Static Knowledge Representation of Multi-Agent System's Specification by Description Logics

Haiyan Che[1], Jigui Sun[1,2] and Haibo Yu[1]
[1]College of Computer Science and Technology, Jilin University
Changchun 130012, China
{chehy,jgsun,yuhb}@jlu.edu.cn

[2]Key Laboratory of Symbolic Computation and
Knowledge Engineering of Ministry of Education, Jilin University
Changchun 130012, China

Abstract

Modularity and rigor are two key elements for multi-agent technology. Hong Zhu's multi-agent system (MAS) development method provides proper language facilities supporting modularity. To enhance this method with rigor advocates a DL method to map the specification of MAS into a DL TBox. Thus, we can use the existing DL reasoners and systems to verify and validate some system's properties.

1 Introduction

Agent technology has been predicted to be the next generation mainstream computation paradigm. However, lack of rigor and language facilities directly supporting modularity and abstract mechanisms hamper the wide-scale adoption of it. Hong Zhu et al. contribute on the research of such language facilities. They advocate a model driven method for MAS development, which designs and implements CAMLE (Caste-centric Agent-oriented Modeling Language and Environment) [1] providing tools for constructing graphic MAS models, automatically checking consistency between various views and models, and automatically transforming these models into formal specifications in SLABS [2]. To enhance Hong Zhu's method with rigor, we propose a DL method to map the specification of MAS written in SLABS into a DL TBox. Taking advantage of DL's decidability and its existing reasoners and systems, we can perform certain reasonings about the system's properties.

2 Static Knowledge Representation with Description Logic $\mathcal{ALCNIF}_{reg}$

The most important language facilities in SLABS are caste, agent and scenario. A MAS is regarded as a set of agents, and castes are the classifiers of agents, or the roles agents play. They define templates of the structure and behavior characteristics of agents. Scenarios are defined as the behavior patterns of the agents in an agent's environment, perceiving which the agent can decide its action rather than driven by message communications. To represent the static part of the MAS's specification we choose $\mathcal{ALCNIF}_{reg}$, which extends $\mathcal{ALC}$ with unqualified number restrictions ($\mathcal{N}$), inverse roles ($\mathcal{I}$), agreement ($\mathcal{F}$), and the composition of roles (reg).Based on the translation method from object-oriented model to DLs proposed by Calvanese et al. [3] we define a map from SLABS to DLs, which maps each caste and agent definition to corresponding concept, and depicts the instance relationship between agent and caste by the concept constructor of role agreement.

3 Conclusion

This paper advocates a DL method to formalize the specification and perform certain reasonings about the properties of MAS. $\mathcal{ALCNIF}_{reg}$ is chose to represent the static part of the specification and the satisfiability of concepts w.r.t. acyclic TBoxes is decidable with the complexity of NExpTime-complete [4]. To make our method practicable we need extend basic DL to represent dynamic knowledge as our further work.

References

[1] Zhu, H. and Shan, L., Caste-Centric Modelling of Multi-Agent Systems: The CAMLE Modelling Language and Automated Tools, in Beydeda, S. and Gruhn, V. (eds) Model-driven Software Development, Research and Practice in Software Engineering, Vol. II, Springer (2005) 57-89

[2] Zhu, H., "SLABS: A Formal Specification Language for Agent-Based Systems", Int. J. of Software Engineering and Knowledge Engineering 11(5) (2001) 529-558

[3] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Unifying class-based representation formalisms. J. of Artificial Intelligence Research, 11 (1999) 199-240

[4] F. Baader, D. McGuinness, D. Nardi, and P. Patel-Schneider. The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press (2003)