Implementação de Diálogos Argumentativos em Sistemas Multiagentes

Adler M. Cachuba, Ayslan T. Possebom

Instituto Federal do Paraná - IFPR
R. José F. Tequinha, 1400, Jardim das Nações – 87703-536 – Paranavaí – PR – Brasil

adlermateuself@gmail.com, ayslan.possebom@ifpr.edu.br

Abstract. *Argumentative dialogues consist in a dynamic process where two or more agents can, by means of an interaction among them, exchange information using arguments. These arguments represent the agents' opinions about certain information (claim), having a justification for those opinions (premise). This work introduces a proposal for computational implementation so that agents can communicate using argumentation. As a result, we have a set containing all the related arguments sent about some issue where different argumentation semantics can be applied.*

Resumo. *Diálogo argumentativo consiste em um processo dinâmico onde dois ou mais agentes podem, por meio de uma interação entre eles, trocar informações por meio de argumentos. Estes argumentos representam as opiniões dos agentes sobre determinada informação (conclusão do argumento), contendo uma justificativa para esta opinião (premissa do argumento). Este trabalho apresenta uma proposta de implementação computacional para que agentes possam se comunicar utilizando argumentação. Como resultado, obtêm-se uma tabela contendo todos os argumentos emitidos onde diferentes semânticas de argumentação podem ser aplicadas.*

1. Introdução

Quando um grupo de agentes pertencem a um determinado ambiente, frequentemente estes agentes se comunicam trocando mensagens entre eles. Estas mensagens são transmitidas do emissor aos seus receptores com o intuito de solicitar (ou fornecer) por alguma informação ou execução de alguma ação [Amgoud, Maudet and Parsons 2002]. Para que agentes se comuniquem e tomem decisões em conjunto (negociação, determinação de ações conjuntas, determinação do grau de consenso do grupo, etc.), as mensagens enviadas devem ter uma sequência que indica o fluxo do diálogo entre os agentes, ou seja, a conversação entre os agentes. Desta forma, podemos saber se as mensagens transmitidas foram aceitas ou rejeitadas pelos receptores, as contrapropostas, entre outros tipos de mensagens [Amgoud, Maudet and Parsons 2002].

O conteúdo das mensagens transmitidas pode representar um argumento, sendo a opinião do agente a respeito do que se está sendo discutido juntamente com a justificativa para esta opinião. De acordo com [Dung 1995], a argumentação consiste na troca de argumentos entre os agentes e na aplicação de semânticas para determinar os argumentos mais aceitos pelo grupo.

Para Greco (2015), um diálogo argumentativo é um processo social de interação entre os agentes que tenta resolver desacordos em um grupo. Na área da Inteligência Artificial, quando a argumentação é aplicada a apenas um agente, ela permite tanto a escolha de um conjunto de argumentos que se sustentam, quanto a escolha do melhor argumento. Já para sistemas multiagentes, a argumentação pode ser utilizada na escolha do melhor argumento a ser enviado ao grupo, ou então na criação de argumentos que sustentem ou que rejeitem um determinado ponto de vista.

Diferentes sistemas de argumentação foram propostos, tais como sistemas para argumentação abstrata [Dung 1995][Caminada 2008] e argumentação estruturada [Besnard and Hunter 2009][Amgoud and Prade 2009]. Apesar da argumentação estar em constante evolução, poucas implementações computacionais podem ser aplicadas em sistemas reais. Desta forma, este trabalho tem o objetivo de desenvolver uma implementação computacional onde um grupo de agentes pode se comunicar, trocando argumentos que representem suas opiniões a respeito dos argumentos transmitidos em um diálogo, de forma que no final do diálogo, o conjunto de argumentos possa ser analisado por alguma semântica de argumentação. Estas semânticas representam a aceitabilidade dos argumentos e podem indicar, por exemplo, uma alternativa de decisão preferida pelo grupo ou um conjunto de argumentos que melhor justificam a preferência ou discordância do grupo sobre determinada informação.

2. Referencial Teórico

Seja AG um conjunto finito de agentes argumentativos onde $AG = \{ag_1, \ldots, ag_n\}$ com $n > 0$, e $ag_i \in AG$ representando um agente. Um agente argumentativo é responsável pela criação de argumentos utilizando uma linguagem base que representa seus conhecimentos. Baseando-se no trabalho de Possebom, Morveli e Tacla (2016), um agente argumentativo pode ser definido como:

Definição 1: Um agente ag_i é uma tripla $< K, A, S >$ onde K representa a base de conhecimentos deste agente, A representa o argumento que está sendo discutido pelo grupo de agentes em um determinado instante e S representa um conjunto de argumentos a serem emitidos ao grupo.

O tipo de argumentação realizada pelos agentes argumentativos usa o conceito de argumento dedutivo [Besnard and Hunter 2009]. Um argumento é formado por um par $< \Phi, \alpha >$ onde Φ representa a premissa do argumento, ou seja, uma justificativa ou explicação para o argumento, e α representa a conclusão deste argumento. Tem-se que (i) a premissa do argumento não pode ser inconsistente, (ii) a premissa deve, por meio de um mecanismo de inferência, levar a conclusão do argumento, e (iii), a premissa deve ser um subconjunto mínimo de K (base de conhecimento do agente argumentativo).

Quando algum agente argumentativo transmite um argumento ao grupo, este argumento deve ser armazenado em suas respectivas bases A. Caso os agentes possuam contra-argumentos, eles devem ser armazenados em suas respectivas bases S. Um contra-argumento representa um conflito com outros argumentos já apresentados no diálogo. Estes conflitos podem gerar duas formas de ataques entre argumentos [Parsons and McBurney, 2003]: *undercut* e *rebuttal*. Em resumo, um argumento $\arg_i$ ataca por *undercut* um argumento $\arg_j$ quando a conclusão de $\arg_i$ é logicamente equivalente à negação de alguma informação presente na premissa de $\arg_j$. Tem-se que um argumento

$\arg_i$ ataca por *rebuttal* um argumento $\arg_j$ quando a conclusão de $\arg_i$ é logicamente equivalente à negação da conclusão de $\arg_j$.

Exemplo 1: considerando o conhecimento do agente representado por fórmulas em lógica proposicional, temos que $\arg_i =< \{a, a \rightarrow b\}, b >$ ataca $\arg_j =< \{\neg b, \neg b \rightarrow c\}, c >$ por *undercut*, enquanto que $\arg_i$ ataca $\arg_k =< \{d, d \rightarrow \neg b\}, \neg b >$ por *rebuttal*.

Para que a troca de mensagens entre os agentes possa ser controlada, utiliza-se um agente chamado mediador. Ele é responsável por controlar a troca de mensagens entre os agentes argumentativos, sincronizando a troca de mensagens e determinando quais argumentos emitidos são válidos para o diálogo atual. Um argumento é válido quando ele ainda não foi discutido pelo grupo até um determinado instante.

Definição 2: Um agente mediador med é uma tripla $< DT, WB, AGENDA, AG >$ onde DT representa uma tabela de diálogo que armazena os argumentos discutidos pelo grupo de agentes argumentativos, WB representa uma fila de agentes argumentativos que desejam emitir argumentos ao grupo, $AGENDA$ representa uma lista de argumentos emitidos por um determinado agente argumentativo e AG consiste em um conjunto de agentes argumentativos que pertencem a um diálogo.

O agente med realiza sequência de tarefas apresentadas no Algoritmo 1:

Algoritmo 1 Processo de diálogo argumentativo
Entrada: Conjunto de agentes argumentativos, Conjunto de assuntos a serem discutidos
Saída: Tabela de diálogo

```
1  procedimento run
2      para cada assunto faça:
3          configurar_diálogo
4          informar_assunto
5          questionar_ataques
6          solicitar_inscrição
7              para cada agente inscrito
8                  solicitar_argumentos
9                  para cada argumento enviado
10                     validar_argumento
11                     informar_argumento
12                     questionar_ataques
13                     solicitar_inscrição
14                 fim_para
15             fim_para
16     fim_para
17 fim_procedimento
```

A configuração do diálogo (linha 3) envolve a inicialização da DT, WB e $AGENDA$ para que o mediador possa receber argumentos e inscrições. Em seguida, o mediador informa o assunto que será discutido ao grupo (linha 4) e os agentes argumentativos buscam por contra-argumentos, armazenando-os em suas respectivas bases S (linha 5). Agentes cujas bases $S \neq \emptyset$ possuem argumentos para serem emitidos ao grupo e informam sua inscrição para falar quando solicitados (linha 6). Agentes que desejam emitir argumentos são inseridos em WB, não podendo ocupar mais de uma posição nesta fila ao mesmo tempo. O mediador solicita os argumentos ao agente argumentativo ocupante da primeira posição em WB (linha 8). Após enviar seus argumentos, sua base S é esvaziada, visto que estes argumentos já foram enviados para discussão. Para cada argumento enviado, este argumento é validado (linha 10) para garantir que ainda não tenha sido discutido pelo grupo (evitando repetições), ele é informado ao grupo (agentes argumentativos armazenam-no em suas bases A) (linha 11),

são questionados por contra-argumentos (agentes argumentativos armazenam seus ataques em suas bases S) (linha 12) e os agentes que possuem argumentos para serem emitidos ao grupo são inscritos novamente em WB (linha 13). O processo de diálogo é encerrado quando não existem mais assuntos para serem discutidos, não existam mais agentes inscritos em WB e não existem mais argumentos para discussão na $AGENDA$.

3. Implementação do diálogo argumentativo

Para implementar o diálogo argumentativo, utilizou-se frameworks já existentes capazes de representar agentes inteligentes e argumentos dedutivos. Observando-se as características reativas dos agentes argumentativos, decidiu-se optar pelo framework JADE[1] (JAVA Agent DEvelopment Framework).

Os agentes argumentativos e o agente mediador são apresentados na Figura 1. A comunicação com o ambiente (e, consequentemente, com os demais agentes) é realizado por meio da troca de mensagens. Em JADE, o envio de mensagens ocorre por meio da classe ACLMessage. De acordo com a performativa estabelecida na mensagem, o seu conteúdo indica uma determinada ação que o agente deve executar

Com relação ao agente argumentativo, ao receber uma mensagem com a performativa CFP, o agente deve enviar todos os seus argumentos presentes na base S e então esvaziá-la, sendo que esta resposta utiliza a performativa PROPOSE. Ao receber mensagem com performativa REQUEST_WHENEVER, o agente argumentativo deve buscar por ataques ao argumento de sua base A e armazená-los em S. Ao receber uma mensagem com performativa INFORM_REF, o agente deverá atualizar o argumento presente na base A. Para mensagens com performativa REQUEST_WHEN, caso a base $S \neq \emptyset$, o agente responde o valor "true" utilizando a performativa SUBSCRIBE, caso contrário, responde com o valor "false". Cada tipo de mensagem recebida é implementada em um comportamento CyclicBehaviour do agente.

Para representar argumentos dedutivos e construção de argumentos, utilizou-se o framework Tweety Project[2] por meio da biblioteca Deductive Argumentation. Como os argumentos criados são do tipo DeductiveArgument e estes objetos não podem ser transmitidos de um agente ao outro utilizando ACLMessage, os argumentos são convertidos para String ao serem enviados. No agente receptor, esta String é convertida novamente para objetos DeductiveArgument para serem manipulados como argumentos.

O agente mediador não possui uma característica reativa, visto que sua atividade consiste em controlar um processo de comunicação que segue uma ordem de execução. Neste sentido, utilizou-se um agente disponível no framework JADE chamado GatewayAgent, implementado pela classe JadeGateway. O uso de GatewayAgent se torna necessário visto que apenas agentes podem enviar e receber mensagens no framework. Com isso, o agente mediador pode se comunicar (por meio do seu GatewayAgent) com os demais agentes argumentativos do sistema e receber respostas quando solicitado.

As ações de informar argumento (performativa INFORM_REF) e assunto (performativa REQUEST_WHENEVER) atual, solicitar inscrições (performativa REQUEST_WHEN), solicitar argumentos (performativa CFP) e solicitar ataques ao

[1] JADE disponível em http://jade.tilab.com
[2] Tweety Project disponível em http://tweetyproject.org

argumento atual (performativa REQUEST_WHENEVER) são implementados utilizando o comportamento OneShotBehaviour. As ações para receber inscrições (performativa SUBSCRIBE) e receber os argumentos de um determinado agente (performativa PROPOSE) são implementados utilizando o comportamento SimpleBehaviour.

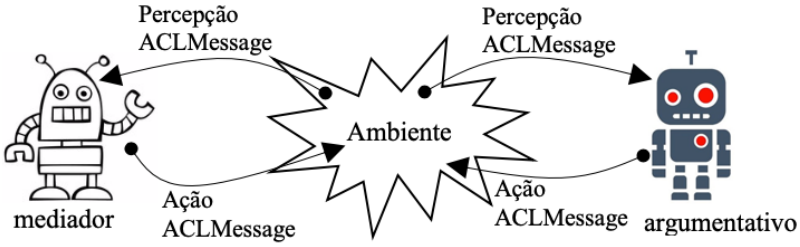

Figura 1. Agente argumentativo e mediador: mensagens recebidas (percepção) e mensagens transmitidas (ação)

4. Resultados Obtidos

A implementação dos diálogos argumentativos usando JADE e TweetyProject pode ser encontrada em https://github.com/apossebom/MASDialogue. O arquivo masdialogue/ Principal.java define as características iniciais para a execução, tais como definir todos os agentes argumentativos e suas respectivas bases de conhecimento e a lista de assuntos a serem discutidos. Em seguida, os agentes argumentativos são criados, seguido pela criação do agente mediador. Dois agentes argumentativos foram fornecidos: Maria e José. A base de conhecimentos do agente Maria é formada por $K_{\{Maria\}} = \{a, a \rightarrow b, \neg b, \neg a, c, c \rightarrow a\}$ e a base de conhecimentos do agente José é formada por $K_{\{José\}} = \{\neg a, b \rightarrow a, \neg c, \neg c \rightarrow d, e\}$. Considere os átomos das fórmulas sendo representados por: a prova fácil; b ir bem na prova; c estudar para a prova; d fazer recuperação; e pedir ajuda ao professor.

O agente med informa o assunto a para discussão (diálogo para saber se uma prova estava fácil) e solicita por ataques. Os agentes José e Maria possuem argumentos para $\neg a$ e se inscrevem para falar. Após José emitir seu argumento $arg_0 = \langle\{\neg a\}, \neg a\rangle$, este argumento é validado e inserido em DT, informado ao grupo e novos ataques são encontrados. O próximo agente a falar, Maria, envia seus argumentos (ataques ao assunto a ou a outros argumentos emitidos no diálogo, no caso, ataques a arg_0). Ao enviar seus argumentos para med, estes argumentos, um a um, serão validados, inseridos em DT, informados ao grupo e novos ataques ao argumento atual sendo discutido são solicitados. O diálogo se encerra quando não existem mais agentes inscritos para falar (a base $S = \emptyset$ para todos os agentes e $WB = \emptyset$) e todos os argumentos já tiverem sido discutidos pelo grupo ($AGENDA = \emptyset$). A execução do diálogo sobre a gerou como resultado a tabela de diálogo apresentada na Tabela 1. Convertendo o diálogo para grafo de argumentos [Dung 1995] e, por meio de alguma semântica de argumentação, podemos observar que uma extensão pode sugerir uma conclusão para o diálogo. Por exemplo, utilizando semântica *Preferred* no conjunto com maior número de argumentos preferidos (ex: $\{arg_0, arg_3, arg_4, arg_5, arg_6, arg_{14}, arg_{15}, arg_{16}, arg_{17}\}$), tem-se que as fórmulas destes argumentos sugerem $\neg a, \neg b$ e $\neg c$, indicando que a prova não estava fácil, não foram bem na prova e não estudaram o suficiente).

Tabela 1. Tabela de diálogo para o assunto *a*.

Sequência	Emissor	Argumento
0	José	<{ !a },!a>
1	Maria	<{ a },a>
2	Maria	<{ c, !clla },a>
3	Maria	<{ !allb, !b },!a>
4	José	<{ !c },!c>
5	Maria	<{ !a, !clla },!c>
6	Maria	<{ !allb, !b, !clla },!c>
7	Maria	<{ !allb, a },b>
8	Maria	<{ c },c>

Sequência	Emissor	Argumento
9	Maria	<{ !allb, c, !clla },b>
10	Maria	<{ !b, a },!(!allb)>
11	Maria	<{ !a, c },!(all!c)>
12	Maria	<{ !b, c, !clla },!(!allb)>
13	Maria	<{ !allb, !b, c },!(all!c)>
14	Maria	<{ !allb },!allb>
15	Maria	<{ !b },!b>
16	Maria	<{ !clla },all!c>
17	José	<{ !blla, !a },!b>

4. Considerações Finais

Este trabalho apresentou uma proposta de implementação de diálogos argumentativos para ser utilizado em sistemas multiagentes, onde os argumentos emitidos/recebidos pelos agentes possuem uma estrutura definida (premissa e conclusão) e utilizam uma linguagem lógica (lógica proposicional) para representar o conhecimento dos agentes.

Dois tipos de agentes foram implementados: argumentativo e mediador. O agente argumentativo é o responsável por construir argumentos e argumentar. O agente mediador é o responsável por conduzir o processo de diálogo, sincronizando a troca de mensagens e garantindo que os argumentos discutidos sejam válidos. A partir desta implementação, diferentes abordagens podem ser estendidas, tais como calcular o consenso do grupo sobre determinado argumento ou sobre alternativas de decisão, aplicação de semânticas de argumentação ou ponderação dos argumentos.

Referências

Amgoud, L. and Maudet, N. and Parsons, S. (2002).An argumentation-based semantics for agent communication languages. In: Proceedings of the 15th ECAI, p. 38–42.

Amgoud, L. and Prade, H. (2009) Using arguments for making and explaining decisions. Artificial Intelligence, Elsevier B.V., v. 173, n. 3-4, p. 413–436.

Besnard, P. and Hunter, A. (2009). Argumentation based on classical logic. In Argumentation in Artificial Intelligence, pages 133–152. Springer.

Caminada, M. (2008). A gentle introduction to argumentation semantics. Disponível em: <https://users.cs.cf.ac.uk/CaminadaM/publications/Semantics_Introduction.pdf>. Acessado em: 24/03/2019.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial intelligence, 77(2):321–357.

Greco, S. (2015). Argumentative Dialogue. Key Concepts in Intercultural Dialogue, No. 73. Disponível em: < https://centerforinterculturaldialogue.files.wordpress.com/2015/10/kc73-argumentative-dialogue.pdf>. Acessado em: 24/03/2019.

Parsons, S. and McBurney, P. (2003). Argumentation-based dialogues for agent coordination. Group Decision and Negotiation, 12(5):415–439.

Possebom, A. T. and Morveli, M. and Tacla, C. A. (2016) Protocolo para diálogos argumentativos no auxílio da decisão consensual em sistemas multiagentes. WESAAC 2016, pp.169-174.