Dynamically Generated Commitment Protocols in Open Systems

(JAAMAS Extended Abstract)

Akın Günay
School of Computer Eng.
Nanyang Technological University
Singapore
akingunay@ntu.edu.sg

Michael Winikoff
Department of Information Science
University of Otago
Dunedin, New Zealand
michael.winikoff@otago.ac.nz

Pınar Yolum
Department of Computer Engineering
Bogazici University
Istanbul,Turkey
pinar.yolum@boun.edu.tr

ABSTRACT

This extended abstract presents our work “Dynamically Generated Commitment Protocols in Open Systems” that appeared in the Journal of Autonomous Agents and Multi-Agent Systems [1].

Keywords

Commitment Protocol; Generation; Ranking

Protocols are used in multiagent systems to regulate interactions of agents. Typically, protocols are developed at design-time and embedded into the implementations of the agents. However, there are three shortcomings of this approach in open systems: (1) New types of agents, which do not comply with all the requirements of the design-time protocols, may join to an open system. (2) The environment of an open systems may evolve, and as a result, the design-time protocols may become insufficient to allow agents to achieve their goals in the changing environment. (3) Preferences of the agents may change over time, and accordingly the design-time protocols may fail to satisfy the new preferences of the agents.

For these reasons, we argue that agents in an open multiagent system should be able to derive new protocols at run-time, rather than completely relying on the design-time protocols. The ability to derive new protocols at run-time requires a formalism to define protocols, which can be interpreted by the agents. We define protocols in terms of *social commitments* [2], rather than sequences of message exchanges that are embedded into the implementations of the agents. The use of commitment protocols allow agents to reason about their situation, and carry out their interactions in a flexible manner while preserving their autonomy [3]. This paper proposes run-time generation of commitment-based interaction protocols.

Our contribution is not just a mechanism for creating a commitment protocol at run-time, but also a framework for the life-cycle of such protocols. To this end, we develop a complete agent process that defines how an agent can generate a set of candidate commitment protocols that allow the agent to achieve its goals, rank the candidate protocols according to the agent's preferences, and reach agreement with other agents on a suitable candidate protocol for enactment. Our main contributions are: (1) We develop two algorithms for the generation of commitment protocols, which use the agent's own capabilities, and also its knowledge of other agents' services and incentives to generate a set of candidate commitment protocols that support the agent's goals. (2) We develop a novel metric, namely risk-discounted utility, that combines utility and trust concepts for ranking the candidate commitment protocols. (3) We define a procedure that can be used by the agents to reach agreement on a suitable candidate protocol for enactment.

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

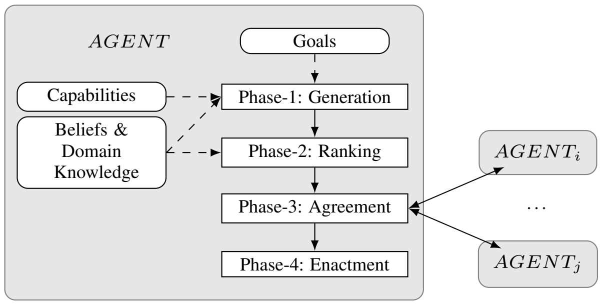

Figure 1: Process of an agent for run-time protocol enactment.

We present the four-phase agent process that we propose in Figure 1. The first phase of the process is the *generation* of a set of candidate commitment protocols by an agent that aims to achieve a certain goal state. All of the candidate protocols ensure that the agent reaches its goal state. Furthermore, the candidate protocols also take into account the goals and services of the other agents to create incentive and ensure their collaboration. In the second phase, the generating agent evaluates the candidate protocols, and *ranks* them according to its preferences. This ranking is subjective by nature, and different parties may prefer different protocols over others. After the ranking, in the third phase, the generating agent engages in negotiation with the other agents in order to reach *agreement* on a candidate protocol for enactment. Finally, in the last phase, the chosen candidate protocol is *enacted* by the agents.

Generation: We propose the algorithms PROTOCOLBASED and GOALBASED for generating a set of commitment protocols that allow the generating agent to achieve its goals. The algorithms use the generating agent's capabilities in conjunction with its beliefs about the other agents, which define the services that they provide, and also what they may expect for the provision of their services. The core criterion of our algorithms is to establish the necessary *support* for the goals of the generating agent. We say that an agent supports a goal, if either the goal can be achieved by the agent itself using its own capabilities, or there are established commitments

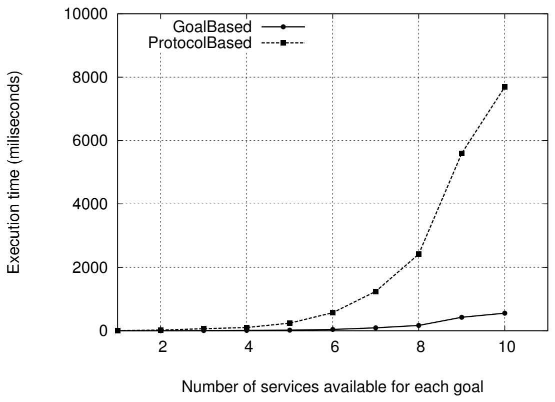

Figure 2: Execution times of ProtocolBased and GoalBased according to the number of available services.

from the other agents for the provision of the services that allow the agent to achieve its goal.

Our first algorithm ProtocolBased is a direct recursive implementation of the support definition using depth-first search. The algorithm starts from an empty protocol. At each step it considers one goal from the generating agent's goal set, and tries to establish support for it either by using the agent's own capabilities, or by creating a new commitment to utilize a service of another agent. As a result of this step, new goals may be generated, which represent the intermediary conditions that should be achieved by the generating agent to use her own capabilities (e.g., preconditions of the capabilities), and also to create incentive on other agents to provide their services (e.g., the expectations of the other agents in return for the provision of their services). The base case is reached if the agent can support a goal using her own capabilities in an unconditional manner. This procedure generates a tree structure, where each path from the root to a leaf node, in which the base case is supported, corresponds to a candidate commitment protocol.

ProtocolBased provides us a baseline algorithm. However, it is inefficient and suffers from lack of scalability, mainly because it finds support for a goal multiple times, if the goal is involved in the context of different candidate protocols. To overcome this drawback, we develop an efficient algorithm using the divide-and-conquer strategy, which we call GoalBased. This algorithm considers goals independently from the candidate protocols. Accordingly, for each individual goal of the agent the algorithm generates a set of sub-protocols that support only that goal. Then these sub-protocol are merged to build a complete protocol to support all the goals of the agent. This modular approach allows us to reuse the sub-protocols that are generated in the context of different candidate protocols. Hence, the algorithm generates sub-protocols for each goal only once and avoids redundant computation.

We conducted experiments to evaluate efficiency and scalability of our proposed algorithm. In the experiments, we used a data set that we generated systematically according to several parameters, such as the number of available services to support a goal, and the number of preconditions required for using the agent's capabilities. We show the execution times of our algorithms with respect to the number of available services to support a goal in Figure 2, which shows the efficiency and scalability of GoalBased.

Ranking: After the generation of the candidate protocols, the next phase is to evaluate and rank the candidate protocols to reflect the generating agent's preferences. To this end, we develop a metric for ranking commitment protocols, namely *risk-discounted utility*, that combines *utility* and *trust* concepts. The *utility* of a protocol for an agent reflects how much an agent would gain by enacting the protocol. The utility is basically the difference between the benefit of the protocol and its cost ($utility = benefit - cost$). Intuitively, each agent would want to maximize its utility, given that two protocols achieve the same goals. There may also be some protocols that are not acceptable to a given agent, i.e., protocols with negative utility. The *risk-discounted utility* extends utility by also considering risk. Some of the agents that enact a protocol may fail to fulfill their commitments, and accordingly the expected benefit of the protocol may not be realized. We therefore extend the definition of utility by *discounting* the benefits of a protocol based on the risk of the protocol due to other agents failures. The risk of a protocol is based on how trustworthy the agents enacting the protocol are for the various services that they provide in the context of the protocol. If all the agents are completely trustworthy, then there is no risk associated with a protocol. However, if any agent in the protocol is somewhat untrustworthy, then the protocol would be in risk, and the expected utility of the protocol may be jeopardized.

Agreement & Enactment: After the ranking of the candidate protocols, the next phase is to reach agreement between the agents on the enactment of one of the candidate protocols. To achieve this we develop a procedure that is led by the generating agent using the monotonic concession concept as follows. First, the generating agent selects the candidate protocol that it prefers most (e.g., the protocol that has the best risk-discounted utility). Then, it proposes creation of the candidate protocol's commitments to the corresponding debtors. If all the debtors accept to create the proposed commitments, an agreement is established over the protocol. Otherwise, the generating agent selects another candidate protocol that it prefers less (e.g., the protocol that is ranked next to the last offered protocol in terms of risk-discounted utility), and repeats the procedure. Note that in this phase, the debtors only accept to create the commitments, but they do not actually create the commitments. Actual creation of the commitments happens in the enactment phase, which follows the agreement phase. Enactment of a commitment protocol is system specific, hence we do not consider the details of this phase, and assume that enactment can be done in the targeted system. Note that our agreement procedure is independent from the ranking of protocols. Hence, alternative ranking metrics to risk-discounted utility can also be used for agreement.

In conclusion, in this paper we develop an agent process that defines how an agent can generate a set of candidate commitment protocols to support its goals, rank the candidate protocols according to its preferences, and reach agreement with other agents on a candidate protocol for enactment. This process allows agents of an open multiagent system to create and enact new protocols at run-time according to their changing needs and preferences, and reduces their dependency on the design-time protocols.

REFERENCES

[1] A. Günay, M. Winikoff, and P. Yolum. Dynamically generated commitment protocols in open systems. *Autonomous Agents and Multi-Agent Systems*, 29(2):192–229, 2015.

[2] M. P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7(1):97–113, 1999.

[3] P. Yolum and M. P. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 527–534, 2002.