Generating and Ranking Commitment Protocols (Extended Abstract)

Akın Günay
Department of Computer Engineering
Bogazici University
Istanbul, Turkey
akin.gunay@boun.edu.tr

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

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

ABSTRACT

Interaction is a fundamental part of multiagent systems, and is usually regulated by protocols. Typically, these protocols are defined at design-time. However, there exist situations where it is desirable to generate protocols at runtime. We develop an algorithm to generate commitment protocols considering the agent's goals, its capabilities and its domain knowledge about the other agents' goals and services. We then provide a method to rank these protocols in terms of their risk and benefit in order to select the one that best suits the agent's current state.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

Keywords

Commitment Protocols; Agent Interaction; Protocol Generation

1. PROTOCOL GENERATION

Interaction is a fundamental element of multiagent systems. Such interactions are generally regulated by interaction protocols that define the messages that can be exchanged among agents. *Social commitments* [2, 3] are widely used to represent interaction protocols, since they allow agents to reason about and carry out their interactions in a flexible manner.

Typically, protocols are defined at design time. Although this approach simplifies the design process of the multiagent system, it also has various drawbacks. In an open agent system, agents may come up with new tasks for which no protocol exists in the protocol library. Additionally, even if a protocol exists to do a certain task, since agents are heterogeneous, they may not comply with the protocol. For instance, a purchase protocol may require a credit card for payment. If the buyer does not have this capability, then the protocol cannot be used.

Accordingly, in this paper we argue that an agent should be able to generate protocols itself at run time. For this purpose we propose an algorithm to generate commitment protocols that allows an agent to achieve its goals when executed. To generate such protocols our algorithm considers the capabilities of the agent as well as the services provided by other agents. Besides, in order to induce other agents to provide their services, the algorithm also takes their goals into account in generating the commitment protocols.

In many cases, services and especially goals of the other agents may not be known completely. However, agents may acquire such information various ways. For instance, agents may advertise the services that they offer (e.g., selling goods) or a goal of an agent may be deduced using previous interactions (e.g., an agent that prefers to receive cash) or via domain knowledge (e.g., merchants aim to sell goods to receive payment). Accordingly, we assume that agents have domain knowledge (not necessarily complete) about the goals and services of other agents. This information can be used to generate possible commitment protocols.

We develop a recursive procedure to generate commitment protocols in which every goal of an agent a is supported by using either a's own capabilities or other agents' services. When one of a's own capabilities with a precondition is used to support a goal, then the algorithm tries to find support also for this precondition. On the other hand, when a service of another agent is aimed to be used to support a goal, then the algorithm generates a commitment, in which a offers a property to the service provider, which is believed to be desirable to it (i.e., one of its goals), in return to the service. The algorithm is sound and complete: if the agent's goals can be supported, then it will find a protocol, and it generates all (minimal) protocols that will support the agent's goals.

Example: consider a customer agent (Cus) that wants to have a certain type of furniture, but does not have the necessary materials nor the know-how to build the furniture. Considering the agents around herself, she can produce various protocols to get the job done: A first protocol could involve paying a retailer (Ret) to buy the furniture, a second protocol could involve paying and thus obtaining the materials from a store and then paying a builder (Bui) to put the furniture together, and so on. Our algorithm generates all possible protocols and thus enables the agent that generated the protocol to pick the most desired one and offer it to the participants. If the participants agree on the protocol then the necessary interactions follow. Otherwise, the agent can offer a different protocol to the participants. The algorithm generates 15 protocols, including $p_1 = \{\mathsf{C}(Bui, Cus, \text{MaterialsProvided} \wedge \text{CostPaid}, \text{HaveFurniture}),$ $\mathsf{C}(Ret, Cus, \text{MaterialsPaid}, \text{HaveMaterials})\}$ in which the builder commits to providing furniture if it is given materials and payment, and the retailer commits to providing materials if it is paid.

This paper builds on our earlier work [1] which defined an algorithm for generating protocols. The most significant new contribution is the consideration of ranking of protocols, which was not considered earlier. However, there are also other significant differences such as a more expressive language and formal results (soundness and completeness).

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

2. PROTOCOL RANKING

The key idea here is that the agent should be able to decide which protocols are more desirable for its current state by evaluating and ranking the protocols. In evaluating a protocol, there are two important concepts. One is the *risk* associated with the achievement of a protocol. A protocol might serve to achieve goals and have a low cost, but in reality the agents involved in the protocol might be unreliable, casting doubt on the successful execution. The second is the *benefit* of a protocol for an agent. The benefit is basically the difference between the *utility* of the protocol and its *cost*. Intuitively, each agent would want to maximize its benefit and reduce its risk.

Risk: In order to understand the risk of a protocol, we start from the trust relation among agents that are involved in a protocol. In general, an agent's trust in another depends on the particular service in question. An agent might trust a retailer for delivering furniture but may not trust him for actually assembling the furniture. Hence, we consider the trust of an agent a for another agent a' with respect to a particular service S, denoted as $T_a(a', \mathsf{S}_{a'}(d, u)) \in [0, 1]$. This trust value represents how likely a' is to complete service $\mathsf{S}_{a'}(d, u)$ from agent a's perspective. A trust value of 1 would mean that agent a believes a' would definitely carry out the service, whereas a trust value of 0 would mean that a believes a' will definitely *not* carry out the service. In general, the trust value is updated dynamically based on the outcomes of agents' interactions or new information coming in about the agent from other (trusted) sources. We assume that the agent has one of the many existing mechanisms in the literature for managing and disseminating trust.

Given $T_a(a', \mathsf{S}_{a'}(d, u))$, we proceed to define the trust that an agent a has in another agent's ability to bring about a proposition u (denoted $T_a(a', u)$); the trust that it has in a commitment being achieved (denoted $T_a(a', \mathsf{C}(a_1, a_2, q, r))$); and the trust that it has in a protocol p (denoted $T_a(p)$).

The key intuition is that trust in a protocol boils down to trusting each agent to play its part when it is called upon to do so. Note that due to the way in which protocols are generated, any required preconditions for services are taken care of (see example below).

The trust that agent a has in the ability of another agent a' to bring about proposition u can be defined in terms of the relevant services. One special case is that if a' does not have any relevant services, then the trust is 0. In other words, a considers all services that would enable u to be realized and combines the trust for these services using an auxiliary function $\oplus$. This auxiliary function can be defined using max, i.e. $x \oplus y = \max(x, y)$, meaning that the combined trust can at most be equal to the most trusted service.

The trust that an agent a has in other agents playing their parts in realizing a commitment is denoted $T_a(\mathsf{C}(a_1, a_2, q, r))$. In order for a commitment to be correctly discharged in a protocol we need the debtor to bring about r "and" for the creditor to bring about q (so that the debtor is obliged to bring about r). Formally we define:$T_a(\mathsf{C}(a_1, a_2, q, r)) = T_a(a_1, r) \otimes T_a(a_2, q)$where we also define $x \otimes y = x \times y$, and extend trust to deal with conjunctions: $T_a(a', t \wedge t') = T_a(a', t) \otimes T_a(a', t')$. Note that if agent a is either the debtor or the creditor (and it has the ability to bring about the relevant condition), then one of the trust terms reduces to 1, since agents trust themselves. Next we define trust in a protocol, $T_a(p)$ as the combined trust in the commitments: since all the commitments need to be realized for the protocol to succeed the trust is "anded" (i.e, combined using $\otimes$): Hence, in order to have a high trust in a protocol, every agent that participates in the protocol should be highly trusted for the services that they provide as part of the protocol. Finally, we define the risk of a protocol p as simply being the inverse trust: $R_a(p) = 1 - T_a(p)$.

After an agent calculates the risk of a protocol, it can decide whether the risk is acceptable for its current situation. Depending on the situation and agent characteristics, different risk ranges might be accepted. For example, a risk-taker agent in the face of an emergency might choose to enter a protocol with a risk value of 0.9, whereas a risk-averse agent in a non-emergency setting might not enter a protocol with a risk value of 0.5.

Benefit: The benefit of a protocol captures what an agent gains by actually carrying out a protocol. The overall benefit of each protocol may vary from each agent's perspective. In defining the benefit of a protocol we assume that we know the cost of each service ($cost_a(\mathsf{S}_a(d, u))$). We then define the cost of a *proposition* as: $cost_a(u) = \max_{\mathsf{S}_a(d', u') \in \mathcal{S} \wedge u' \Rightarrow u} cost_a(\mathsf{S}_a(d', u'))$. We use a maximum since in general we may not be able to decide or control which service is usable to achieve the desired property. We also assume that we know each agent's assigned utility for each proposition ($utility_a(u)$).

To calculate the overall cost (or utility) of a protocol we consider the overall set of propositions involved, and then sum their costs (or utilities). We consider all the propositions for which the agent is a debtor in the cost summation and all the propositions for which the agent is a creditor in the utility summation. We then find the benefit as $benefit_a(p) = utility_a(p) - cost_a(p)$ Given the benefit of a protocol, we can then consider whether a protocol is acceptable to an agent. The basic intuition is that a protocol is acceptable if it has a benefit that is ≥ 0.

If we compare protocols in terms of cost and benefit, we see that the rank of a protocol might be significantly different based on the criterion that is used. For example, a protocol may have a high benefit, but also a high risk. In principle, an agent would want to combine the rankings from different metrics to achieve a balance between benefit and risk. We use Borda count to combine the risk ranking and benefit ranking to obtain an overall ranking for the protocol. Generally, the best ranked protocol would be first offered to the participants since that is the most preferred protocol from the generating agent's perspective.

Our work here opens up interesting directions for future research. One direction is the variations on benefit calculation, where different ways could reflect other concerns about protocol executions. For example, an agent that would want minimal loss in any protocol can can compute all subsets of a protocol, calculate each subset's benefit, and require that the benefit be positive for all subsets. Another direction could be in incorporating rankings from different metrics. While Borda count served our purpose well, other election methods could also be investigated.

3. REFERENCES

[1] A. Günay, M. Winikoff, and P. Yolum. Commitment protocol generation. In *10th International Workshop on Declarative Agent Languages and Technologies (DALT)*, 2012.

[2] M. P. Singh. An Ontology for Commitments in Multiagent Systems. *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 1st Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 527–534, 2002.