Computational Logic in Genova
Logica Computazionale a Genova

Viviana Mascardi, Giorgio Delzanno, Maurizio Martelli
DISI, Università degli Studi di Genova,
Via Dodecaneso 35, 16146, Genova, Italy
E-mail: {viviana.mascardi,giorgio.delzanno,
maurizio.martelli}@unige.it

SOMMARIO/*ABSTRACT*

La Logica Computazionale gioca un ruolo molto rilevante nella ingegnerizzazione di sistemi complessi: può essere usata per specificare sistemi al livello di astrazione più opportuno, la specifica può essere eseguita fornendo gratuitamente un prototipo funzionante e, grazie alla sua semantica ben fondata, può essere usata per verificare formalmente proprietà di programmi e sistemi, cosa fondamentale nello sviluppo di applicazioni critiche dal punto di vista della sicurezza.

Nell'ultimo decennio, il Gruppo di Programmazione Logica del Dipartimento di Informatica e Scienze dell'Informazione (DISI) dell'Università degli Studi di Genova ha applicato la Logica Computazionale per modellare, prototipare e verificare sistemi complessi. Le tre linee di ricerca hanno ampie aree di sovrapposizione: i sistemi complessi che prendiamo in considerazione sono spesso sistemi multiagente per i quali proponiamo linguaggi di modellazione, ambienti di prototipazione e tecniche di verifica. Inoltre usiamo la logica temporale sia per modellare agenti BDI cooperativi, sia per verificare processi a stati infiniti.

In questo articolo descriviamo le attività condotte recentemente in ciascuna direzione di ricerca.

Computational Logic plays a very relevant role in engineering complex systems: it can be used to specify systems at the right level of abstraction, the specifications can be executed, thus providing a working prototype for free, and thanks to its well-founded semantics it can be used to formally verify properties of programs, which is fundamental when safety critical applications are developed.

In the last decade, the Logic Programming Group at the Department of Computer and Information Science (DISI) of Genova University has been applying Computational Logic for modelling, prototyping, and verifying complex systems. These three research lines are largely overlapping: the complex systems we take under consideration are often multiagent systems, for which we propose modelling languages as well as prototyping environments and verification techniques. Also, we use temporal logic both for modelling cooperative BDI agents and for verifying infinite-state processes.

In this paper, we describe the activities that we carried out in the recent years in each research line.

Keywords: Computational Logic, Intelligent Agents, Rapid Prototyping, Verification of Protocols.

Logic Languages for Modelling Rational Agents

Many logics for modelling beliefs, desires and intentions of agents, such as Rao and Georgeff's BDI logic [36, 34, 35] and Wooldridge's $\mathcal{LORA}$ [40], are based on temporal logics like CTL/CTL* (Computational Tree Logic, [24, 17]) where the structure of time is branching in the future and linear in the past. In 2005 we started to explore the advantages of substituting ATL* (Alternating-Time Temporal Logic [1]) to CTL* in Rao and Georgeff's logic. This activity, resulted into the formalization of BDI^{ATL} [33], was born from our effort to find a BDI logic suitable for modelling the behaviour of agents structured according to the CooBDI architecture [2].

A CooBDI agent, whose behavioral specification was given using Prolog, is characterised by a built-in mechanism for retrieving plans from cooperative agents, for example when no local plans suitable for achieving a certain desire are available. In particular, the cooperation strategy of an agent includes the set of agents with which is expected to cooperate (its partner agents, or its "friends"). BDI^{ATL} allows us to express new commitment strategies that are more realistic than those proposed by Rao and Georgeff (and that could not be defined in their logic), since they take collaboration among agents into account. In particular, we can express three variants of Rao and Georgeff's "open minded" commitment: "independent open minded", "optimistic open minded", and "pessimistic

open minded". In these commitment strategies we exploit the new feature that ATL* adds to CTL*, namely *cooperation modalities*, to express the way of thinking of CooBDI agents.

Other logic-based languages conceived for specifying BDI-style and, more in general, rational agents, are CongoLog [27], AGENT-0 [38], Concurrent METATEM [25], $\mathcal{E}_{hhf}$ [20], the IMPACT language [23], and "Dynamics in Logic" [10]. In 2004, we published a survey of these six languages [32], chosen because of the availability, for each of them, of a working interpreter or an automatic mechanism for animating specifications. In our survey we described the logic foundations of each language and we gave an example of use. A comparison along twelve dimensions (purpose of use, language support to time, sensing, concurrency, nondeterminism, etc.) was also provided.

Computational Logic for MAS Prototyping

It is well known that computational logic and logic programming in particular are very suitable to implement sophisticated, self-aware agents able to reason about themselves and the other agents in a multiagent system (MAS). DCaseLP (*Distributed Complex Applications Specification Environment based on Logic Programming* [31]) is an environment for rapid prototyping of MASs developed by the Logic Programming Group at DISI. DCaseLP was initially born as a logic-based framework, as the acronym itself suggests, and then evolved into a multi-language prototyping environment that integrates both imperative (object-oriented) and declarative (rule-based and logic-based) languages, as well as graphical ones. The languages and tools that DCaseLP integrates are UML and an XML-based language for the analysis and design stages, Java, JESS [26] and tuProlog [22] for the implementation stage, and JADE [12] for the execution stage. Software libraries for integrating JESS and tuProlog agents into the JADE platform and for translating UML class diagrams into JESS and tuProlog code are also provided[1]. The methodological integration of DCaseLP with the "Dynamics in Logic" agent programming language is described in [6].

All the applications that we developed with DCaseLP in collaboration with Italian industries, exploit tuProlog for implementing the MAS.

The most recent application, described in [30], is a MAS that monitors processes running in a railway signalling plant, detects functioning anomalies, provides diagnoses for explaining them, and early notifies problems to the Command and Control System Assistance. This work is part of an ongoing project that involves DISI and Ansaldo Segnalamento Ferroviario, the Italian leader in design and construction of signalling and automation systems for railway lines.

[1]The source code of DCaseLP libraries together with manuals and tutorials is available from `http://www.disi.unige.it/person/MascardiV/Software/DCaseLP.html`.

The work described in [37] deals with an electronic implementation of different auction mechanisms. There are many different auction mechanisms that can be classified according to their features [29]. We ran experiments with all the implemented mechanisms under the hypotheses, that, according to the "Revenue Equivalence Theorem" (RET [39]), lead to the existence of an optimal bidder's strategy. The experiments demonstrated that RET is satisfied (up to some error due to discretisation), giving empirical evidence of the correctness of the implementation.

Many applications had also been developed using the ancestor of DCaseLP, CaseLP: a prototype of a multimedia, multichannel, personalised news provider, [19], was developed in collaboration with Ksolutions s.p.a. as part of the ClickWorld project, a research project partially funded by the Italian Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR). Older industrial applications involve freight train traffic [18] and vehicle monitoring [4].

The industrial applications of CaseLP and DCaseLP show an increased industrial interest and trust in both agent-based and declarative technologies, and demonstrate the liveliness of computational logic outside the boundaries of academia.

Verifying Interaction Protocols with Logic

We have recently developed a tool aimed at supporting verification of finite-state interaction protocols in a MAS setting, *West2East* [16], that exploits "*WE*b *S*ervice *T*echnologies to *E*ngineer *A*gent-based *S*of*T*ware" starting from the specification of an Agent Interaction Protocol (AIP). West2East exploits AUML [11] for representing AIPs, many different languages, including standard languages for Web Services, for sharing them, and Computational Logic to reason about them. In particular, West2East consists of a set of libraries for

1. *Translating visual AUML AIPs to various formats*: starting from an AUML interaction diagram graphically drawn using any UML editor, West2East generates the corresponding representation in many formats, including a Prolog term.

2. *Generating code compliant to the AIP*: starting from the Prolog term, a tuProlog program for each agent involved in the AIP is automatically generated by West2East. After a manual completion for adding the information missing in the AIP's specification, such as agents' state and guards of conditions, the tuProlog code can be run inside JADE thanks to the DCaseLP libraries.

3. *Reasoning about the AIP*: a mechanism for allowing tuProlog agents to reason about an AIP by exploiting meta-programming techniques is provided by West2East. Existential and universal properties, such

as "There is one path of the protocol where I will receive $message_1$", and "Whatever the path, I will send $message_2$", can be verified.

In [21] we have further investigated in the relation between (constraint) logic programming and infinite-state verification. More specifically, in [21] we show that a CLP bottom-up evaluation procedure can be applied to automatically verify safety and liveness properties for skeletons of communication protocols (with a fixed number of processes) like mutual-exclusion algorithms. In the case-studies described in [21] the source of infiniteness is the presence of potentially unbounded integer variables in the specification of individual processes. Constraints are used here to symbolically represent infinite collections of system configurations with a fixed number of processes.

Another interesting research line concerns with the application of linear logic programming to verification of infinite-state systems. Linear logic [28] is a suitable logical framework for the specification of concurrent systems. The LO fragment [3] of full linear logic provides multi-headed linear implications with only multiplicative disjunction and additive conjunction in the body. By exploiting and generalizing the connection between verification and logic programming described in [21], in [14] we have defined a bottom-up evaluation strategy for (first order) LO programs based on an effective fixpoint operator à-la T_P (the immediate consequence operator for (constraint) logic programs). The LO T_P operator works on first order multi-headed LO clauses [14]. Furthermore, it can be viewed as a symbolic predecessor operator for transition systems described via multiset rewriting systems defined over first-order atomic formulas. In [15] we have extended the bottom-up evaluation procedure to first order linear logic specification with universally quantified goals. In [13] we have applied the resulting procedure to verify properties of cryptographic protocols for any possible number of principals and parallel sessions.

Conclusions

Research on computational logic in Genova is very lively, and will be even more in the future thanks to the interest on its practical applications raised outside the boundaries of academia. Part of this research has been carried out in joint projects with the Logic Programming and Automated Reasoning Group in Torino. The results of these projects are described in [7, 5], and the active collaboration in witnessed by many other joint activities [8, 9].

The connections between the Logic Programming Groups in Torino and Genova date back to more than 30 years ago. The heads of the groups, Alberto and Maurizio Martelli, besides the same family name, share many common experiences: they worked together at the National Research Council in Pisa, were involved in the committees of conferences and workshops on Computational Logics, and, when moved to Torino and Genova respectively, founded research groups with the same objectives. The profitable collaboration will be pursued in the future with the hope to contribute in making research on Computational Logic an Italian excellence.

REFERENCES

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49:672–713, 2002.

[2] D. Ancona and V. Mascardi. Coo-BDI: Extending the BDI model with cooperativity. In J. A. Leite, A. Omicini, L. Sterling, and P. Torroni, editors, *Proc. of the 1st Declarative Agent Languages and Technologies Int. Workshop, DALT'03, Revised Selected and Invited Papers*, LNAI, pages 109–134. Springer, 2004.

[3] J-M. Andreoli and R. Pareschi. Linear ojects: Logical processes with built-in inheritance. *New Generation Comput.*, 9(3/4):445–474, 1991.

[4] E. Appiani, M. Martelli, and V. Mascardi. A multi-agent approach to vehicle monitoring in motorway. Technical report, Computer Science Department of Genova University, 2000. DISI TR-00-13, Poster session of the Second European Workshop on Advanced Video-Based Surveillance Systems, AVBS 2001.

[5] M. Baldoni, C. Baroglio, G. Berio, A. Martelli, V. Patti, M. L. Sapino, C. Schifanella, M. Alberti, M. Gavanelli, E. Lamma, F. Riguzzi, S. Storari, F. Chesani, A. Ciampolini, P. Mello, M. Montali, P. Torroni, A. Bottrighi, L. Giordano, V. Gliozzi, G. L. Pozzato, D. Theseider Dupré, P. Terenziani, G. Casella, and V. Mascardi. Modeling, verifying and reasoning about web services. *Intelligenza Artificiale*. To appear.

[6] M. Baldoni, C. Baroglio, I. Gungui, A. Martelli, M. Martelli, V. Mascardi, V. Patti, and C. Schifanella. Reasoning about agents' interaction protocols inside DCaseLP. In J. A. Leite, A. Omicini, P. Torroni, and P. Yolum, editors, *Proc. of the 2nd Declarative Agent Languages and Technologies Int. Workshop, DALT'04, Revised Selected and Invited Papers*, volume 3476 of *LNCS*, pages 112–131. Springer, 2004.

[7] M. Baldoni, C. Baroglio, A. Martelli, V. Patti, C. Schifanella, L. Torasso, and V. Mascardi. Personalization, verification and conformance for logic-based communicating agents. In F. Corradini, F. De Paoli, E. Merelli, and A. Omicini, editors, *Proc. of the WOA 2005 National Workshop, Dagli Oggetti Agli Agenti*, pages 177–183. Pitagora Editrice Bologna, 2005.

[8] M. Baldoni, C. Baroglio, and V. Mascardi, editors. *Proceedings of the Multi-Agent Logics, Languages, and Organisations, Federated Workshops, MALLOW'007, Agent, Web Services and Ontologies, Integrated Methodologies (MALLOW-AWESOME'007) workshop, Durham, GB*. 2007.

[9] M. Baldoni, A. Boccalatte, F. De Paoli, M. Martelli, and V. Mascardi, editors. *WOA, Workshop dagli Oggetti agli Agenti, Proceedings*. Seneca Edizioni (Italy), 2007.

[10] M. Baldoni, L. Giordano, A. Martelli, and V. Patti. Modeling agents in a logic action language. In *Proc. of the Workshop on Practical Reasoning Agents, FAPR 2000*, 2000.

[11] B. Bauer, J. P. Müller, and J. Odell. Agent UML: A formalism for specifying multiagent software systems. In P. Ciancarini and M. Wooldridge, editors, *Proc. of the 1st Agent-Oriented Software Engineering Int. Workshop, AOSE'00, Revised Papers*, volume 1957 of *LNCS*, pages 91–104. Springer, 2000.

[12] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley, 2007.

[13] M. Bozzano and G. Delzanno. Automatic verification of secrecy properties for linear logic specifications of cryptographic protocols. *J. Symb. Comput.*, 38(5):1375–1415, 2004.

[14] M. Bozzano, G. Delzanno, and M. Martelli. An effective fixpoint semantics for linear logic programs. *TPLP*, 2(1):85–122, 2002.

[15] M. Bozzano, G. Delzanno, and M. Martelli. Model checking linear logic specifications. *TPLP*, 4(5-6):573–619, 2004.

[16] G. Casella and V. Mascardi. West2East: exploiting WEb Service Technologies to Engineer Agent-based SofTware. *IJAOSE*, 1(3/4):396–434, 2007.

[17] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, pages 52–71, 1981.

[18] A. Cuppari, P. L. Guida, M. Martelli, V. Mascardi, and F. Zini. An agent-based prototype for freight trains traffic management. In P. G. Larsen, editor, *Proc. of the 5th FMERail Workshop. Held in conjunction with FM'99*. Springer, 1999.

[19] M. Delato, A. Martelli, M. Martelli, V. Mascardi, and A. Verri. A multimedia, multichannel and personalized news provider. In G. Ventre and R. Canonico, editors, *Proc. of the 1st Int. Workshop on Multimedia Interactive Protocols and Systems, MIPS 2003*, volume 2899 of *LNCS*, pages 388–399. Springer, 2003.

[20] G. Delzanno and M. Martelli. Proofs as computations in linear logic. *Theoretical Computer Science*, 258(1–2):269–297, 2001.

[21] G. Delzanno and A. Podelski. Constraint-based deductive model checking. *STTT*, 3(3):250–270, 2001.

[22] E. Denti, A. Omicini, and A. Ricci. Multi-paradigm Java-Prolog integration in tuProlog. *Sci. Comput. Program.*, 57(2):217–250, 2005.

[23] T. Eiter, V.S. Subrahmanian, and G. Pick. Heterogeneous active agents, I: Semantics. *Artificial Intelligence*, 108(1-2):179–255, 1999.

[24] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.

[25] M. Fisher and H. Barringer. Concurrent METATEM processes – A language for distributed AI. In *Proceedings of the European Simulation Multiconference*. SCS Press, Copenhagen, Denmark, 1991.

[26] E. Friedman-Hill. *Jess in Action : Java Rule-Based Systems (In Action series)*. Manning Publications, 2002.

[27] G. De Giacomo, Y. Lespérance, and H. J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121:109–169, 2000.

[28] J-Y. Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

[29] P. Klemperer. *Auctions: Theory and practice*. Princeton University Press, 2004.

[30] V. Mascardi, D. Briola, M. Martelli, R. Caccia, and C. Milani. Monitoring and diagnosing railway signalling with rule-based distributed agents. Technical report, Dipartimento di Informatica e Scienze dell'Informazione, University of Genova, Italy, 2008. Technical Report DISI-TR-08-04.

[31] V. Mascardi, M. Martelli, and I. Gungui. DCaseLP: a prototyping environment for multi-language agent systems. In M. Dastani, A. El-Fallah Seghrouchni, J. Leite, and P. Torroni, editors, *Proc. of the 1st Int. Workshop on Languages, Methodologies and Development Tools for Multi-Agent Systems, LADS'007*, LNCS. Springer, 2008. To appear.

[32] V. Mascardi, M. Martelli, and L. Sterling. Logic-based specification languages for intelligent software agents. *TPLP*, 4(4):429–494, 2004.

[33] R. Montagna, G. Delzanno, M. Martelli, and V. Mascardi. BDI^{ATL}: An alternating-time BDI logic for multiagent systems. In M. P. Gleizes, G. A. Kaminka, A. Nowé, S. Ossowski, K. Tuyls, and K. Verbeeck, editors, *Proc. of the 3rd European Workshop on Multi-Agent Systems, EUMAS'05*, pages 214–223. Koninklijke Vlaamse Academie van Belie voor Wetenschappen en Kunsten, 2005.

[34] A. S. Rao and M. P. Georgeff. Asymmetry thesis and side-effect problems in linear-time and branching-time intention logics. In J. Myopoulos and R. Reiter, editors, *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence, IJCAI-91*. Morgan Kaufmann publishers, 1991.

[35] A. S. Rao and M. P. Georgeff. Deliberation and intentions. In *Proc. of 7th Conference on Uncertainity in Artificial Intelligence*. Morgan Kaufmann publishers, 1991.

[36] A. S. Rao and M. P. Georgeff. Modelling rational agents within a BDI-architecture. In *Proc. of the 2nd Int. Conference of Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann publishers, 1991.

[37] D. Roggero, F. Patrone, and V. Mascardi. Designing and implementing electronic auctions in a multiagent system environment. In F. Corradini, F. De Paoli, E. Merelli, and A. Omicini, editors, *Proc. of the WOA 2005 National Workshop, Dagli Oggetti Agli Agenti*, pages 157–163. Pitagora Editrice Bologna, 2005.

[38] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.

[39] W. Vickrey. Auction and bidding games. In *Recent advances in Game Theory*, pages 15–27. Princeton University Conference, 1962.

[40] M. Wooldridge. *Reasoning about rational agents*. Mit press, 2000.