A new Hierarchical Agent Protocol Notation

JAAMAS Track

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

Nitin Yadav
University of Melbourne
Melbourne, Australia
nitin.yadav@unimelb.edu.au

Lin Padgham
RMIT University
Melbourne, Australia
lin.padgham@rmit.eduau

ABSTRACT

Agent interaction protocols are a key aspect of the design of multi-agent systems. However, commonly-used notations are, we argue, difficult to use, and lack expressiveness in certain areas. In this paper we present a new notation for expressing interaction protocols, focussing on key issues that we have found to be problematic. The notation is evaluated against criteria, and using a human subject evaluation of usability.

KEYWORDS

Protocol notations; agent-based systems

ACM Reference Format:
Michael Winikoff, Nitin Yadav, and Lin Padgham. 2018. A new Hierarchical Agent Protocol Notation. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 MOTIVATION

When designing multi-agent systems one important aspect of the software engineering design process is designing interactions between the agents. The outcome of this activity is captured as *interaction protocols* expressed in an *agent protocol notation*. However, in our 20 years of experience in designing, developing and teaching about agent systems, we have found protocol design to be one of the more problematic aspects, and a key issue concerns the notations used. Existing notations, of which Agent UML (AUML) is perhaps most popular, are, in our experience, difficult to use correctly, and are unable to capture certain key aspects of interactions.

One area where expressivity is lacking is in providing adequate support for *modularity*. For example, if we have a manufacturing cell, where a rotating table needs to be locked in order to perform some task, we would like to lift out the locking protocol, and use it in many places. However, it is not possible in AUML to express a locking protocol consisting of the sequence of messages *request-Lock*, *receive-Lock*, *release-Lock* and allow a variety of steps (depending on situation and who took the lock), between receive and release.

Another area where we have encountered difficulties is in the clean specification of protocols for collecting some specific set of data, where the data does not need to be collected in a particular order. This is typical of protocols for interacting with a human, and is indeed where we have encountered the issue multiple times. For example, a protocol for collecting from a person the information required to schedule a meeting may need to allow for information to be provided in (almost) any order.

These issues (and others) relate to the *expressiveness* of the notation. However, we also want a notation that is both *pragmatic* and *precise*. Precision means that the notation's syntax and semantics are clearly and unambiguously defined, which is important for comprehensibility, for avoiding ambiguity in the meaning of a given protocol, and for providing advanced tool support. A *pragmatic* notation is one that is usable by practicing software designers, which typically implies a graphical notation, eschewing logics, and dealing simply with common cases, such as sequence, selection and iteration. It is also desirable that a notation be as simple as possible. *What makes developing a good notation hard is that there is a trade-off between these objectives.* For instance, a simpler notation is easier to specify precisely, but is more likely to lack expressiveness.

This paper presents a new notational framework (Hapn: Hierarchical Agent Protocol Notation) that we have developed to address these issues. We are very aware that many notations have been proposed in the past. However, perhaps due to premature convergence in the field, we do not believe that there exists a notation that meets the needs outlined in this section. For further details we refer the reader to the JAAMAS paper [7]. An earlier version of the Hapn notation was presented at a COIN workshop [9], and a support tool was demonstrated at AAMAS 2015 [8].

2 THE HAPN NOTATION

The Hapn notation is based on hierarchical Finite State Machines (FSMs). We chose to use FSMs as the starting point since they are familiar, graphical, simple, and precisely defined. However, FSMs are not sufficiently expressive, and so we extend them by:

(1) Adding variables to protocols;
(2) Precisely defining the structure of transitions (Sender $\rightarrow$ Recipient(s) : msg(*args*)[*guard*]/*effects*), where a *guard* is a logical formula that can test, for example, conditions of variables, and an *effect* captures e.g. the binding of variables to values, and domain-specific actions; and
(3) Extending to *hierarchical* protocols by allowing states in a protocol to have sub-protocols. A crucial property of Hapn is synchronisation: where two protocols that are concurrently active transition on the same message, they must both transition *simultaneously*. This allows protocols to be reused and combined.

Figure 1 shows an example: an English auction protocol in Hapn. This protocol illustrates the use of these extensions to FSMs. The left side of Figure 1 shows the top level protocol $\mathcal{P}_{\mathcal{A}}$, with the start node having interface variables I for the item, and T for the finishing time, which are bound for the particular auction instance.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

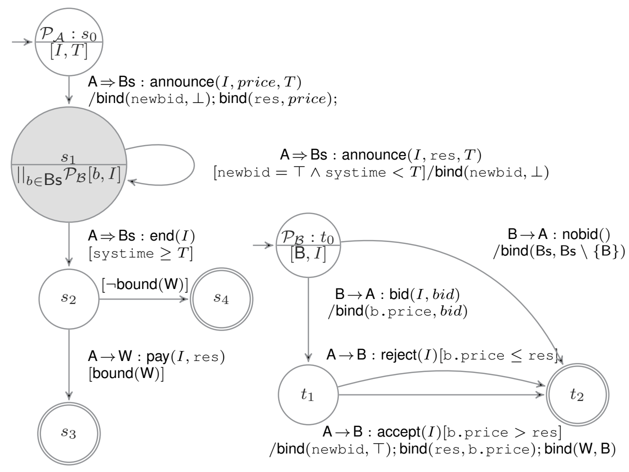
$\mathcal{P}_A : s_0$
$[I, T]$
A ⇒ Bs : announce$(I, price, T)$
/bind(newbid, ⊥); bind(res, $price$);
s_1
$||_{b\in \mathsf{Bs}}\mathcal{P}_{\mathcal{B}}[b, I]$
A ⇒ Bs : announce$(I$, res, $T)$
[newbid = ⊤ ∧ systime < T]/bind(newbid, ⊥)
A ⇒ Bs : end(I)
[systime ≥ T]
s_2
[¬bound(W)]
s_4
A → W : pay$(I$, res)
[bound(W)]
s_3
$\mathcal{P}_{\mathcal{B}} : t_0$
[B, I]
B → A : nobid()
/bind(Bs, Bs \ {B})
B → A : bid(I, bid)
/bind(b.price, bid)
A → B : reject(I)[b.price ≤ res]
t_1
t_2
A → B : accept(I)[b.price > res]
/bind(newbid, ⊤); bind(res, b.price); bind(W, B)

Figure 1: Auction protocol in HAPN

The first transition occurs when the auctioneer broadcasts an announcement stating the item, the starting price, and the finish time. The variables newbid and res are bound to ⊥ (False) and the starting price respectively, as the effect of this transition. The state s_1 has a bidding sub-protocol, which conceptually is a set of protocol instances, one for each bidder, with the particular bidder bound to b, and the item to I as previously.

The sub-protocol $\mathcal{P}_{\mathcal{B}}$ (right side of Figure 1) then specifies the interaction between the auctioneer and an individual bidder. One feature of the semantics of HAPN is that a state with sub-protocols may not be transitioned from unless all the sub-protocols are in a final state. In this case, state s_1 cannot proceed to another round, or (once time has elapsed) to the end of the auction, unless all bidding agents have either indicated nobid, or have bid, and had their bids accepted or rejected.

The HAPN notation has been precisely and formally defined. The longer description [7] includes two other case studies: a playdate scenario that demonstrates how flexible information collection can be easily yet precisely specified using HAPN, and a Holonic manufacturing protocol that demonstrates how modular and reusable protocols (e.g. locking) can be specified and reused.

3 EVALUATING HAPN

In order to evaluate HAPN we conducted a *feature* evaluation, where we assessed HAPN against a number of prominent existing notations[1] with respect to features that we identify and argue are important, and a *usability* evaluation using an empirical human subject experiment. Space precludes a discussion of more than just the key findings, and we refer the reader to the JAAMAS paper for details [7].

In order to evaluate our notation in comparison to other notations we consider a number of features: being precisely defined, having a graphical notation, being simple, supporting modularity, and allowing various important interaction patterns to be specified. We assess a notation's simplicity by counting distinct graphical element types following the methodology of Moody and van Hillegersberg [3, p30-31]. The expressiveness score in Table 1 is how many of the four interaction types[2] are handled.

	FSM	Petri Nets	AUML	State-charts	CMs	BSPL	HAPN
Precise	✔	✔		✔*	✔	✔	✔
Graphical	✔	✔	✔	✔			✔
Simple	4	3	17	11	N/A	N/A	5
Modularity			✔	✔	✔	✔	✔
Express.	0	1	3*	3	2	3	4*

Table 1: Criteria-based evaluation (key: "✔" = "yes", "*" = "yes, but ..." (see [7])

We observe that (1) Both FSMs and Petri nets are too simple: they are very simple (and hence amenable to being precisely defined), but lack expressivity; (2) Both AUML and Statecharts are too complex: they are fairly expressive (covering most desired cases), but they are complex and are either not precisely defined (AUML), or suffer from having multiple differing formal definitions (Statecharts); (3) Commitment Machines and BSPL are simple and expressive, but (as argued in the JAAMAS paper [7, Section 6.1]) there are concerns relating to their usability; and (4) HAPN manages to retain simplicity while providing expressiveness.

Finally, to assess the usability of HAPN we conducted a human-subject evaluation. Participants' ability to comprehend protocols did not vary significantly depending on the notation used (HAPN, AUML, Statecharts). This suggests that, at least as far as ability to read and interpret protocols, the three notations lead to comparable performance. In other words, *HAPN is as easy to read and understand as AUML or Statecharts*, despite being more expressive, and having a higher level of precision and formality. Interestingly, although participants' *performance* on a given protocol was not affected by the notation they used, the participant's *subjective assessment* of the notations did vary. Participants considered HAPN to be somewhat harder to read and understand, but the difference was not statistically significant. Each notation had some participants who ranked it as easiest to understand, and each notation had participants who ranked it as least easy to understand. However, while the *notation* used did not significantly affect the subjective assessment, the *protocol* did. If we consider for each participant how they ranked the notation that they used for the Auction task, then we found that almost all participants ranked the specific notation they used as easiest to understand, regardless of which notation was used. Similarly, the majority of participants ranked the notation that they used for the playdate protocol as being hardest to understand. In other words, regarding understandability, the protocol being considered mattered more than the notation that was used to present the protocol. The difference between the rankings, grouped by protocol, rather than by notation was statistically significant ($p = 0.001$).

Acknowledgements This work was partially supported by the Australian Research Council and Real Thing Entertainment Pty. Ltd. under Linkage grant number LP110100050.

[1]FSMs, Petri nets [4], AUML [2], Statecharts [1], Commitment Machines (CMs) [6, 10], and BSPL [5]

[2]Parallelism and synchronisation, exceptions, information-driven interactions, and interactions with multiple role instances.

REFERENCES

[1] David Harel. 1987. Statecharts: a visual formalism for complex systems. *Science of Computer Programming* 8, 3 (1987), 231 – 274. https://doi.org/10.1016/0167-6423(87)90035-9

[2] Marc-Philippe Huget and James Odell. 2005. Representing Agent Interaction Protocols with Agent UML. In *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004, Revised Selected Papers*, James Odell, Paolo Giorgini, and Jörg P. Müller (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 16–30. https://doi.org/10.1007/978-3-540-30578-1_2

[3] Daniel L. Moody and Jos van Hillegersberg. 2009. Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams. In *First International Conference on Software Language Engineering (Lecture Notes in Computer Science)*, Dragan Gasevic, Ralf Lämmel, and Eric Van Wyk (Eds.), Vol. 5452. Springer, 16–34. https://doi.org/10.1007/978-3-642-00434-6_3

[4] Wolfgang Reisig. 1985. *Petri Nets: An Introduction.* Springer-Verlag.

[5] 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)*. 491–498.

[6] Michael Winikoff, Wei Liu, and James Harland. 2004. Enhancing Commitment Machines. In *Declarative Agent Languages and Technologies II (Lecture Notes in Artificial Intelligence)*, João Leite, Andrea Omicini, Paolo Torroni, and Pınar Yolum (Eds.). Springer, 198–220.

[7] Michael Winikoff, Nitin Yadav, and Lin Padgham. 2018. A new Hierarchical Agent Protocol Notation. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 32, 1 (2018), 59–133. doi:10.1007/s10458-017-9373-9.

[8] Nitin Yadav, Lin Padgham, and Michael Winikoff. 2015. A Tool for Defining Agent Protocols in HAPN: (Demonstration). In *Autonomous Agents and MultiAgent Systems (AAMAS)*. IFAAMAS, 1935–1936.

[9] Nitin Yadav, Michael Winikoff, and Lin Padgham. 2015. HAPN: Hierarchical Agent Protocol Notation. In *International Workshop on Coordination, Organisation, Institutions and Norms in Multi-Agent Systems*.

[10] P. Yolum and M.P. Singh. 2002. Commitment Machines. In *Agent Theories, Architectures, and Languages (ATAL) (Lecture Notes in Computer Science)*, John-Jules Ch. Meyer and Milind Tambe (Eds.), Vol. 2333. Springer, 235–247.