Modello e Verifica di Processi di Business e Coreografie in ALP

Modeling and Verification of Business Processes and Choreographies in ALP

Federico Chesani Paola Mello Marco Montali Paolo Torroni

SOMMARIO/*ABSTRACT*

Questo articolo introduce brevemente le nostre recenti attività di ricerca in relazione all'uso della programmazione logica per la specifica e verifica delle interazioni in vari contesti. L'articolo evidenzia alcuni risultati riportati in vari ambiti: sistemi multi-agente, servizi web e argomentazione, con particolare enfasi sugli aspetti collegati ai processi di business e alle coreografie di servizi Web.

In this article we overview our recent research activity concerning the use of logic programming for interaction specification and verification in several domains. We outline relevant results in the areas of multi-agent systems, argumentation and web services, and we devote a special emphasis to issues related to business processes and Web service choreographies.

Keywords: Logic programming, hypothetical reasoning, interaction, modelling, verification, multi-agent systems, protocols, business processes, web services, choreographies, semantic web, argumentation.

1 Background

Over two decades ago, a significant part of the Italian and European CS research community and IT industry expressed great interest in the new Logic Programming (LP) paradigm. Such an interest was sensibly encouraged by the pioneering work of Giorgio Levi, Franco Turini, Ugo Montanari, Alberto Martelli, and others. The Artificial Intelligence group of DEIS, School of Engineering, University of Bologna, was born at the time of the LP wave of the Eighties. Our group was attracted by the unique features of LP, including its ability to marry formal and practical aspects, and to enable the correspondence of a declarative language with an underlying execution model.

The main research directions of our group back then were centered around distribution, modularity, parallelism, and language extensions such as Constraint and Abductive Logic Programming. In order to enable programming in the large in the LP paradigm, two approaches have been studied for structuring logic programs: an algebraic method based on meta-operators, and another approach based on language extensions. The first model brought to the definition of an extended LP language called StructuredProlog [16], while the second approach was based on the introduction of negation in LP, to support non-monotonic reasoning.

StructuredProlog allows to to integrate blocks, modules, hypothetical reasoning, logical theory and object taxonomies. It has been implemented as an extension of the Warren Abstract Machine, via software emulation and then in hardware, and optimized using partial evaluation techniques. Past research also focussed on parallel logic languages with AND parallelism and no variable sharing on a MIMD architecture. Inter-process communication and synchronization was possible via multi-headed clauses and a shared blackboard, and an optimized unification mechanism specifically tailored to serve the purpose. Finally, the LP paradigm has been integrated with the OO paradigm, to define the Distributed Logic Objects language (DLO). In DLO, methods are expressed via multi-headed clauses, in a purely declarative style, while specific constructs are defined to express interaction among objects and inheritance.

2 The SOCS Project

Since 2001, the group has devoted most of its resources to the study of computational logic-based multi-agent systems [19], specifically agent interaction: the aim was to develop an LP-based language and an operational model for the specification and verification of agent interaction protocols. Such work has been carried out in the context of the EU-funded SOCS project [1] . The SOCS society model

[1] Project IST-2001-32530, 5FP. SOCS stands for "Societies Of ComputeeS: a computational logic model for the description, analysis and ver-

[21, 3], developed by a joint effort between the University of Bologna and the University of Ferrara, gives concrete guidelines for the formal specification of the interaction among agents that form a society, and for the definition of a computational logic-based architecture for agent interaction. In the proposed architecture, the society defines the allowed interaction protocols, which in turn are defined by means of *Social Integrity Constraints* (ICs). The society knowledge is defined as an abductive logic program [9]: ICs are used in order to express constraints on the communication patterns, while expected communicative acts ("*expectations*") are expressed as abducible predicates. Both the specification language and the underlying proof-procedure are called $\mathcal{S}$CIFF.

Expectations, whose intuition recalls the usual deontic operators of permission, obligation, and prohibition [8], are used to provide a semantics to both agent communication languages and to interaction protocols [6]. The resulting model is based on a declarative (logic) representation, therefore easy to understand.Moreover, its operational model can be exploited to achieve an implementation of societies of computees based on their formal specifications [2]. Thanks to the link between formal specification and implementation, the model provides also a good ground for the automatic verification and formal proof of properties [10].

The society model and the $\mathcal{S}$CIFF operational model were satisfactorily tested on a number of applications. These include resource exchange [11], e-commerce protocols [7], and combinatorial auctions [1]. A repository of protocols specified using $\mathcal{S}$CIFF is publicly available through the project's home page [22].

The SOCS-SI tool [4]) supports $\mathcal{S}$CIFF models and have been used for extensive experimentation. It takes as input the declarative formalisation, and it allows the automated verification of the social aspects of a SOCS application. SOCS-SI is general in its scope, and has been interfaced to other implemented agent platforms, such as JADE, and to other non-agent related communication platforms, like e.g. TuCSoN. SOCS-SI uses the $\mathcal{S}$CIFF proof procedure, that has been implemented using SICStus Prolog, and in particular its CHR library. The interested reader can learn more about $\mathcal{S}$CIFF in [5], and in the tutorial paper [17]. SOCS-SI and $\mathcal{S}$CIFF are publicly available on the web [2] .

3 Current Research Directions

Most of our current research has originated from the outcomes of SOCS. Starting form the many analogies between the agent paradigm and the Web service model, interaction protocols and choreographies has been the subject of conspicuous research carried out in the context of two recent national projects lead by Alberto Martelli [3] . Part of the research activity done within these projects has built on $\mathcal{S}$CIFF to i) produce new formalisms for the specification and verification of interaction protocols and choreographies; and to ii) develop new techniques for automatic property verification and reasoning about Web Services.

The translation of graphical modeling languages into the formal languages developed in these projects has been also subject of research. Our group has studied the translation of choreographies (represented in WS-CDL or in BPMN) into its corresponding $\mathcal{S}$CIFF specification, focussing on verification of compliance. Several tools, based on the $\mathcal{S}$CIFF procedure, have been developed to cope with complete logs and with run-time events. Further supported types of verification regard the proof of "high level" properties, such as verifying in an e-commerce scenario, that a buyer is guaranteed to receive the good he/she paid for, and the seller is guaranteed to be paid.

Alongside our research on Web Services, we have extended and applied $\mathcal{S}$CIFF in the context of agent-oriented requirements engineering. This has brought to the development of $\mathcal{B}$-Tropos ($\mathcal{B}$ standing for *Business*): a unified framework for information systems engineering, with the aim to reconcile requirements elicitation with declarative specification, prototyping, and analysis [15]. $\mathcal{B}$-Tropos, built on the well-known Tropos methodology [14], lets the user to express temporal and data constraints between tasks, hence introducing also the concepts of start and completion times, triggering events, and deadlines. The verification capabilities supported by the $\mathcal{S}$CIFF proof allow prototyping (animation) and analysis (properties and conformance verification) directly in $\mathcal{B}$-Tropos. Early requirements engineers will be able to test their models directly; engineers testing model properties will not have to resort to ad-hoc, error-prone translations of high-level models into other languages, thanks to the automatic translation of $\mathcal{B}$-Tropos models into $\mathcal{S}$CIFF programs; finally, managers monitoring the correct behavior of a system will exploit the $\mathcal{S}$CIFF specification to check the compliance using the SOCS-SI runtime and off-line checking facilities [4].

Another current research direction which builds on SCIFF concerns argumentation in the Semantic Web [24]. Our work resulted in the development of an operational argumentation framework, called ArgSCIFF, to support dialogic argument exchange between Semantic Web Services. In ArgSCIFF, an intelligent agent can interact with a Web Service and reason from the interaction result. The reasoning semantics is an argumentation semantics that views the interaction as a dialogue. The dialogue lets two parties exchange arguments and attack, challenge, and justify them

ification of global and open societies of heterogeneous computees." The project run for 42 months, from January 2001 until June 2005, and it involved 6 academic institutions, including the University of Bologna and the University of Ferrara. See [12, 23].

[2] `http://lia.deis.unibo.it/research/socs_si` and `http://lia.deis.unibo.it/research/sciff`

[3] In 2004-2005, our group has been involved as a partner in the National MIUR (ex 40%) project on "Development and verification of logic-based multi-agent systems," and in 2006-2007 on the National PRIN (ex 40%) project on "Specification and verification of agent interaction protocols." For more information, see the project Web site [20]. A report on the most recent project is due to appear on this magazine [13].

on the basis of their knowledge. This format has the potential to overcome a well-known barrier to human users adoption of IT solutions because it permits interaction that includes justified answers that can be reasoned about and rebutted.

4 CLIMB

Actually, a great deal of our resources are devoted to the development of LP-based techniques for modeling and verifying business processes and choreographies. The reference framework for this work is called CLIMB [4] . As specification language, CLIMB adopts an extension of DecSerFlow/Condec, a family of graphical languages for the declarative specification of service/business flows [26]. Graphical models are then automatically mapped onto $\mathcal{S}$CIFF, integrating the best of the two approaches:

- CLIMB models are declarative and open. They do not specify one particular flow of execution, but rather focus on the set of constraints that must be satisfied by interacting entities. Constraints specify either what is mandatory or forbidden during execution.
- Different verification tasks can be applied on CLIMB models by exploiting the proof-theoretic operational counterpart of $\mathcal{S}$CIFF as well as different logic programming techniques.

In particular, CLIMB exploits $\mathcal{S}$CIFF for carrying out both run-time and a-priori verification tasks.

At run-time, $\mathcal{S}$CIFF can be used as an alerting infrastructure capable to perform *compliance checking*, i.e., verifying whether a concrete process execution (or service interaction) complies with the prescribed model (and detecting violations as soon as possible). Such a verification can be seamlessly applied a-posteriori as well, checking already completed execution traces. In this respect, CLIMB rules are used as an intuitive classification criterion which split analyzed traces into a compliant and non compliant sub-sets; a plug-in which exploits such a reasoning technique has been implemented and integrated inside the ProM[25] process mining framework.

At static time, the "generative" variant of the $\mathcal{S}$CIFF proof procedure can be exploited to check the consistency of developed models, by detecting the presence of conflicts (which undermine the possibility of executing the model) and by discovering if they contain dead activities (i.e., activities that can be never executed). Such verifications constitute the basis also for determining if different CLIMB models can be composed without introducing conflicts. This is particularly important in a service-oriented setting, where a choreography can be intended as a contract aiming to make different partners correctly collaborate, and then a set of compatible concrete services implementation must be found to concretely implement the system.

It is worth noting that DecSerFlow/Condec models have an alternative underlying semantics in terms of Linear Temporal Logic formulas, which enable the possibility to apply model checking techniques in order to verify the designed models. In this respect, a research activity focused on more foundational aspects is being carried out, to compare expressivity, complexity and reasoning capabilities of the two frameworks.

[4]CLIMB stands for "Computational Logic for the verIfication and Modeling of Business processes and choreographies." For more information and up-to-date resources the interested reader can refer to the CLIMB Web site [18].

Acknowledgements

Much of the work presented here has been done in tight cooperation with the AI group of the University of Ferrara. This paper is complementary to [12], where they focus on the learning and property verification issues in relation with the work carried out within and following SOCS.

REFERENCES

[1] M. Alberti, F. Chesani, M. Gavanelli, A. Guerri, E. Lamma, P. Mello, and P. Torroni. Expressing interaction in combinatorial auction through social integrity constraints. *Intelligenza Artificiale*, II(1):22–29, 2005.

[2] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. A logic based approach to interaction design in open multi-agent systems. In *Proc. 13th IEEE international Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WET-ICE 2004)*, pages 387–392. IEEE Press, 2004.

[3] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. The SOCS computational logic approach for the specification and verification of agent societies. In *Global Computing*, volume 3267 of *LNAI*, pages 324–339. Springer-Verlag, 2005.

[4] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Compliance verification of agent interaction: a logic-based tool. *Applied Artificial Intelligence*, 20(2-4):133–157, Feb.-Apr. 2006.

[5] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Verifiable agent interaction in abductive logic programming: the SCIFF framework. *ACM Transactions on Computational Logic*, 9(4), 2008.

[6] M. Alberti, A. Ciampolini, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. A social ACL semantics by deontic constraints. In *Multi-Agent Systems and*

Applications III, volume 2691 of *LNAI*, pages 204–213. Springer-Verlag, 2003.

[7] M. Alberti, D. Daolio, P. Torroni, M. Gavanelli, E. Lamma, and P. Mello. Specification and verification of agent interaction protocols in a logic-based system. In *Proceedings of the 19th Annual ACM Symposium on Applied Computing (SAC 2004)*, pages 72–78. ACM Press, 2004.

[8] M. Alberti, M. Gavanelli, E. Lamma, P. Mello, G. Sartor, and P. Torroni. Mapping deontic operators to abductive expectations. *Computational and Mathematical Organization Theory*, 12(2-3):205–225, Oct. 2006.

[9] M. Alberti, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. An Abductive Interpretation for Open Societies. In *Advances in Artificial Intelligence*, volume 2829 of *LNAI*, pages 287–299. Springer-Verlag, 2003.

[10] M. Alberti, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Specification and verification of agent interactions using social integrity constraints. *ENTCS*, 85(2), 2003.

[11] M. Alberti, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Modeling interactions using *Social Integrity Constraints*: A resource sharing case study. In *Declarative Agent Languages and Technologies*, volume 2990 of *LNAI*, pages 243–262. Springer-Verlag, May 2004.

[12] M. Alberti, M. Gavanelli, E. Lamma, F. Riguzzi, and S. Storari. Inducing specification of interacting systems and proving their properties: An approach grounded on computational logic. *Intelligenza Artificiale*, In this issue.

[13] M. Baldoni *et al.* Modeling, verifying and reasoning about web services. *Intelligenza Artificiale*, In press.

[14] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8:203–236, 2004.

[15] V. Bryl, P. Mello, M. Montali, P. Torroni, and N. Zannone. B-tropos: Agent-oriented requirements engineering meets computational logic for declarative business process modeling and verification. In *Computational Logic in Multi-Agent Systems VIII*, LNAI. Springer-Verlag, 2008.

[16] M. Bugliesi, E. Lamma, and P. Mello. Modularity in logic programming. *Journal of Logic Programming*, 19/20:43–502, May/Jun 1994. Special Issue on "10 years of Logic Programming.".

[17] F. Chesani, M. Gavanelli, M. Alberti, E. Lamma, P. Mello, and P. Torroni. Specification and verification of agent interaction using abductive reasoning (tutorial paper). In *Computational Logic in Multi-Agent Systems VI*, volume 3900 of *LNAI*, pages 243–264. Springer-Verlag, 2006.

[18] CLIMB: Computational logic for the verification and modeling of business processes and choreographies, 2008. `http://lia.deis.unibo.it/research/climb`.

[19] M. Fisher, R. H. Bordini, B. Hirsch, and P. Torroni. Computational logics and agents: a road map of current technologies and future trends. *Computational Intelligence*, 23(1):61–91, 2007.

[20] MASSiVE: sviluppo e verifica di sistemi multiagente basati sulla logica. `http://www.di.unito.it/massive`.

[21] P. Mello, P. Torroni, M. Gavanelli, M. Alberti, A. Ciampolini, M. Milano, A. Roli, E. Lamma, F. Riguzzi, and N. Maudet. A logic-based approach to model interaction amongst computees. Technical report, SOCS Consortium, 2003. Deliverable D5. SOCS project web site [22].

[22] Societies Of ComputeeS (SOCS): a computational logic model for the description, analysis and verification of global and open societies of heterogeneous computees. IST-2001-32530, 2002-2005. `http://lia.deis.unibo.it/research/socs`.

[23] F. Toni. Multi-agent systems in computational logic: Challenges and outcomes of the SOCS project. In *Computational Logic in Multi-Agent Systems VI*, volume 3900 of *LNAI*, pages 420–426. Springer-Verlag, 2006.

[24] P. Torroni, M. Gavanelli, and F. Chesani. Argumentation in the semantic web. *IEEE Intelligent Systems*, 22(6):66–74, Nov/Dec 2007.

[25] W. van der Aalst, B. van Dongen, C. Günther, R. Mans, A. A. de Medeiros, A. Rozinat, V. Rubin, M. Song, H. Verbeek, and A. Weijters. ProM 4.0: Comprehensive Support for Real Process Analysis. In J. Kleijn and A. Yakovlev, editors, *Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007)*, volume 4546 of *LNCS*, pages 484–494. Springer-Verlag, 2007.

[26] W. M. P. van der Aalst and M. Pesic. Decserflow: Towards a truly declarative service flow language. In M. Bravetti, M. Núñez, and G. Zavattaro, editors, *Web Services and Formal Methods, Third International Workshop, WS-FM 2006 Vienna, Austria, September 8-9, 2006, Proceedings*, volume 4184 of *LNCS*, pages 1–23. Springer, 2006.