Type Checking for Protocol Role Enactments via Commitments[*]

JAAMAS Track

Matteo Baldoni
Università di Torino, Dip. Informatica
matteo.baldoni@unito.it

Cristina Baroglio
Università di Torino, Dip. Informatica
cristina.baroglio@unito.it

Federico Capuzzimati
Università di Torino, Dip. Informatica
federico.capuzzimati@unito.it

Roberto Micalizio
Università di Torino, Dip. Informatica
roberto.micalizio@unito.it

KEYWORDS

Agent Typing; Social Relationships; Static and dynamic type checking; Commitments; Commitment-based Interaction Protocols.

ACM Reference Format:

Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, and Roberto Micalizio. 2019. Type Checking for Protocol Role Enactments via Commitments. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

The major contribution of [8] is an agent-based, dynamic, and declarative *type checking system* for agent interactions, which is meant to support the implementation of Multi-Agent Systems (MAS). We observed that MAS abstractions and means (e.g., the possibility to encompass heterogeneous, autonomously developed agents, which operate in a same environment, and contribute to goal achievement), suit well the characteristics of the present IT applications, like the presence of global, pervasive software infrastructures, where computing is ubiquitous, with embedded and distributed devices interacting with each other. However, there is a lack of effective tools for reasoning on properties of the MAS which is being realized. Our focus is on *interaction* [21]: How can an agent designer verify that an agent has the means for carrying out the encoded interaction? How to decide whether the agent is capable of behaving in a certain way, or whether it shows specific skills/properties?

We claim that the answer is *agent typing*. Typing provides abstractions to perform sophisticated forms of program analysis and verification: compile-time/run-time error checking, modeling, documentation, verification of conformance and of compliance, reasoning about programs and components. It allows light forms of (a priori/runtime) component verification [1–3, 6, 32, 34, 36]. Which characteristics should the conceptualization and realization of a typing system for interacting agents consider? Briefly, we propose the following: (a) *natural to programmers*: it should rely on notions that are the basic building blocks of the agent paradigm, such as those of behaviors and goals; (b) *declarative*: to accommodate the agents' autonomy by being the least prescriptive; (c) *dynamic type checking*: because agents may acquire or lose capabilities along time; (d) *distributed*: or local, it should rely on information possessed by the agent. The agent typing we presented is the first in the literature to show all such characteristics. It has the ultimate purpose of allowing (at *role-enactment time* [13, 14, 16, 20]) the verification that agents satisfy the requirements for the protocol roles they mean to enact. Building upon a wide literature that considers relationships as one of the basic building blocks of the human way of interpreting reality, the proposal is grounded on the *social relationships* which agents may create along their interaction when playing *protocol roles.* We represent such social relationships in terms of *social commitments* [17, 35] and, thus, rely on commitment-based protocols.

The proposal is not bound to a specific agent programming language; rather, it can be implemented in different frameworks. As an exemplification, the typing system and the checking performed at enactment-time were implemented in 2COMM4JADE [5].

2 TYPING AGENTS THROUGH SOCIAL RELATIONSHIPS

An agent can deliberate to enact a protocol role at any time of its execution. Thus, a type checking system should verify, dynamically, that only an agent adequately equipped to play the selected role is actually enabled to enact that role. In other words, the type checking should prevent an agent from playing a role when it has not a proper set of behaviors to carry out the duties associated with the role. Of course, the type checking is grounded on the assumption that the underlying protocol is "well-defined" (correct).

We focus on commitment protocols and, based on the definitions of *residuation* and *progression* (see journal paper) we have singled out the class of *socially progressive* protocols as adequate for supporting the agent type checking at enactment time. These protocols satisfy the following properties: (a) *closeness*: in a closed protocol, for any relevant event, there is at least one role that can make it occur, and all events generated by the protocol are relevant to some commitment generated by the same protocol [9]; (b) *role-distinctness*: for each commitment in the protocol, the events occurring in the consequent condition amount to the powers of the debtor, and similarly, the events occurring in the antecedent condition amount to the powers of the creditor [30, 31]; (c) *coordinability*: for any pair of commitments in the protocol there exists at least one sequence of events that satisfies both commitments.

An important property, proved in [8], is that a socially-progressive protocol puts the agents, playing its roles, in condition to discharge

[*]This paper is an extended abstract of an article published in Autonomous Agents and Multi-Agent System [8].

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. © 2019 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

all the commitments that they may take along their interaction, i.e., to satisfy one such commitment without violating another. The key problem, now, is verifying that when an agent tries to enact a role in a socially-progressive protocol, that agent has the capabilities for accomplishing the role's duties. This is right the goal of our typing system, which aims at verifying that an agent implementation produces the proper subset of the relevant events in response to commitment progressions. An important consequence is that an agent's behavior can be programmed as a reaction to a change in the social state. Another consequence is that, since socially-progressive protocols are role-distinct, each agent can individually and locally check its capacity to play a protocol role – by being able to make the commitments, it may be involved into, progress. Of course, having the capacity to satisfy a commitment does not mean that at runtime agents will never violate their commitments. Their autonomy in deliberating which actions to execute is not reduced in any way.

To type agents, we introduce the notion of *behavioral type* (*b-type*) as a way to annotate each behavior that an agent exposes at the role-enactment time. A *b-type* is a set of type expressions, each consisting of a pair: a commitment c and a temporal expression $\vec{e} + e'$, meaning that, in a context where $\vec{e}$ has occurred, the typed behavior will bring about the event e' which will make commitment c progress one step further towards either satisfaction or detachment.

Type checking at role-enactment time is as follows. When an agent tries to enact role x from a socially-progressive protocol P, the agent declares the set of behaviors it will use while playing x, together with their *b-types*. The checking system verifies the suitability of the agent to play x by assessing whether the agent is both *debtor-compliant* and *creditor-compliant*. An agent is *debtor-compliant* if for each commitment c that may occur in P where x appears as debtor, the set of behaviors offered by the agent is such to generate an actualization for the consequent condition of c, meaning that these behaviors can bring c to satisfaction. An agent is *creditor-compliant* if for each commitment c that may occur in P where x appears as creditor, the behaviors declared by the agent are either an actualization of the antecedent condition of c (i.e., they detach c), or they perform an explicit *release* of c, letting the commitment expire. Intuitively, an agent, that is debtor compliant w.r.t. a commitment, has a set of behaviors that, when executed in the proper order, will allow it to make the commitment progress from the Detached to the Discharged state. In other words, that agent is equipped with an implementation to deal with its obligations [24]. For enacting a protocol role, an agent must be debtor-compliant with all the commitments where such a role appears as debtor.

In general, a commitment that can neither be detached nor released by an agent is a symptom that the agent program is incomplete [28, 29]. The proposed typing system allows devising the type checking in such a way that it identifies such situations, that can, thus, be raised to the attention of the programmer for they may amount to some programming mistake. To avoid being too strict, and in particular to force creditors to include implementations for activating commitments, in which they are not interested, we ask the programmer to include in the creditors programs at least a behavior for releasing such commitments. We proved [8] that when all roles of a socially-progressive protocol are plaid by debtor- and creditor-compliant agents, the role players can produce at least one course of events that satisfies any commitment they created.

3 DISCUSSION

The key characteristic of the proposed typing system is its being based on notions that are typical of agents. Specifically, it relies on the direct use of relationships among agents, intended as first-class entities. As such, the proposal represents a novelty w.r.t. previous work on agent typing that, by relying on functional types [19, 26, 27, 33], do not fully meet the needs of agent programmers. The paper also describes an implementation in the context of the 2COMM framework [5, 7] and Jade [15]. 2COMM enables programming social relationships by exploiting a declarative, interaction-centric approach. The social relationships that arise along the interaction among agents are captured as social commitments while interaction is mediated by protocol artifacts. The choice of commitments is motivated by the desire of typing agents and roles in a way that results minimally prescriptive, so to preserve the autonomy of the agents as far as possible.

The type checking we have proposed can be generalized by leveraging the notions of *accountability* and *responsiblity*. Indeed, some recent works [10, 18, 22] have started to address the problem of designing and developing complex, distributed systems that ground on the notions of responsibility and accountability. Accountability relationships are promising tools for the design of a system where control is distributed over a set of interacting agents because they both (1) support checking that the agents altogether provide the desired control skills over the situation to be managed, and (2) "connect" the needed, distributed control over the goal in a way that enables its achievement. In this, there is surely a similarity between the proposed commitment-based typing system, on the one hand, and accountabilities and responsibilities, on the other. What makes approaches based on accountability and responsibility go one step further is that not only they provide a structure for coordinating responsibility assumption by the agents, but they also introduce mechanisms for *account giving*. An accountable agent is one that is devised in such a a way that it provides an account of its conduct. Following [23], when the agent behaved correctly, the account, or proof, is evident by way of how the agent has operated in, and changed, the environment. Instead, when the agent did not reach the agreed "standard of performance" [25], it will provide an explicit account of what done. Account exchange, composition, and management allows the system, made of interacting parties, to redress failures, thus becoming more robust. ADOPT [11, 12] is an example of how accountability can generalize agent type checking. In ADOPT, each agent declares, at enactment time, the powers it means to use within an organization to bring about its commitments towards other members. This step is similar to a type declaration of the agent. Further interaction between the agent and the organization brings to an agreement about the goals the agent may be asked to bring about. When a commitment is violated it is possible to indentify which agent has to give an account. As a concluding remark, it is worth noting that accountability and responsibility can also be used as a means for the specification of organizations. As shown in [4], the accountability property can be a byproduct of the design process. Relying on such a specification, it would be possible to develop agent programming patterns with the goal of supporting the programmer in developing agents that, by construction, pass the type checking while being accountable.

REFERENCES

[1] Davide Ancona, Daniela Briola, Amal El Fallah-Seghrouchni, Viviana Mascardi, and Patrick Taillibert. 2014. Efficient Verification of MASs with Projections. In *Engineering Multi-Agent Systems - Second International Workshop, EMAS 2014, Paris, France, May 5-6, 2014, Revised Selected Papers (Lecture Notes in Computer Science)*, Fabiano Dalpiaz, Jürgen Dix, and M. Birna van Riemsdijk (Eds.), Vol. 8758. Springer, 246–270. https://doi.org/10.1007/978-3-319-14484-9_13

[2] Davide Ancona, Daniela Briola, Angelo Ferrando, and Viviana Mascardi. 2015. Global Protocols as First Class Entities for Self-Adaptive Agents. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, Gerhard Weiss, Pinar Yolum, Rafael H. Bordini, and Edith Elkind (Eds.). ACM, 1019–1029. http://dl.acm.org/citation.cfm?id=2773282

[3] Davide Ancona, Sophia Drossopoulou, and Viviana Mascardi. 2013. Automatic Generation of Self-monitoring MASs from Multiparty Global Session Types in Jason. In *Declarative Agent Languages and Technologies X*, Matteo Baldoni, Louise Dennis, Viviana Mascardi, and Wamberto Vasconcelos (Eds.). Lecture Notes in Computer Science, Vol. 7784. Springer Berlin Heidelberg, 76–95. https://doi.org/10.1007/978-3-642-37890-4_5

[4] Matteo Baldoni, Cristina Baroglio, Olivier Boissier, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. 2018. Accountability and Responsibility in Agents Organizations. In *PRIMA 2018: Principles and Practice of Multi-Agent Systems, 21st International Conference (Lecture Notes in Computer Science)*, T. Miller, N. Oren, Y. Sakurai, I. Noda, T. Savarimuthu, and Tran Cao Son (Eds.). Springer, Tokyo, Japan, 403–419. http://dx.doi.org/10.1007/978-3-030-03098-8_16

[5] Matteo Baldoni, Cristina Baroglio, and Federico Capuzzimati. 2014. A Commitment-Based Infrastructure for Programming Socio-Technical Systems. *ACM Transactions on Internet Technology* 14, 4, Article 23 (Dec. 2014), 23 pages. https://doi.org/10.1145/2677206

[6] Matteo Baldoni, Cristina Baroglio, and Federico Capuzzimati. 2014. Typing Multi-Agent Systems via Commitments. In *Post-Proc. of the 2nd International Workshop on Engineering Multi-Agent Systems, EMAS 2014, Revised Selected and Invited Papers (LNAI)*, F. Dalpiaz, J. Dix, and M. B. van Riemsdijk (Eds.). Springer, 388–405.

[7] M. Baldoni, C. Baroglio, F. Capuzzimati, and R. Micalizio. 2018. Commitment-based Agent Interaction in JaCaMo+. *Fundamenta Informaticae* 157 (2018), 1–33.

[8] Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, and Roberto Micalizio. 2018. Type Checking for Protocol Role Enactments via Commitments. *Journal of Autonomous Agents and Multi-Agent Systems* 32, 3 (May 2018), 349–386.

[9] Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, and Munindar P. Singh. 2015. Composing and Verifying Commitment-Based Multiagent Protocols. In *Proc. of 24th International Joint Conference on Artificial Intelligence, IJCAI 2015*, M. Wooldridge and Q. Yang (Eds.). Buenos Aires, Argentina. http://ijcai-15.org/

[10] Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. 2016. Computational Accountability. In *Deep Understanding and Reasoning: A challenge for Next-generation Intelligent Agents, URANIA 2016*, F. Chesani, P. Mello, and M. Milano (Eds.), Vol. 1802. CEUR, Workshop Proceedings, Genoa, Italy, 56–62. http://ceur-ws.org/Vol-1802/

[11] Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. 2017. ADOPT JaCaMo: Accountability-Driven Organization Programming Technique for JaCaMo. In *PRIMA 2017: Principles and Practice of Multi-Agent Systems, 20th Int. Conf. (LNCS)*, A. Bo, A. Bazzan, J. Leite, L. van der Torre, and S. Villata (Eds.). Springer, Nice, France, 295–312.

[12] Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. 2018. Computational Accountability in MAS Organizations with ADOPT. *Journal of Applied Sciences, special issue "Multi-Agent Systems"* 8, 4 (March 2018), 489. http://www.mdpi.com/2076-3417/8/4/489/html

[13] M. Baldoni, G. Boella, and L. van der Torre. 2006. powerJava: Ontologically Founded Roles in Object Oriented Programming Languages. In *Proc. of 21st ACM Symposium on Applied Computing, SAC 2006, Special Track on Object-Oriented Programming Languages and Systems, OOPS 2006*, D. Ancona and M. Viroli (Eds.). ACM, Dijon, France, 1414–1418.

[14] Matteo Baldoni, Guido Boella, and Leon van der Torre. 2007. Interaction between Objects in powerJava. *Journal of Object Technology, Special Issue OOPS Track at SAC 2006* 6, 2 (2007).

[15] Fabio Bellifemine, Federico Bergenti, Giovanni Caire, and Agostino Poggi. 2005. JADE - A Java Agent Development Framework. In *Multi-Agent Programming: Languages, Platforms and Applications*, R. H. Bordini, M. Dastani, J. JDix, and A. El Fallah-Seghrouchni (Eds.). Multiagent Systems, Artificial Societies, and Simulated Organizations, Vol. 15. Springer, 125–147.

[16] Guido Boella and Leendert W. N. van der Torre. 2007. The Ontological Properties of Social Roles in Multi-Agent Systems: Definitional Dependence, Powers and Roles playing Roles. *Artificial Intelligence and Law Journal (AILaw)* 15, 3 (2007), 201–221.

[17] Cristiano Castelfranchi. 1997. Principles of Individual Social Action. In *Contemporary Action Theory: Social Action*, G. Holmstrom-Hintikka and R. Tuomela (Eds.), Vol. 2. Kluwer, Dordrecht, 163–192.

[18] Amit K. Chopra and Munindar P. Singh. 2016. From social machines to social protocols: Software engineering foundations for sociotechnical systems. In *Proc. of the 25th Int. Conf. on WWW.*

[19] Ferruccio Damiani, Paola Giannini, Alessandro Ricci, and Mirko Viroli. 2012. Standard Type Soundness for Agents and Artifacts. *Scientific Annals of Computer Science* 22, 2 (2012), 267–326.

[20] Mehdi Dastani, M. Birna van Riemsdijk, Joris Hulstijn, Frank Dignum, and John-Jules Ch. Meyer. 2005. Enacting and Deacting Roles in Agent Programming. In *Agent-Oriented Software Engineering V (Lecture Notes in Computer Science)*, James Odell, Paolo Giorgini, and Jörg P. Müller (Eds.), Vol. 3382. Springer Berlin Heidelberg, 189–204. https://doi.org/10.1007/978-3-540-30578-1_13

[21] Yves Demazeau. 1995. From interactions to collective behaviour in agent-based systems. In *Proc. of the 1st. European Conference on Cognitive Science*. Saint-Malo.

[22] Christophe Feltus. 2014. *Aligning Access Rights to Governance Needs with the Responsability MetaModel (ReMMo) in the Frame of Enterprise Architecture.* Ph.D. Dissertation. University of Namur, Belgium.

[23] Harold Garfinkel. 1967. *Studies in ethnomethodology.* Prentice-Hall Inc., Englewood Cliffs, New Jersey.

[24] Guido Governatori. 2010. Law, logic and business processes. In *Third International Workshop on Requirements Engineering and Law, RELAW 2010, Sydney, NSW, Australia, September 28, 2010.* IEEE, 1–10. https://doi.org/10.1109/RELAW.2010.5625356

[25] Ruth W. Grant and Robert O. Keohane. 2005. Accountability and Abuses of Power in World Politics. *The American Political Science Review* 99, 1 (2005).

[26] Claudia Grigore and Rem Collier. 2011. Supporting Agent Systems in the Programming Language. In *Web Intelligence/IAT Workshops*, Jomi Fred Hübner, Jean-Marc Petit, and Einoshin Suzuki (Eds.). IEEE Computer Society, 9–12.

[27] Claudia Grigore and Rem W. Collier. 2011. AF-Raf: an Agent-Oriented Programming Language with Algebraic Data Types. In *SPLASH Workshops.* 195–200.

[28] Özgür Kafali, Nirav Ajmeri, and Munindar P. Singh. 2016. Revani: Revising and Verifying Normative Specifications for Privacy. *IEEE Intelligent Systems* 31, 5 (2016), 8–15. https://doi.org/10.1109/MIS.2016.89

[29] Özgür Kafali, Munindar P. Singh, and Laurie A. Williams. 2016. NANE: Identifying Misuse Cases Using Temporal Norm Enactments. In *24th IEEE International Requirements Engineering Conference, RE 2016, Beijing, China, September 12-16, 2016.* 136–145. https://doi.org/10.1109/RE.2016.34

[30] Nadin Kökciyan and Pinar Yolum. 2016. PriGuard: A Semantic Approach to Detect Privacy Violations in Online Social Networks. *IEEE Trans. Knowl. Data Eng.* 28, 10 (2016), 2724–2737. https://doi.org/10.1109/TKDE.2016.2583425

[31] Nadin Kökciyan and Pinar Yolum. 2016. PriGuardTool: A Tool for Monitoring Privacy Violations in Online Social Networks (Demonstration). In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, Catholijn M. Jonker, Stacy Marsella, John Thangarajah, and Karl Tuyls (Eds.). ACM, 1496–1497. http://dl.acm.org/citation.cfm?id=2937225

[32] Oscar Nierstrasz and Dennis Tsichritzis (Eds.). 1995. *Object-Oriented Software Composition.* Prentice Hall, Chapter 6, 99–121.

[33] Alessandro Ricci and Andrea Santi. 2012. Typing Multi-agent Programs in simpAL. In *Programming Multi-Agent System (Lecture Notes in Computer Science)*, Mehdi Dastani, Jomi Fred Hübner, and Brian Logan (Eds.), Vol. 7837. Springer, 138–157.

[34] Andrea Santi and Alessandro Ricci. 2012. An Eclipse-based IDE for Agent-Oriented Programming in simpAL. In *Proc. of The Seventh Workshop of the Italian Eclipse Community.*

[35] Munindar P. Singh. 1999. An Ontology for Commitments in Multiagent Systems. *Artificial Intelligence and Law Journal (AILaw)* 7, 1 (1999), 97–113.

[36] Michael Zapf and Kurt Geihs. 2000. What Type Is It? A Type System For Mobile Agents. In *15th European Meeting on Cybernetics and Systems Research (EMCSR).*