Programming Interaction-Oriented Cognitive Agents

Demonstration Track

Matteo Baldoni
University of Turin
Turin, Italy
matteo.baldoni@unito.it

Amit K. Chopra
Lancaster University
Lancaster, United Kingdom
amit.chopra@lancaster.ac.uk

Munindar P. Singh
North Carolina State University
Raleigh, NC, USA
mpsingh@ncsu.edu

ABSTRACT

Interactions are central to the notion of multiagent systems. Notions such as commitments and protocols enable modeling interactions; however, they are not adequately supported in cognitive programming models such as Jason. We demonstrate novel programming abstractions for engineering Jason agents that communicate on the basis of commitments and protocols. Specifically, we demonstrate how to specify commitments and protocols; automatically generate role-specific Jason adapters from them; and use the generated adapters toward implementing an agent's business logic. Our approach shines in the implementation of flexible, loosely-coupled agents, long a challenge for BDI-based agent programming approaches.

KEYWORDS

Decentralization; Interaction Protocols; BDI; Programming Model

ACM Reference Format:
Matteo Baldoni, Amit K. Chopra, and Munindar P. Singh. 2025. Programming Interaction-Oriented Cognitive Agents: Demonstration Track. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 3 pages.

1 INTRODUCTION

In 2012, Michael Winikoff [14] highlighted two shortcomings about agent-oriented programming languages (AOPLs). One, despite the importance of modeling interactions in multiagent systems (MAS), AOPLs supported little more than primitives for sending and receiving messages. He saw the use of such primitives as transferring control between agents and drew an unflattering analogy with the use of *gotos* in programming. Two, *interaction protocols*, typically expressed in notations such as AUML [8], were *message-centric* and overconstrained the interactions between agents. With the aim of supporting robustness and flexibility in interactions, as agent autonomy demands, Winikoff advocated higher-level abstractions that hid low-level messaging concerns. In 2024, AOPLs still suffer from the shortcomings Winikoff highlighted.

In recent work [1, 2], we have developed Azorus, a programming model for multiagent systems that combines cognitive abstractions with high-level, flexible models of multiagent interaction. In Azorus, we capture the meaning of interaction via a specification of commitments [3–5, 13, 15] and the operational constraints on interaction via information protocols expressed in the Blindingly Simple Protocol Language (BSPL) [6, 7, 9–11]. For programming agents, we build upon Jason [12] to fully exploit an agent's cognitive autonomy through the agent's goals, beliefs, and intentions. The centerpiece of Azorus is the generation of Jason adapter that supports an agent programming interface ("API") that enables engineering loosely coupled, flexible, and decentralized multiagent systems.

Our approach demonstrates how to overcome the limitations pointed out by Winikoff. Our approach overcomes shortcomings of message-centric interaction protocols, such as *incompatibilities* between agents due to the message schemas being blended into business logic; *semantic errors* due to a lack of a formal model; and *inflexibility* due to the programmer having to maintain the protocol state via a state machine. Moreover, it fully exploits the *agent's social autonomy* through the adoption of commitments and information protocols.

2 SPECIFYING MULTIAGENT SYSTEMS

Listing 1 gives the Azorus specification of a MAS for conducting ebusiness. The first half of the listing gives a BSPL protocol *Ebusiness*. The protocol specifies the roles and the message schemas. The protocol is specified declaratively via information constraints. We refer to an agent's communication history as its *local state*. The basic idea is that in any protocol enactment, as identified by ⌜key⌝ parameters, an agent can send any message whose ⌜in⌝ parameters already known (that is, bindings for them exist in the agent's local state) and whose ⌜out⌝ parameters are not already known (bindings for them don't exist in the local state). The *Ebusiness* protocol thus captures the operational constraints on protocol enactments. In contrast to protocols specified as communication state machines, it is highly flexible, e.g., allowing *shipment* to be sent any time after *offer*.

The second half of the listing gives the commitments in a language inspired from Cupid [4]. The commitments specify the meanings of the messages in the*Ebusiness* protocol. The commitment `OfferCom` specifies that *offer creates* a commitment (instance) from SELLER to BUYER. This commitment is *detached* if *transfer* happens within 5 time units (for purposes of this paper, seconds) of the creation and `Payment` in the *transfer* is at least as much as `Price` in the *offer*. The commitment *expires* (fails to be detached) if either of these conditions is not met. The commitment is *discharged* if *shipment* happens within 5 time units of being detached. The commitment is *violated* if it fails to be discharged, that is, if *shipment* fails to occur within the stipulated time. The other commitments have analogous readings.

Listing 1: The *Ebusiness* BSPL protocol and the commitments that capture the meaning of the messages in the protocol.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

```
Ebusiness {
  roles Buyer, Seller, Bank
  parameters out Id key, out Item, out Price, out
      Status

  Seller -> Buyer: offer[out Id key, out Item, out
      Price]
  Buyer  -> Seller: accept[in Id key, in Item, in Price
      , out Decision]
  Buyer  -> Bank: instruct[in Id key, in Price, out
      Details]
  Bank   -> Seller: transfer[in Id key, in Price, in
      Details, out Payment]
  Seller -> Buyer: shipment[in Id key, in Item, in
      Price, out Status]
  Seller -> Bank:  refund[in Id key, in Item, in
      Payment, out Amount, out Status]
}
 commitment OfferCom Seller to Buyer
  create offer
  detach transfer[, created OfferCom + 5]
       where "Payment>=Price"
  discharge shipment [, detached OfferCom + 5]

 commitment RefundCom Seller to Buyer
  create offer
  detach violated OfferCom
  discharge refund[, detached RefundCom + 2]
     where "Amount >= Payment"

 commitment TransferCom Bank to Seller
  create instruct
  discharge transfer[, created TransferCom + 2]
       where "Payment=Price"
```

3 TOOLING

Given the MAS specification and the role an agent wants to play, our tooling generates a Jason adapter (the red components) that lets the agent query for

- *enabled* messages (partial message instances whose ⌜in⌝ parameters are known but ⌜out⌝ parameters are unknown). These are the messages that an agent may potentially emit at that point in the agent's execution, and
- commitments in particular states.

Depending on the agent's internal reasoning (which includes querying commitment states), an agent may flesh out an enabled message by supplying bindings for its out parameters and *attempt* to send it (attempts may sometimes fail due to concurrency). Thus the API for programming agents consists the queries and attempt. Our tooling generates the requisite plans to support the API.

Azorus is more general than traditional agent programming approaches, which focus on the idea of message handling. In Azorus, a basic plan pattern is the following. A plan may be triggered by whatever event is of interest to the agent. The plans then run queries to compute commitments, enabled messages, and other conditions of interest. Depending on the results, it may *completing* the enabled messages (by providing bindings for the ⌜out⌝ parameters) and attempt to send them. Listing 2 shows a SELLER agent's plan for sending a *shipment*. It is triggered by AKC 1: describe the code below.

Listing 2: Commitments as queries in Azorus.

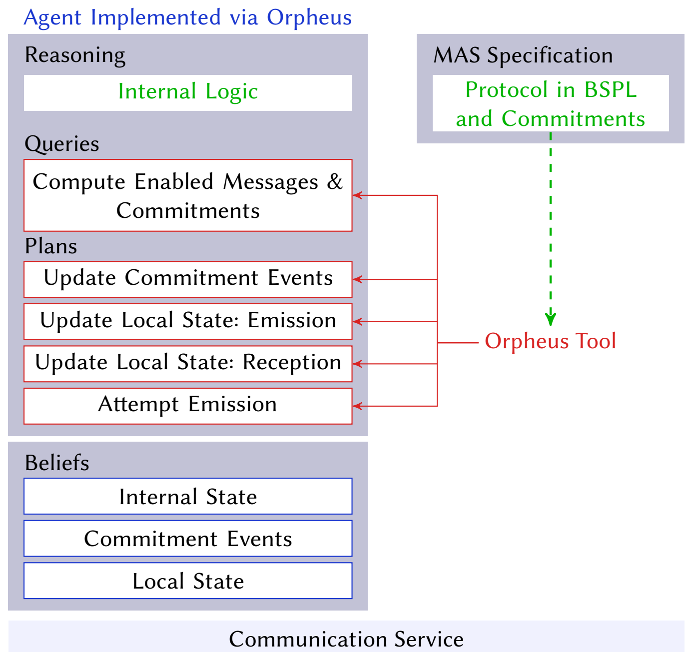

Figure 1: Tooling and programming model.

```
+!handle_form([shipment(Id, Item, Price, out)[receiver(
     Buyer)]|_])
 :  in_stock(Item) &
    enabled(shipment(Id, Item, Price, out)[receiver(
         Buyer)]) &
    now_detached_OfferCom(Seller, Buyer, Id, Item,
         Price, Bank, Payment, Timestamp) &
   & condition(Status)
 <- !attempt(shipment(Id, Item, Price, Status)[
      receiver(Buyer)]);
   -in_stock(Item).
```

4 CONCLUSIONS

We demonstrate Azorus, which provides an interaction-oriented programming model based on commitments and information protocols. A Its value proposition to engineering MAS is in reducing code complexity, avoiding repetition of business and interaction logic, and thereby facilitating the implementation of loosely coupled agents. Azorus supports the implementation of MAS on fully asynchronous communication services, multiparty (more than two) interactions, and multiple concurrent instances of a protocol.

5 REPRODUCIBILITY

The entire codebase (including tooling) and full versions of all examples are available at https://gitlab.com/masr.

ACKNOWLEDGMENTS

Thanks to the NSF (grant IIS-1908374) for partial support.

REFERENCES

[1] Matteo Baldoni, Samuel H. Christie V, Munindar P. Singh, and Amit K. Chopra. 2025. Orpheus: Engineering Multiagent Systems via Communicating Agents. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, Philadelphia, 1–9.

[2] Amit K. Chopra, Matteo Baldoni, Samuel H. Christie V, and Munindar P. Singh. 2025. Azorus: Commitments over Protocols for BDI Agents. In *Proceedings of the 24th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, Detroit.

[3] Amit K. Chopra, Samuel H. Christie V, and Munindar P. Singh. 2020. An Evaluation of Communication Protocol Languages for Engineering Multiagent Systems. *Journal of Artificial Intelligence Research (JAIR)* 69 (Dec. 2020), 1351–1393. https://doi.org/10.1613/jair.1.12212

[4] Amit K. Chopra and Munindar P. Singh. 2015. Cupid: Commitments in Relational Algebra. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI)*. AAAI Press, Austin, Texas, 2052–2059. https://doi.org/10.1609/aaai.v29i1.9443

[5] Amit K. Chopra and Munindar P. Singh. 2016. Custard: Computing Norm States over Information Stores. In *Proceedings of the 15th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, Singapore, 1096–1105. https://doi.org/10.5555/2936924.2937085

[6] Samuel H. Christie V, Amit K. Chopra, and Munindar P. Singh. 2022. Mandrake: Multiagent Systems as a Basis for Programming Fault-Tolerant Decentralized Applications. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 36, 1, Article 16 (April 2022), 30 pages. https://doi.org/10.1007/s10458-021-09540-8

[7] Samuel H. Christie V, Munindar P. Singh, and Amit K. Chopra. 2023. Kiko: Programming Agents to Enact Interaction Protocols. In *Proceedings of the 22nd International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, London, 1154–1163. https://doi.org/10.5555/3545946.3598758

[8] Marc-Philippe Huget and James Odell. 2004. Representing Agent Interaction Protocols with Agent UML. In *Proceedings of the 5th International Workshop on Agent-Oriented Software Engineering (AOSE) (Lecture Notes in Computer Science, Vol. 3382)*. Springer, New York, 16–30. https://doi.org/10.1007/978-3-540-30578-1_2

[9] Munindar P. Singh. 2011. Information-Driven Interaction-Oriented Programming: BSPL, the Blindingly Simple Protocol Language. In *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, Taipei, 491–498. https://doi.org/10.5555/2031678.2031687

[10] Munindar P. Singh. 2012. Semantics and Verification of Information-Based Protocols. In *Proceedings of the 11th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, Valencia, Spain, 1149–1156. https://doi.org/10.5555/2343776.2343861

[11] Munindar P. Singh and Samuel H. Christie V. 2021. Tango: Declarative Semantics for Multiagent Communication Protocols. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI, Online, 391–397. https://doi.org/10.24963/ijcai.2021/55

[12] Renata Vieira, Álvaro F. Moreira, Michael J. Wooldridge, and Rafael H. Bordini. 2007. On the Formal Semantics of Speech-Act Based Communication in an Agent-Oriented Programming Language. *Journal of Artificial Intelligence Research (JAIR)* 29 (June 2007), 221–267. https://doi.org/10.1613/jair.2221

[13] Michael Winikoff. 2007. Implementing Commitment-based Interactions. In *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems*. 1–8.

[14] Michael Winikoff. 2012. Challenges and Directions for Engineering Multi-Agent Systems. *CoRR* abs/1209.1428 (2012), 12 pages.

[15] Pınar Yolum and Munindar P. Singh. 2002. Flexible Protocol Specification and Execution: Applying Event Calculus Planning using Commitments. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. ACM Press, Bologna, 527–534. https://doi.org/10.1145/544862.544867

REQUIREMENTS FOR THE DEMO

These are our requirements for the demo:

- A table and two/three chairs;
- A monitor with HDMI connectivity;
- Flip chart (if possible);
- poster display stand (if possible).