High-Level Program Execution in Multi-Agent Settings *

Liangda Fang
Department of Computer Science
Sun Yat-Sen University
Guangzhou 510006, China
fangld@mail2.sysu.edu.cn

Abstract

In this paper, we state the challenges of high-level program execution in multi-agent settings. We first introduce high-level program execution and the related work. Then we describe the completed work, the future work and its approaches. We conclude with the expected contributions of our research.

1 Introduction

One of the central problems in Artificial Intelligence is to generate a plan (*i.e.*, a sequence of actions) to fulfill the agent's goal. High-level program execution [Levesque *et al.*, 1997] is an approach to this problem, and it is based on the situation calculus (SitCalc) [Reiter, 2001]. It requires the user to encode his comprehension about the domain into a control program that contains nondeterministic choices, and the interpreter transforms this high-level program to a feasible plan. If the program is almost deterministic, a plan can be produced very efficiently. Otherwise, the complexity of high-level program execution is more like that of planning, which is another solution to this problem. By and large, high-level program execution is more efficient than planning.

2 Background Work

Currently, the area of high-level program execution mainly concerns single-agent settings [Giacomo and Levesque, 1999; Giacomo *et al.*, 2000; Fan *et al.*, 2012]. However, in most scenarios many agents interact with each other. Each agent decides the next action according to her mental attitudes not only about the world but also about other agents' mental attitudes. Several scholars have started to study high-level program execution in multi-agent settings [Shapiro *et al.*, 1998; Kelly, 2008].

Traditionally, high-level program execution consists of three parts including *an extension to the SitCalc*, *static and dynamic reasoning* of the extension to the SitCalc and *the semantics of the control program*. Regression and progression are two powerful solutions to dynamic reasoning. Roughly, regression reduces a query about the future to a query about the initial knowledge base (KB). Progression, on the other hand, updates the initial KB according to the effects of each action and then checks whether the formula holds in the resulting KB. We will analyze the related work and our proposed work based on these three parts in the following.

Shapiro *et al.* represented the knowledge of agents by incorporating the standard possible-world model to the SitCalc. Due to the lack of reasoning about the extension to the SitCalc, the interpreter could not be implemented. Kelly extended the SitCalc so that it can describe first-order dynamic logic formulas. In Kelly's framework, a formula in the extension to the SitCalc can be translated into an equivalent formula in propositional dynamic logic (PDL) with the restriction of finite domains. So Kelly utilized a PDL prover to implement static reasoning. To solve dynamic reasoning, he resorted to regression technique. Finally, Kelly integrated true concurrency, temporal component and natural actions into high-level program execution, and used transition semantics to define the given constructs. But the propositionalisaztion of a formula in the extension to the SitCalc causes exponential blowup in size, and the satisfiability and validity problems for PDL are EXPTIME-complete, so its interpreter cannot be used even in some simple cases.

3 Problem Statements

As mentioned above, high-level program execution in multi-agent settings is necessary to generate a plan in actual environments. However, existing solutions, *e.g.* [Shapiro *et al.*, 1998] and [Kelly, 2008], are insufficient in multi-agent environments. They have three disadvantages: shortage of reasoning about beliefs, inefficient reasoning, and requirements to successful execution of actions. By successful executions, we mean that an action just being executed causes respective effects. Otherwise, we say the executions are abnormal. *Suppose that an agent believes that a book is on the table but in fact it is the opposite, she still insists on picking up the book from the table. Hence, she certainly picks up nothing.* In this example, we know that agents reason about beliefs and may execute abnormal actions in most cases. At the same time, efficient reasoning is crucial to the practicability of high-level program execution. In this research, we try to overcome the disadvantages. We firstly extend the SitCalc to reason about beliefs and actions. Moreover, we will investigate the computation problem in our extension to the SitCalc. At last, we will solve the abnormal execution problem.

*This work was supported by the Natural Science Foundation of China under Grant No. 61073053.

4 Completed Work

We presented a general framework to reasoning about actions and knowledge/belief change in multi-agent settings on the basis of Baltag and Smets' work [2008]. Our framework is based on a multi-agent extension to the SitCalc (MASC), augmented by a plausibility relation over situations and another one over actions, which is used to describe different agents' perspectives on actions. When an action is executed, we update the agents' plausibility order on situations by giving priority to the plausibility order on actions, in line with the AGM approach of giving priority to new information. In this framework, our notion of knowledge satisfies S5 properties, and that of belief satisfies KD45 properties. Moreover, our notion of knowledge entails that of belief. As to the special case of belief change of single-agent, we discovered that four postulates are not reasonable, and presented AGM2', AGM8', DP1' and DP2' which are more reasonable than original ones. We showed that the whole weak AGM and DP postulates and majority of KM postulates are satisfied. Finally, we also gave properties concerning the change of common knowledge and belief of a group of agents. To the best of our knowledge, our work is the most general one based on the SitCalc that handles a wide range of multi-agent scenarios, knowledge and belief, noisy sensing and belief revision.

5 Future Work and Its Approaches

We will extend the completed work, and continue to study efficient reasoning in the MASC. We use progression technique to solve the dynamic reasoning problem since the efficiency of regression decreases dramatically when the sequence of actions becomes long. Because the entailment and progression problems for the MASC are both undecidable, we may resort to limited static reasoning and incomplete progression to implement reasoning in the MASC. The basic approach is the restriction of language. **Firstly**, we restrict the knowledge and belief to be second-order. Due to finite model of the restricted S5 propositional knowledge and KD45 propositional belief, the entailment problem is at most PSPACE. **Secondly**, we constraint two types of action models. If an action is public, all agents can know the domain of finite actions that contains the actual one. The corresponding action model contains finite actions that are equally plausible. If an action is private, only the executor knows which action was executed, and other agents believe nothing was changed. For this case, the action model contains two actions. One is the actual action and the other is a empty action nil whose meaning is nothing happened. Here, nil is more plausible than the actual one. Two types of action models are finite, so we can get several common knowledge and belief cases, and the progression problems may become decidable. **Moreover**, we require the preconditions and effects of actions to be objective, since whether an action can be performed successfully depends on objective facts and only objective facts can be affected by an action. **Finally**, we just maintain consistent belief and give priority to new information, but lift the principle of minimal change. Under the above restrictions, we believe that an efficient reasoning technique can be implemented.

After developing reasoning techniques, the last step is to design a high-level program interpreter. A high-level program consists of two components: *the basic action theory (BAT)*, and *the control program*. The BAT is used to describe the initial mental attitudes of agents and the preconditions and effects of actions. As has been said before, we will use the MASC to represent the BAT. On the other hand, the control program is transformed into a sequence of actions by the interpreter, according to the BAT and the scenario. After careful analysis, our variant of high-level program execution contains the following constructs: *primitive action*, *sequence*, *nondeterministic branch*, *nondeterministic choice of argument*, *iteration*, *conditional*, *loop* and *procedures*. We will give the transition semantics for the above constructs.

The remaining problem is how to handle abnormal execution of actions. In the constructs above, nondeterministic branch is helpful for the problem. We extend an action that must succeed in previous works to two actions. One is *wrt.* successful execution and the other one is *wrt.* the abnormal alternative. We construct a composite action from these two actions via nondeterministic branch, and use the new action instead of the original one.

6 Expected Contributions

To summarize, we expect to contribute the following:

1. A general framework to reasoning about actions and belief change in multi-agent settings in the SitCalc;
2. Limited but efficient static reasoning about the MASC;
3. Incomplete but efficient progression about the MASC;
4. A high-level program interpreter that incorporates reasoning about beliefs, abnormal execution of actions and efficient reasoning.

References

[Baltag and Smets, 2008] A. Baltag and S. Smets. A qualitative theory of dynamic interactive belief revision. *Texts in logic and games*, pages 9–58, 2008.

[Fan *et al.*, 2012] Y. Fan, M. Cai, N. Li, and Y. Liu. A first-order interpreter for knowledge-based golog with sensing based on exact progression and limited reasoning. In *Proc. AAAI*, 2012.

[Giacomo and Levesque, 1999] G. De Giacomo and H. Levesque. An incremental interpreter for high-level programs with sensing. *Logical Foundations for Cognitive Agents*, pages 86–102, 1999.

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

[Kelly, 2008] R. Kelly. *Asynchronous Multi-Agent Reasoning in the Situation Calculus*. PhD thesis, University of Melbourne, 2008.

[Levesque *et al.*, 1997] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. Golog: A logic programming language for dynamic domains. *J. Logic Programming*, 31:59–84, 1997.

[Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, 2001.

[Shapiro *et al.*, 1998] S. Shapiro, Y. Lespérance, and H. J. Levesque. Specifying communicative multi-agent systems. In *Agents and Multi-Agent Systems, Formalisms, Methodologices, and Applications*, pages 1–14. Springer, 1998.