également parler dans notre cas d'échange de connaissances.

utilise Scheme aussi bien pour le contenu des messages que pour leur représentation. De cette façon nous pouvons utiliser le même interpréteur pour évaluer le message et son contenu. Scheme est très pratique car il explicite bien les trois niveaux d'apprentissage ou de représentation des connaissances que nous pouvons retrouver dans tous les langages : L'apprentissage au niveau *donnée* consiste à affecter des valeurs à des variables déjà existantes ou à définir de nouvelles données (exemple : `(set! a 4)` ou `(define a 3)`) ; Au niveau *contrôle*, consiste à définir de nouvelles fonctions par abstractions sur celles déjà existantes (exemple : `(define (square x) (* x x))`). Et enfin, l'apprentissage au niveau *interpréteur* ou méta-niveau consiste à faire évoluer directement l'interpréteur correspondant à l'Agent (exemple : rajouter une forme spéciale). Notre travail propose un mécanisme d'apprentissage à ces trois niveaux et en particulier au troisième.

3. Représentation des autres

Un des principes de STROBE est d'avoir un « modèle du partenaire », c'est la notion d'Environnement Cognitif. Notre travail exploite cette notion, il se greffe dessus car, comme le concept d'Environnement Cognitif fournit aux Agents un environnement global (ou privé) et plusieurs environnements locaux de représentation des autres, notre proposition fournit aux Agents non pas un interpréteur mais plusieurs dont un global (ou privé) et un pour chaque Agent dont ils ont une représentation. Ainsi, l'évaluation des messages d'une conversation se fait avec un interpréteur donné dans un environnement donné. Notre travail se greffe sur ce concept d'Environnement Cognitif puisque, pour être accessibles, ces interpréteurs doivent eux-mêmes être stockés dans ces environnements ! Nos Agents possèdent les trois attributs suivants :

– **GlobalEnv** leur environnement global.
– **GlobalInter** leur interpréteur global.
– **Other** = (name, interpreter environment) un ensemble de triplet correspondant aux représentations des autres.

Leur environnement global est privé et ne change pas. C'est cet environnement qui est dupliqué[2] lors de l'arrivée d'une nouvelle conversation et c'est son clone, stocké dans un élément de Other, qui est modifié au fur et à mesure de la conversation. Pour chacun de ses interlocuteurs un Agent a une représentation spécifique de celui-ci, ce qui lui permet de tenir compte de ce qu'il apprend tout en gardant son comportement et ses croyances d'origine intactes (ce qui respecte certaines conditions minimales de sécurité).

Pour assurer l'apprentissage à l'issue de la communication, nous utilisons les outils suivants (que nous ne détaillons pas ici) qui vont nous permettre de modifier dynamiquement, au fur et à mesure des conversations, les interpréteurs d'un Agent : Un

2. Pas forcément d'ailleurs si nous considérons un Agent qui veut reprendre une conversation dans le contexte d'une autre déjà existante ou ayant existé (avec son environnement et son interpréteur).

méta-évaluateur Scheme[3] et le mécanisme des *reifying procedures* qu'il propose permettant à un programme d'accéder à son contexte d'évaluation. Permettant donc à un interpréteur de se modifier lui-même. De plus, pour faire communiquer nos Agents nous avons également défini un petit protocole de communication basé sur les actes de langages et utilisant des performatifs.

4. Expérimentations, intérêts et extensions de ces principes

Nous présentons dans [JON 03] une expérimentation de dialogue de type « professeur - élève » où un Agent *student* apprend un nouveau performatif par la communication avec un Agent *teacher* qui lui enseigne. A l'issue du traitement du dernier message le *student* a modifié sa fonction `evaluate-kqmlmsg`. Le code correspondant à cette fonction dans son environnement dédié à cette conversation est changé grâce à l'apprentissage et à l'application d'une *reifying procedure*. Il est alors apte à traiter les messages indexés par ce nouveau performatif. Dans cet exemple, un Agent modifie sa fonction d'évaluation des messages (`evaluate-kqmlmsg`) mais les même principes peuvent être utilisés pour modifier n'importe quelle partie de l'interpréteur d'un Agent. Grâce à ce protocole, **nos Agents possèdent un ensemble d'interpréteurs qui représentent leurs connaissances, ils correspondent aux sous-langages que nos Agents reconnaissent et donc à leurs facultés à effectuer une tâche**. Les Agents peuvent accomplir des tâches pour d'autres, voir même s'échanger leurs interpréteurs comme dans les architectures GRID où il est plus intéressant de déplacer les processus que les données. En outre, ces interpréteurs peuvent même être transmis avant une conversation, comme est transmise aujourd'hui une ontologie. L'ontologie devient intrinsèque à la communication. Ces principes sont particulièrement intéressants pour le Web, où un Agent peut s'intégrer rapidement à une communauté en enseignant les performatifs qu'il utilise[4]. Un autre papier soumis [JON 04] décrit comment considérer les Agents comme des interpréteurs non déterministes[5] permet de spécifier dynamiquement un problème. Ceci est idéal pour la génération dynamique de services sur le Web (et bientôt sur le GRID). Le papier présente un exemple d'application réelle, un scénario de type e-commerce où un Agent *SNCF* spécifie et code dynamiquement une fonction de recherche de billet de train pour un Agent *Client* au fur et à mesure de la conversation.

Nous sommes également en train de travailler sur des utilisations de ces principes pour le *Grid Computing* ou des protocoles de communication type *contract net*. Une analogie avec XML peut être également faite en faisant évoluer dynamiquement des « interpréteurs » XML (programme XSL, DTD...) et donc les documents associés.

Ce modèle est donc très intéressant pour de nombreux domaines. Il s'inscrit cependant dans une logique globale de l'intelligence artificielle ; Il faudrait lui rajouter par

3. Celui de Jefferson et Friedman [JEF 92] auquel nous rajoutons une procédure d'évaluation des messages `evaluate-kqmlmsg`.
4. Ce modèle suppose que les Agents sont construits suivant une architecture minimale commune. Par contre, ils évoluent de manière différente suivant leurs conversations.
5. Avec un évaluateur non déterministe [ABE 96] une expression peut avoir plusieurs valeurs possibles, cela est très utile pour la programmation par contraintes.

exemple un modèle de raisonnement classique lui permettant de décider quand transférer une information d'un environnement local à son environnement global, de façon à réutiliser ce qu'il apprend lors d'une conversation dans une autre. En outre, nous travaillons actuellement sur l'amélioration de ce domaine via l'utilisation des environnements de première classe [QUE 96] qui permettent de modifier dynamiquement des environnements lexicaux sans utiliser la réflexivité.

5. Conclusion

Nous avons essayé de brièvement présenter dans ce papier une méthode d'apprentissage, pour des Agents cognitifs, issue de la communication. Cet apprentissage peut se faire par communication simple (niveau *donnée* et *contrôle*), ou par modification interne de l'Agent (niveau *interpréteur*). Si les Agents interprètent de façon dynamique les messages qu'ils reçoivent, ils deviennent adaptables et, sans aucune intervention extérieure, peuvent communiquer avec des entités qu'ils n'ont jamais rencontrées auparavant. Ce papier n'a pas simplement pour vocation de proposer un artefact de plus de programmation à ajouter aux Agents, mais l'idée est plutôt de montrer une technique, faite de manière simple et utilisable[6], d'évolution autonome des Agents dans une société.

6. Bibliographie

[ABE 96] ABELSON H., SUSSMAN G. J., SUSSMAN J., *Structure and Interpretation of Computer Programs*, MIT Press, Cambridge, Massachusetts, 2nd édition, 1996.

[CER 99] CERRI S. A., « Shifting the Focus from Control to Communication : the STReams OBjects Environments Model of Communicating Agents », *In Padget, J.A. (ed.) Collaboration between Human and Artificial Societies*, Berlin, Heidelberg, 1999, New York : Springer-Verlag, Lecture Notes in Artificial Intelligence, p. 74-101.

[CER 00] CERRI S. A., SALLANTIN J., CASTRO E., MARASCHI D., « Steps towards C+C : A Language for Interactions », *AIMSA2000*, Berlin, Heidelberg, 2000, New York : Springer-Verlag, Lecture Notes in Artificial Intelligence, p. 34-48.

[JEF 92] JEFFERSON S., FRIEDMAN D. P., « A Simple Reflective Interpreter », *IMSA'92 International Workshop on Reflection and Meta-level architecture*, Tokyo, novembre92.

[JON 03] JONQUET C., CERRI S. A., « Cognitive Agents Learning by Communicating », *Actes du Colloque Agents Logiciels, Coopération, Apprentissage et Activité Humaine, ALCAA'03*, Bayonne, septembre2003, p. 29-39.

[JON 04] JONQUET C., CERRI S. A., « Agents as Scheme Interpreters : Enabling Dynamic Specification by Communicating », *Soumission au congrès Reconnaissance des Formes et Intelligence Artificielle, RFIA'04*, Toulouse, janvier2004.

[QUE 96] QUEINNEC C., DE ROURE D., « Sharing Code through First-class Environments », *Proceedings of ICFP'96 — ACM SIGPLAN International Conference on Functional Programming*, Philadelphia, Pennsylvania, 1996, p. 251–261.

6. Le modèle présenté ici a été sujet à une petite implementation, non complète encore à ce jour, mais d'ores et déjà fonctionnelle (`www.lirmm.fr/~jonquet`).