Undefined 0 (2010) 1
IOS Press

Some Applications of Computational Logic to the Development of Intelligent Systems and Verification Methods

Viviana Mascardi, Giorgio Delzanno, Maurizio Martelli
Department of Computer and Information Science (DISI), Università degli Studi di Genova, via Dodecaneso 35, 16146, Genova, Italy

Abstract. Computational Logic plays a very relevant role in engineering complex systems. It can be used to specify systems at different levels of abstraction. The specifications are executable, thus providing a working prototype for free. Thanks to its well-founded semantics it can be used to reason about the correctness of the specifications, a fundamental aspect when safety critical applications are developed. Researchers working in the Logic Programming Group at DISI, a Genova University Department, have applied methods and tools of Computational Logic for modelling, prototyping, and verifying complex systems. These three research lines are largely overlapping: the complex systems we take into account are often multiagent systems, for which we propose modelling languages as well as prototyping environments and verification techniques. In this paper we describe activities that, in the last decade, we carried out along these research lines.

Keywords: Computational Logic, Intelligent Agents, Rapid Prototyping, Protocol Verification

Logic Languages for Modelling Rational Agents

Many logics for modelling beliefs, desires and intentions of agents, such as Rao and Georgeff's BDI logic [47,46] and Wooldridge's $\mathcal{LORA}$ [51], are based on temporal logics like CTL/CTL* (Computational Tree Logic [34,26]) 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 [2]) for CTL* in Rao and Georgeff's logic. This activity, that resulted in the formalisation of BDI^{ATL} [45], was born from our effort to find a BDI logic suitable for modelling the behaviour of agents structured according to the CooBDI architecture [3].

A CooBDI agent, whose behavioural 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 it 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 "Dynamics in Logic" [12], $\mathcal{E}_{hhf}$ [30], the IMPACT language [33], Concurrent METATEM [35], CongoLog [37], and AGENT-0 [49]. In 2004, we published a survey of these six languages [43], 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 logical 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 other agents in a multi-agent system (MAS). DCaseLP (*Distributed Complex Applications Specification Environment based on Logic Programming* [42]) 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 [36] and tuProlog [32] for the implementation stage; JADE [14] for the execution stage. DCaseLP provides libraries for integrating JESS and tuProlog agents into the JADE platform and libraries for translating UML class diagrams into JESS and tuProlog code. 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 methodological integration of DCaseLP with the "Dynamics in Logic" agent programming language is described in [8].

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

One recent activity, described in [40,23], 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. See also [22].

The research activity carried out with Ansaldo led us to modelling and simulating the behaviour of complex systems where a limited set of agents must cooperate (or compete) to share a limited set of resources under topological and spatial constraints. Problems of this kind are known as "Multiagent Resource Allocation (MARA) problems" [25]. Although in [21] we concentrated on a specific MARA problem, the methodology we followed for solving it and the tools we used (or that we mean to use in the future) could be exploited as well for different problems and scenarios involving autonomous entities that are difficult to model with standard simulation techniques.

The work described in [48] deals with an electronic implementation of different auction mechanisms. There are many different auction mechanisms that can be classified according to their features [39]. The experiments ran with all the implemented mechanisms satisfied the "Revenue Equivalence Theorem" [50] (up to some error due to discretisation), giving empirical evidence of the correctness of our implementation.

Many applications had also been developed using the ancestor of DCaseLP, CaseLP: a prototype of a multimedia, multichannel, personalised news provider, [28], 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 [27] and vehicle monitoring [6].

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.

The expertise obtained in these years allowed us to apply MAS techniques also to very different fields such as web applications for the cultural heritage in connection with the new field of research of the Virtual Institutions [15,4].

We were also interested in introducing the use of ontologies [20] and of sophisticated ontology matching techniques [41] into our framework.

Verification of Interaction Protocols

To this end we developed a tool aimed at supporting verification of finite-state interaction protocols in a MAS setting, *West2East* [24], 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 [13] 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, 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.

Verification of Communication and Security Protocols

In addition to the work on intelligent systems, in the last ten years researchers of our group have studied computational logic for the specification and analysis of concurrent systems. One of these formalisms is linear logic [38] and logic programming languages inspired to it like LO [5]. Linear logic is based on the metaphor "formulas and resources" that finds a natural application when modelling state-based and concurrent computation [44]. The LO fragment [5] of full linear logic extends Horn formulas (the class of formulas underlying languages like Prolog) by allowing clauses with multiple heads. Multi-headed clauses have a natural interpretation as multiset rewriting rules, which in turn can be applied to locally specify the possible interaction of concurrent activities.

As discussed in [31], a natural way to define a verification method for temporal properties consists in the definition of a bottom-up evaluation procedure for the considered type of logic programs. The bottom-up evaluation procedure computes immediate consequences of formulas that describe violations of a given property. Then, by saturating the set of immediate consequences, we compute formulas that represent preconditions for exposing a possible error.

Following this idea, in [17] we have defined a bottom-up evaluation strategy for LO programs obtained by an adequate extension of the immediate consequence operator used in the fixpoint semantics of logic programs. In the propositional case the termination of the resulting procedure is guaranteed whenever intermediate formulas computed during the bottom-up evaluation steps are compared using a well-quasi ordering like multiset inclusion (over a finite alphabet).

In [18] we have extended the bottom-up evaluation procedure in order to deal with first-order multi-headed LO clauses. This extension is achieved by replacing term instantiation with term unification in the definition of the immediate consequence operator. A single step of the bottom-up evaluation procedure can be computed in an effective way. The termination of the procedure can be guaranteed only in very special cases (monadic atomic formulas without function symbols).

In [19] we have further extended the bottom-up evaluation procedure in order to deal with LO clauses with universally quantified goals. The extension is based on the observation that, when reasoning bottom-up, universal quantification can be handled locally to the application of the fixpoint operator by ensuring that eigen-variables are not extruded from their scope during the unification with already deduced formulas.

In [16] we have applied the resulting procedure to verify secrecy properties of examples of cryptographic protocols with an unbounded number of agents and parallel sessions. LO programs are used to specify the behaviour of honest agents and of the intruder. Communication is assumed to be asynchronous and is modelled using rewriting of atoms that represent messages. A top-down execution of the resulting LO program represents a finite set of parallel protocol sessions. Universal quantification in goal formulas is used to specify the dynamic generation of fresh names (nonces). Violation of secrecy properties (e.g. secret data are exposed to the intruder) are specified using LO axioms. The bottom-up evaluation of the resulting LO program generates the collection of preconditions under which the intruder is able to capture the secret data exchanged by honest agents.

In [29] we have presented a generalisation of the verification methods developed in [17,19,16] for specifications given in multiset rewriting with equality and ordering constraints over an infinite domain (e.g. to model pure names or indexes). This specification language can be used to specify concurrent systems with relations over data of individual processes (or of group of processes). Decidability results for verification problems in fragments of the resulting specification language are discussed in [1]. In the same paper we compare the proposed specification language with other models of concurrent computation like (Colored) Petri nets.

In this research line we are currently addressing the problem of finding more general constraint languages to symbolically represent and manipulate dif-

ferent types of communication topologies (e.g. trees and graphs) useful to reason about the behaviour of a larger class of concurrent systems.

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. Some results of this effort are described in [9], and the active collaboration is witnessed by many other joint activities [10,11].

Moreover, recently we decided to propose projects and joint activities for the development of an Interaction-oriented Framework for Designing, Verifying and Programming Multi-Agent Systems, called MERCURIO [7], in collaboration also with the groups of the Universities of Bologna and Parma.

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] P. Abdulla, G. Delzanno, and L. Van Begin. A classification of the expressive power of well-structured transition systems. Technical report, 2010. To appear in Information and Computation.

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

[3] 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.

[4] M. Ancona, V. Mascardi, G. Quercini, A. Bogdanovych, H. de Lumley, L. Papaleo, S. Simoff, and A. Traverso. Virtual institutions for preserving and simulating the culture of Mount Bego's ancient people. In *Proc. of the 11th VAST International Symposium on Virtual Reality, Archaeology and Cultural Heritage*, 2010.

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

[6] E. Appiani, M. Martelli, and V. Mascardi. A multi-agent approach to vehicle monitoring in motorway. Technical report, CS Department of Genova University, 2000. DISI TR-00-13, Poster session of AVBS 2001.

[7] M. Baldoni, C. Baroglio, F. Bergenti, A. Boccalatte, E. Marengo, M. Martelli, V. Mascardi, L. Padovani, V. Patti, A. Ricci, G. Rossi, and A. Santi. MERCURIO: An interaction-oriented framework for designing, verifying and programming multi-agent systems. In N. Fornara and G. Vouros, editors, *Notes of the Eleventh International Workshop on Coordination, Organization, Institutions and Norms in Agent Systems*, pages 134–149, 2010.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

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

[15] A. Bogdanovych, L. Papaleo, M. Ancona, V. Mascardi, G. Quercini, S. Simoff, A. Cohen, and A. Traverso. Integrating agents and virtual institutions for sharing cultural heritage on the web. In *Notes of the Workshop On Intelligent Cultural Heritage (satellite workshop of AI*IA 2009)*, 2009.

[16] 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.

[17] M. Bozzano, G. Delzanno, and M. Martelli. A bottom-up semantics for linear logic programs. In *2nd International Conference on Principles and Practice of Declarative Programming, PPDP 2000*, pages 92–102, 2000.

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

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

[20] D. Briola, A. Locoro, and V. Mascardi. Ontology Agents in FIPA-compliant Platforms: a Survey and a New Proposal. In M. Baldoni, M. Cossentino, F. De Paoli, and V. Seidita, editors, *Proc. of WOA 2008: Dagli oggetti agli agenti*. Seneca Edizioni, 2008.

[21] D. Briola, M. Martelli, and V. Mascardi. Specification, simulation and verification of negotiation protocols in a unified agent-based framework (extended abstract). In *Notes of the 12th Italian Conference on Theoretical Computer Science, ICTCS*, 2010.

[22] D. Briola, V. Mascardi, and M. Martelli. Intelligent agents that monitor, diagnose and solve problems: Two success stories of industry-university collaboration. *Journal of Information Assurance and Security*, 4(2):106–116, 2009.

[23] D. Briola, V. Mascardi, M. Martelli, G. Arecco, R. Caccia, and C. Milani. A prolog-based MAS for railway signalling monitoring: Implementation and experiments. In *Workshop Dagli Oggetti agli Agenti, WOA'08, Proceedings*. Seneca Edizioni, 2008.

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

[25] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. A. Padget, S. Phelps, J. A. Rodríguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica (Slovenia)*, 30(1):3–31, 2006.

[26] 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.

[27] 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.

[28] 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.

[29] G. Delzanno. Constraint-based automatic verification of abstract models of multithreaded programs. *TPLP*, 7(1-2):67–91, 2007.

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

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

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

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

[34] 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.

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

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

[37] 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.

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

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

[40] V. Mascardi, D. Briola, M. Martelli, R. Caccia, and C. Milani. Monitoring and diagnosing railway signalling with logic-based distributed agents. In *Proc. of the International Workshop on Computational Intelligence in Security for Information Systems, CISIS'08*, pages 108–115. Springer, 2008.

[41] V. Mascardi, A. Locoro, and P. Rosso. Automatic ontology matching via upper ontologies: A systematic evaluation. *IEEE Trans. Knowl. Data Eng.*, 22(5):609–623, 2010.

[42] 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*, volume 5118 of *LNCS*, pages 139–155. Springer, 2008.

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

[44] Dale Miller. Specifications using multiple-conclusion logic programs. In *ALP*, pages 3–4, 1994.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

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

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

[51] M. Wooldridge. *Reasoning about rational agents*. MIT press, 2000.