Towards an Enthymeme-Based Communication Framework

Extended Abstract

Alison R. Panisson
Department of Computing – UFSC
Araranguá, Brazil
alison.panisson@ufsc.br

Peter McBurney
Department of Informatics – KCL
London, UK
peter.mcburney@kcl.uk

Rafael H. Bordini
School of Technology – PUCRS
Porto Alegre, Brazil
rafael.bordini@pucrs.br

ABSTRACT

In this work, we give an operational semantics for speech acts that BDI agents can use to communicate enthymemes. The approach uses argumentation schemes as common organisational knowledge to guide the construction of enthymemes by the proponents of arguments. Such schemes are also used to guide the reconstruction of the intended argument by the recipients of such enthymemes.

KEYWORDS

Multi-agent systems, Argumentation, Enthymemes

ACM Reference Format:

1 INTRODUCTION

Enthymemes are arguments in which one or more statements (which are part of the argument) are not explicitly stated [17]. They represent more realistic arguments, in the sense that arguments made by humans usually do not have enough explicitly stated premises for the entailment of the claim. This is because there is common knowledge that can be assumed by the arguers, which allows them to encode arguments into a shorter message by ignoring the common knowledge [1]. In particular, in this work, we use Argumentation Schemes (AS) [17] as common organisational knowledge to guide the construction of enthymemes by the proponents of arguments, as well as to guide the reconstruction of the intended argument by the recipients of such enthymemes.

Definition 1 (Argumentation Scheme). An argumentation scheme is a tuple \((SN, C, P, CQ)\) with \(SN\) the argumentation scheme name (which must be unique within the system), \(C\) the conclusion of the argumentation scheme, \(P\) the premises, and \(CQ\) the associated critical questions.

Definition 2 (Argument). An argument is a tuple \((S, c)^{\theta}_{sn_1}\), where \(sn_1\) is the name of the argumentation scheme \((sn_1, C, P, CQ) \in \Delta_{AS}\), \(\theta\) is a most general unifier for the premises in \(P\) and the agent’s current belief base, \(S\) is the set of premises and the inference rule of the scheme used to draw \(c\) (the conclusion). That is, \(S\) includes all instantiated premises from \(P\) — i.e., for all \(p \in P, \phi \in S\) — and the inference rule corresponding to the scheme \((P \Rightarrow C)\); the conclusion \(c\) is the instantiation \(C\theta\) such that \(S \models c\).

In our approach, AS are specified on top of a MAS as part of an organisational model [6], therefore they are shared by agents. Also, agents are able to instantiate and evaluate arguments based on [5, 8, 10, 11]. Furthermore, agents share knowledge regarding the organisational structure of the system, such as roles, authority links, etc. We denote all knowledge coming from the organisational infrastructure available to agents as \(\Delta_{org}\), and we emphasise that all agents are aware of the information in \(\Delta_{org}\). We denote all AS available for agents to instantiate arguments in the MAS as \(\Delta_{AS}\). Finally, we denote all knowledge available to an agent \(ag_i\) as \(\Delta_{ag_i}\), where \((\Delta_{org} \cup \Delta_{AS}) \subset \Delta_{ag_i}\). Enthymemes are defined as follow:

Definition 3 (Enthymeme). Let \((S, c)^{\theta}_{sn_1}\) be an argument to agent \(ag_i\). An enthymeme for \((S, c)^{\theta}_{sn_1}\) is a tuple \((S', c)^{\theta}_{sn_1}\) where \(S' = (S \setminus (\Delta_{org} \cup \Delta_{AS}))\).

2 SEMANTICS FOR ENTHYMEMES

Considering the performatives assert, question, justify, refuse, and accept [3, 4], we gave the operational semantics that formalises the construction (encoding) and reconstruction (decoding) of enthymemes associated with the speech act used by agents during dialogues. We define the semantics of speech acts for argumentation-based dialogues using enthymemes in AgentSpeak [15] (and in particular the Jason dialect [2]) using a widely-known method for giving operational semantics to programming languages [14].

The operational semantics is given by a set of inference rules that define a transition relation between agent configurations \(\langle ag, C, M, T, s \rangle\) [16], where: i) An agent \(ag\) is a set of beliefs \(bs\) and a set of plans \(ps\); ii) An agent circumstance \(C\) is a tuple \((I, E)\) where \(I\) is a set of intentions \((i, i', \ldots)\). Each intention \(i\) is a stack of partially instantiated plans and \(E\) is a set of events \(\{(te, i), (te', i'), \ldots\}\), in which each event is a pair \((te, i)\), where \(te\) is a triggering event and \(i\) is an intention — a stack of plans in the case of an internal event, or the empty intention \(T\) in the case of an external event; iii) \(M\) is a tuple \((In, Out)\) whose components characterise the aspects of communicating agents (typically asynchronous). \(In\) is the mail inbox, which includes all messages addressed to this agent. Elements of \(In\) have the form \((mid, id, if, cnt)\), where \(mid\) is a message identifier, \(id\) identifies the sender of the message, \(if\) is the illocutionary force of the message, and \(cnt\) its content; \(Out\) is where the agent posts messages it wishes to send; messages in this set have exactly the same format as above, except that here \(id\) refers to the agent to which the message is to be sent; iv) \(T\) is a tuple with temporary information originally defined in [16]; in this paper, we only need the i component, which keeps track of a particular intention being

---

\[1\] We use here only the components that are needed to define our semantics.
considered along the execution of a reasoning cycle; v) The current step within an agent’s reasoning cycle is symbolically annotated by \( s \in \{ \text{ProcMsg, ExecInt} \} \), with ProcMsg the step for processing a message from \( M \), and ExecInt for executing the selected intention.

The semantics of AgentSpeak makes use of “selection functions” which allow for user-defined components of the agent architecture. We use here only the \( S_M \) function, which is used to select one message from an agent’s mail inbox, as originally defined in [16]. In the interest of readability, we adopt the following notational conventions in our semantic rules: i) If \( C \) is an AgentSpeak agent circumstance, we write \( C_E \) to make reference to the \( E \) component of \( C \) (similarly for other components); ii) We write: \( b(s(id)) \) to identify the origin of a belief \( b \), where \( id \) is an agent identifier (\( s \) is an abbreviation for source); and \( b(\text{dec}(sn_i)) \) to identify information that was decoded from an enthymeme, guided by the AS \( sn_i \); iii) We use a function \( \text{prem}(S) \) which returns all premises in the support of the argument \( (S, c) \).

\[
\text{ExActSndAssert} \quad T_i = \{ \text{head } \leftarrow \text{send}(id, \text{assert}, c), h \} \\
(\text{ag}, C, M, T, \text{ExecInt}) \longrightarrow (ag', C', M', T, \text{ProcMsg})
\]

where:
\[
M'_{\text{Out}} = M_{\text{Out}} \cup \{( \text{mid, id, assert}, c) \}
\]

\[
C'_{\text{E}} = (C_{\text{E}} \setminus \{ \text{head } \leftarrow h \})
\]

\[
\text{ExActSndJustify} \quad T_i = \{ \text{head } \leftarrow \text{send}(id, \text{justify}, (\mathcal{S}, c)_{\text{sn}_i}^{\text{P}}), h \} \\
S \in \text{domain}_\mathcal{A}_{\text{org}} \quad E \in \text{prem}(S) \\
\mathcal{P} = \mathcal{P} \cup \{ (\mathcal{S}, c, \mathcal{P}, \mathcal{C}Q) \} \in \text{domain}_\mathcal{A}_{\text{org}} \\
(\text{ag}, C, M, T, \text{ExecInt}) \longrightarrow (ag', C', M', T, \text{ProcMsg})
\]

where:
\[
M'_{\text{Out}} = M_{\text{Out}} \cup \{( \text{mid, id, justify}, (\mathcal{S}, c)_{\text{sn}_i}^{\text{P}}) \}
\]

\[
\mathcal{S}' = \mathcal{S} \setminus \{ \text{domain}_\mathcal{A}_{\text{org}} \}
\]

\[
C'_{\text{E}} = (C_{\text{E}} \setminus \{ \text{head } \leftarrow h \})
\]

Sending an assert, question, accept, and refuse message: when an agent executes the internal action for sending a message with these performatives, that message is posted in the agent’s mailbox, \( M_{\text{Out}} \), and the current agent intention is updated, removing the internal action, given that its execution is completed. In the semantic rule \text{ExActSndAssert}, the intention being considered is given by \( T_i \), and it corresponds to \( \{ \text{head } \leftarrow \text{send}(id, \text{assert}, c), h \} \), in which the current step of the plan adopted to reach that particular goal is to execute the action \( \text{send}(id, \text{assert}, c) \). Thus, after executing that action, that particular intention is updated to \( \{ \text{head } \leftarrow h \} \), given that action has already been executed by the agent.

Sending a justify message: when an agent executes the internal action for sending a message with the performative justify, the agent needs to have an argument for that particular conclusion that was drawn using the AS \( sn_i \), according to Definition 2. Such argument is encoded into an enthymeme \( (\mathcal{S}', c)_{\text{sn}_i}^{\text{P}} \), where all common knowledge is removed from the argument support, i.e., the organisational knowledge \( \text{domain}_\mathcal{A}_{\text{org}} \) and the contents of the used AS \( \text{domain}_\mathcal{A} \) are removed. The corresponding message is posted in the agent’s mailbox, \( M_{\text{Out}} \), and the current agent intention is updated, removing the internal action, given that its execution is completed. The semantic rule \text{ExActSndJustify} implements the process for encoding an argument as an enthymeme.

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

\( \text{SENTENCE} \) \[\text{SENTENCE} \]

3 CONCLUSION

In this work, we proposed an operational semantics that specifies an enthymeme-based communication approach for BDI agents. Our approach is based on the idea that agents share AS and knowledge from the organisation, using that information to construct enthymemes from arguments, as well to reconstruct arguments from the enthymemes communicated by agents. We defined our operational semantics making reference to general components of the BDI architecture, therefore any language based on concepts such as beliefs, intentions, etc., could also benefit from our formalisation. Our work moves towards an enthymeme-based communication framework in MAS, extending our previous work [7], using the framework for AS proposed in [8, 10]. In future work, we intend to explore our approach for enthymemes in explainable MAS [9].

\footnote{Note that the semantic rules for the performatives question, accept, and refuse are similar to rule \text{ExActSndAssert}, thus they were omitted.}

\footnote{While we use, in this work, the \textit{thoughtful} attitude [12, 13], other agent attitudes could be used just as well.}
REFERENCES


