UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

HELIO HENRIQUE LOPES COSTA MONTE-ALTO

RACIOCÍNIO DERROTÁVEL DISTRIBUÍDO BASEADO EM ARGUMENTAÇÃO PARA AMBIENTES ABERTOS E DINÂMICOS

TESE

CURITIBA

2021

HELIO HENRIQUE LOPES COSTA MONTE-ALTO

RACIOCÍNIO DERROTÁVEL DISTRIBUÍDO BASEADO EM ARGUMENTAÇÃO PARA AMBIENTES ABERTOS E DINÂMICOS

Argumentation-Based Distributed Defeasible Reasoning for Open and Dynamic Environments

Tese apresentada como requisito para obtenção do título de Doutor em Ciências, do Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, da Universidade Tecnológica Federal do Paraná (UTFPR) – Área de Concentração: Engenharia de Computação.

Orientador: Prof. Dr. Cesar Augusto Tacla
Coorientadora: Dra. Miriam Mariela Mercedes Morveli-Espinoza

CURITIBA

2021

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

HELIO HENRIQUE LOPES COSTA MONTE ALTO

RACIOCÍNIO DERROTÁVEL DISTRIBUÍDO BASEADO EM ARGUMENTAÇÃO PARA AMBIENTES ABERTOS E DINÂMICOS

Trabalho de pesquisa de doutorado apresentado como requisito para obtenção do título de Doutor Em Ciências da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Engenharia De Computação.

Data de aprovação: 25 de Novembro de 2021

Prof Cesar Augusto Tacla, Doutorado - Universidade Tecnológica Federal do Paraná

Prof Ayslan Trevizan Possebom, Doutorado - Instituto Federal de Educação, Ciência e Tecnologia do Paraná (Ifpr)

Prof Fabiano Silva, Doutorado - Universidade Federal do Paraná (Ufpr)

Prof Gleifer Vaz Alves, - Universidade Tecnológica Federal do Paraná

Prof Gustavo Alberto Gimenez Lugo, Doutorado - Universidade Tecnológica Federal do Paraná

Prof.a Miriam Mariela Mercedes Morveli Espinoza, - Universidade Tecnológica Federal do Paraná

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 25/11/2021.

Dedico este trabalho a minha família: aos meus pais, Helio e Dinalva, aos meus irmãos, Eric e Eduardo, e especialmente a minha esposa Gabriela, pelo carinho e incentivo, e pelos momentos de ausência e dificuldades durante o desenvolvimento desta obra.

AGRADECIMENTOS

Primeiramente a meu Deus e Pai, e meu Senhor Jesus Cristo, pela minha vida, pela salvação que recebi gratuitamente pela Sua graça e amor, por Seu sangue derramado por mim na cruz, por ter me dado forças para concluir esta tese, e pela esperança de Sua vinda.

A minha esposa, Gabriela, pelo amor, paciência e apoio durante todo o longo processo de doutoramento.

Aos meus pais, Dinalva e Helio, pelo carinho, incentivo e total apoio em todos os momentos da minha vida.

Ao meu orientador e coorientadora, Prof. Cesar Augusto Tacla e Mariela Morveli-Espinoza, que me auxiliaram durante todo o processo, indicando caminhos e ideias, me incentivando, e pela confiança em mim depositada.

Ao Prof. Ayslan Trevizan Possebom, que me incentivou em momentos em que pensei em desistir, e que também colaborou em diversos momentos na escrita de artigos e nas discussões sobre o trabalho.

À Profª Elisa Hatsue Moriya Huzita, que me introduziu ao trabalho acadêmico e sempre me incentivou durante a graduação e mestrado.

Aos professores Fabiano Silva, Gustavo Giménez-Lugo e Gleifer Vaz Alves por terem lido esta tese e participado da banca de qualificação e de defesa, e pelos comentários e sugestões de melhorias e trabalhos futuros.

Ao Prof. Antonis Bikakis pelas sugestões de ideias que me ajudaram a achar um caminho para o tema desenvolvido na tese.

A todos os professores e colegas da Universidade Tecnológica Federal do Paraná que fizeram parte dessa jornada.

Aos colegas e ex-alunos da Universidade Federal do Paraná.

Enfim, a todos os que de alguma forma contribuíram para a realização deste trabalho.

RESUMO

MONTE-ALTO, Helio Henrique Lopes Costa. **Raciocínio Derrotável Distribuído Baseado em Argumentação para Ambientes Abertos e Dinâmicos**. 2021. 189 f. Tese (Doutorado em Engenharia Elétrica e Informática Industrial) – Universidade Tecnológica Federal do Paraná. Curitiba, 2021.

Este trabalho apresenta um modelo de raciocínio distribuído e não-monotônico, no qual entidades distribuídas – agentes – podem ter conhecimentos que, tomados em conjunto, levam a conclusões conflitantes. Tais conflitos são resolvidos por meio de uma abordagem de argumentação estruturada baseada em regras com uma função de cálculo de força de argumentos que leva em consideração o grau de confiança que os agentes têm uns nos outros e o grau de certeza acerca da correspondência entre conhecimentos heterogêneos adquiridos de diferentes agentes. Presume-se, também, que o ambiente no qual esses agentes estão inseridos é aberto e dinâmico, de modo que os agentes podem entrar e sair do ambiente a qualquer momento, assim como ter suas crenças atualizadas constantemente. Isso gera a necessidade de que os agentes sejam capazes de consultar agentes arbitrários acerca de conhecimentos específicos, mesmo sem saberem *a priori* se os agentes consultados possuem tais conhecimentos. Também se considera a possibilidade de um agente possuir conhecimentos relacionados ao seu foco atual – isto é, conhecimentos relacionados a sua situação atual, localização, atividade, objeto(s) de interesse e/ou objetivos – que podem ser relevantes para que outros agentes possam raciocinar corretamente acerca de uma consulta enviada por aquele agente. Assim, propõe-se que uma consulta enviada de um agente para o outro permita que o primeiro compartilhe conhecimentos de foco que são relevantes no escopo da consulta. Portanto, é proposto um modelo de argumentação estruturado distribuído baseado em regras que formaliza as estruturas de raciocínio e a semântica de argumentação, assim como um algoritmo distribuído de processamento de consultas baseado nesse modelo. Também são apresentadas formas de cálculo alternativas de força de argumentos baseadas em diferentes intuições sobre possíveis atitudes mentais de um agente e que tiram proveito das estruturas de argumentação de maneiras distintas. Esta tese também abre possibilidades para trabalhos futuros, tal como no escopo de revisão de crenças e geração de explicações baseadas nas estruturas de argumentação construídas pelos agentes.

Palavras-chave: argumentação; raciocínio derrotável; sistemas multiagente; raciocínio distribuído; ambientes abertos e dinâmicos.

ABSTRACT

MONTE-ALTO, Helio Henrique Lopes Costa. **Argumentation-Based Distributed Defeasible Reasoning in Open and Dynamic Environments**. 2021. 189 p. Thesis (PhD in Electrical Engineering and Industrial Informatics) – Universidade Tecnológica Federal do Paraná. Curitiba, 2021.

This work presents a distributed and non-monotonic reasoning model, in which distributed entities – agents – may have knowledge that, taken together, lead to conflicting conclusions. Such conflicts are resolved through a rule-based structured argumentation approach with an argument strength calculation function that takes into account the degree of trust agents have in each other and the degree of certainty about the correspondence between heterogeneous knowledge acquired from different agents. It is also assumed that the environment in which these agents are situated is open and dynamic, so that agents can enter and leave the environment at any time, as well as have their beliefs constantly updated. This generates the need for agents to be able to query arbitrary agents about specific knowledge, even without knowing *a priori* whether the queried agents have such knowledge or not. It is also considered the possibility that an agent has knowledge related to its current focus – that is, knowledge related to its current situation, location, activity, object(s) of interest and/or objectives – that may be relevant so that other agents can reason correctly about a query sent by that agent. Thus, it is proposed that a query sent from one agent to another allows the former to share focus knowledge that is relevant in the scope of the query. Therefore, a rule-based distributed structured argumentation model that formalizes reasoning structures and argumentation semantics is proposed, as well as a distributed query processing algorithm based on this model. Alternative formulas for argument strength calculation based on different intuitions about possible mental attitudes of an agent and that take advantage of argumentation structures in different ways are also presented. This thesis also opens up possibilities for future work, such as in the scope of belief revision and generation of explanations based on the argumentation structures constructed by agents.

Keywords: argumentation; defeasible reasoning; multi-agent systems; distributed reasoning; open and dynamic environments.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

ABREVIATURAS

cr-literal	literal rotulado concreto
er-literal	literal rotulado esquemático
fr-literal	literal rotulado de foco
i.e.	isto é
ir-literal	literal rotulado instanciado
n.q.d.r.	no que diz respeito
r-literal	literal rotulado
sse	se e somente se
t.q.	tal que

SIGLAS

BC	base de conhecimento
BDI	crença-desejo-intenção, do Inglês, *belief-desire-intention*
DDMCS	Sistema Multicontexto Distribuído e Dinâmico, do Inglês, *Dynamic Distributed Multi-Context System*
DDRMAS	Sistema Multiagente baseado em Raciocínio Derrotável Distribuído, do Inglês, *Distributed Defeasible Reasoning-based Multi-Agent System*
DL	Lógica Derrotável, do Inglês, *Defeasible Logic*
EDP	Programação Lógica Disjuntiva Estendida, do Inglês, *Extended Disjunctive Programming*
IA	Inteligência Artificial
P2P	par-a-par, do Inglês, *peer-to-peer*
SMA	sistema multiagente
SMC	sistema multicontexto

ACRÔNIMOS

DeLP	Programação Lógica Derrotável, do Inglês, *Defeasible Logic Programming*

LISTA DE SÍMBOLOS

LETRAS LATINAS

∂	Etiqueta de literal derrotavelmente provável ou não derrotavelmente provável em DL, página 37
$\mathcal{DL}$	Teoria em DL, página 37
P_a	Função de confiança de um agente a, página 63
$\mathcal{S}$	Sistema DDRMAS, página 63
st	Limite de similaridade, página 63
$\top$	Tautologia, página 78

LETRAS GREGAS

Δ	Etiqueta de literal definitivamente provável ou não definitivamente provável em DL, página 37
Θ	Função de similaridade, página 63
θ	Grau de similaridade, página 75

NOTAÇÕES

$(p \leftharpoonup X,Y,...Z)$	Notação em linha de um argumento, onde p é um *cr-literal* e $X,Y,...Z$ são subargumentos , página 78
$(p \leftharpoonup \top)$	Notação em linha de um argumento base, onde p é um *cr-literal*, página 78
$\langle a, x \rangle$	Estrutura de *cr-literal*, sendo a um agente e x um literal, página 65
$\langle @, x \rangle$	Estrutura de *er-literal*, sendo x um literal, página 65
$\square$	Marcador de fim de exemplo, página 72
$\langle \mathcal{F}, x \rangle$	Estrutura de *fr-literal*, sendo x um literal, página 65
$\langle b, x, \theta \rangle$	Estrutura de um *ir-literal* referente a um *cr-literal* definido pelo agente b com grau de similaridade θ, página 75
$\triangle$	Marcador de fim de prova, página 177
$r_{ai} : p \Leftarrow q_1,...,q_n$	Estrutura de regra derrotável com corpo não vazio de tamanho n, página 64
$r_{ai} : p \leftarrow q_1,...,q_n$	Estrutura de regra estrita com corpo não vazio de tamanho n, página 64
$r_{ai} : p \hookleftarrow q_1,...,q_n$	Estrutura genérica de regra com corpo não vazio de tamanho n, onde p, $q_1,...,q_n$ são *r-literais*, página 64
$r_{ai} : p \hookleftarrow$	Estrutura genérica de regra com corpo vazio, onde p é um *r-literal*, página 64
$\langle \mathbb{D}, x \rangle$	Estrutura genérica de *r-literal*, sendo $\mathbb{D}$ o definidor e x um literal, página 65

ÍNDICES E CONJUNTOS

$a,b,... \in Ags$	Agentes e conjunto de agentes, página 63
$\mathbb{D} \in ArgD(A)$	Definidor de um argumento A, página 76

$A, B,... \in Args$	Argumentos e conjunto de argumentos, página 74
A_1^α,A_2^α,...A_n^α	Argumentos de um agente a em um foco de consulta α, página 78
$p \in Conc(A)$	Conclusão de um argumento A, página 76
p,$q \in V^{CR}$	Conjunto de *cr-literais*, página 65
$q \in DIRLs(A)$	*Ir-literais* diretos de um argumento A, página 77
$B \in DISA(A)$	Subargumentos instanciados diretos de um argumento A, página 76
p,$q \in V^{ER}$	Conjunto de *er-literais*, página 65
$Fall(A)$	Valor-verdade ($true$ ou $false$) que indica se um argumento A é falacioso ou não, página 77
α,β,... $\in F_Q$	Conjunto de focos de consulta, página 63
$q \in IRLs(A)$	*Ir-literais* de um argumento A, página 77
$B \in ISA(A)$	Subargumentos instanciados de um argumento A, página 76
J_i	Conjunto de argumentos justificados no estágio i de justificação, página 93
r,$s \in KB_a$	Base de conhecimento de um agente a, página 63
r,$s \in KB_{a\alpha}$	Base de conhecimento estendida local, composta das regras de um agente a e das regras do foco α, página 71
r,$s \in KB^d$	Conjunto de regras derrotáveis locais, página 65
r,$s \in KB^@$	Conjunto de regras derrotáveis esquemáticas, página 65
$r_{\alpha i}^{\mathcal{F}} \in KB_\alpha^{\mathcal{F}}$	Regra de foco e base de conhecimento de foco de um foco de consulta α, página 70
r,$s \in KB^m$	Conjunto de regras derrotáveis de mapeamento, página 65
r,$s \in KB^s$	Conjunto de regras estritas, página 65
r,$s \in KB_{\mathcal{S}\alpha}$	Base de conhecimento estendida global, composta das regras de todos os agentes do sistema $\mathcal{S}$ juntamente com as regras do foco α, página 71
x,$y \in V$	Conjunto de literais, página 65
$q \in Prem(A)$	Premissas de um argumento A, página 77
R_i	Conjunto de argumentos rejeitados no estágio i de rejeição, página 95
$\mathbb{D} \in D(p)$	Definidor de um *r-literal* p, página 65
p,$q \in V^R$	Conjunto de *r-literais*, página 65
$x \in L(p)$	Literal encapsulado por um *r-literal* p, página 65
$r \in Rule(A)$	Regra na qual se baseia um argumento A, página 76
$B \in SA(A)$	Subargumento de um argumento A, página 76
$StArg(A)$	Força de um argumento A, página 87
$StIRLs(p)$	Força de um *ir-literal* p, página 86

SUMÁRIO

1 INTRODUÇÃO

No campo da Inteligência Artificial, agentes inteligentes são projetados para atingirem seus objetivos desempenhando suas atividades de forma autônoma. Agentes também são capazes de adquirir conhecimento a partir do ambiente em que estão situados, e assim tirar novas conclusões com base em inferências sobre tal conhecimento. Tais conclusões podem, então, servir de insumo para que os agentes possam tomar decisões e realizar as ações necessárias para o alcance de seus objetivos. No entanto, considerando um sistema multiagente (SMA) no qual os agentes possuem uma visão limitada e possivelmente imperfeita do ambiente, é importante que tais agentes possam se comunicar, solicitando e compartilhando conhecimento de forma conjunta. Dessa forma, além dos agentes possuírem a capacidade individual de realizar raciocínio com base em seu próprio conhecimento, é importante que tenham capacidade de realizar raciocínio distribuído, de forma colaborativa, a fim de tirar proveito de conhecimentos de outros agentes, provenientes de fontes diversas. Além disso, supõe-se que os agentes estão situados em um ambiente aberto e dinâmico, no sentido de que agentes podem entrar e sair constantemente do ambiente e terem seu estado mental e atitudes alteradas a todo momento, o que traz consigo desafios relacionados à confiança entre agentes e à imperfeição de seus conhecimentos.

Este trabalho trata de representação de conhecimento e raciocínio distribuído em SMAs em que os agentes podem possuir conhecimento imperfeito acerca do ambiente em que estão situados. Há diversas definições de conhecimento ou informação imperfeita na literatura, mas este trabalho essencialmente aborda conhecimento que pode ser parcial, inconsistente e/ou incerto (PARSONS, 1996; HENRICKSEN; INDULSKA, 2004).

Conhecimento parcial diz respeito à possibilidade de os agentes individualmente não terem um conhecimento completo sobre o ambiente e, portanto, dependerem de conhecimentos de outros agentes para chegar a conclusões. Por exemplo, suponha um agente robô a que pode sentir o vento, mas é cego, e que tem o seguinte conhecimento na forma de uma regra: “Posso concluir que *vai chover* (conclusão) se estiver *ventando* (premissa 1) e *nublado* (premissa 2)”. O agente a não pode chegar à conclusão de que vai chover, porque não consegue encontrar o valor-verdade da premissa 2 por si mesmo. Supondo que haja no ambiente um agente b, então a pode perguntar a b se ele sabe se está nublado. Se b responder positivamente, então a pode concluir que vai chover.

Outros tipos de conhecimento imperfeito derivam de a possibilidade de agentes possuí-

rem conhecimentos conflitantes devido a capacidades de percepção imperfeitas ou, até mesmo, por agentes mal intencionados ou não confiáveis, o que pode levar a um estado de inconsistência global[1] do conhecimento. Por exemplo, um agente c possui um sensor visual danificado. Se a pergunta a c se está nublado, é possível que c não possa informar corretamente, retornando assim uma resposta que contradiz a resposta dada pelo agente b. Portanto, a teria que decidir em qual agente acreditar ao receber ambas as respostas. Tal decisão poderia ser baseada em um grau de confiança que um agente tem em relação a outro. Em ambientes abertos e dinâmicos, é comum que um agente não possua plena confiança em outro agente. Como será apresentado em detalhes mais adiante, uma abordagem sofisticada de realizar o raciocínio em bases de conhecimento (BCs) inconsistentes é o uso de *frameworks* de argumentação, especialmente os baseados em raciocínio derrotável (BENFERHAT *et al.*, 1993; AMGOUD; CAYROL, 2002; GOVERNATORI *et al.*, 2004; BIKAKIS; ANTONIOU, 2010; GARCÍA; SIMARI, 2014; MODGIL; PRAKKEN, 2014).

Outra preocupação em ambientes distribuídos, especialmente ambientes abertos, dinâmicos e *knowledge-intensive*, é buscar conhecimento de fontes arbitrárias. Algumas abordagens, especialmente aquelas baseadas em sistemas multicontexto (SMC), fornecem um formalismo para representar a troca de conhecimento entre os agentes por meio de regras de mapeamento (ou regras de ponte) (BREWKA *et al.*, 2007; BIKAKIS; ANTONIOU, 2010; DAO-TRAN *et al.*, 2010; BREWKA *et al.*, 2018). No entanto, a maioria desses trabalhos não consegue lidar com ambientes dinâmicos e abertos, ou seja, quando as fontes de conhecimento (os agentes) e seus conteúdos podem mudar com o tempo e, portanto, nem sempre podem ser conhecidas *a priori*. Por exemplo, suponha que d e e entraram no ambiente recentemente, de forma que a não sabe que tipo de conhecimento eles possuem. Suponha novamente que a deseja saber se vai chover, mas os agentes b e c, que antes costumavam responder se está nublado ou não, saíram do sistema. Então, a única maneira de a saber se vai chover é perguntando aos agentes d e e, mesmo sem saber se possuem tal conhecimento ou não.

Além disso, existe a possibilidade de os agentes precisarem raciocinar com base em diferentes graus de certeza sobre a equivalência de conhecimentos provenientes de diferentes fontes com vocabulários heterogêneos. Portanto, este trabalho lida com a incerteza no sentido de ser possível um raciocínio dependente de um grau de similaridade derivado da correspondência

[1] Utiliza-se o termo "global" para se referir a um estado do sistema constituído pela soma de seus componentes. Assim, base de conhecimento global refere-se ao conjunto de conhecimentos resultante da união das bases de conhecimento de todos os agentes.

entre os conhecimentos utilizadas, como ocorre na área de correspondência de ontologias (CROSS, 2003; EUZENAT *et al.*, 2007), embora este trabalho não trate especificamente de ontologias. Suponha que d responda que há *nuvens no céu* enquanto e responde que está *nebuloso*. Considerando que a premissa *"nublado"* da regra de a tem uma similaridade maior com *"nebuloso"* (um céu coberto de nuvens) do que simplesmente *"nuvens no céu"*, a poderia dar preferência à resposta de e. Se apenas d respondesse, haveria um grau de incerteza maior pairando sobre a conclusão tomada, visto que o conhecimento vindo de d não possui exatamente o mesmo significado que o existente na regra de a. Uma intuição similar, embora não especificamente baseada na similaridade entre conhecimentos, é apresentada por Benferhat *et al.* (1993) e Amgoud e Cayrol (2002), que propõem a construção de argumentos a partir de crenças com prioridades explícitas, tais como níveis de certeza. Argumentos que usam mais crenças para as quais há maior certeza são mais fortes do que argumentos com crenças com menor certeza.

Finalmente, um recurso importante em ambientes distribuídos é permitir que os agentes compartilhem conhecimentos relevantes relacionados a sua situação atual, localização, atividade, objeto(s) de interesse ou objetivo ao consultar outros agentes, que podem não estar cientes desses conhecimentos. Neste trabalho, esse tipo de conhecimento é chamado de *foco*. Por exemplo, suponha um agente f em uma cidade distante de onde a está. Ele olha para o céu e vê que está nublado, e também é capaz de sentir o vento, mas não sabe se isso significa que vai chover, por não possuir a mesma regra que a. O único agente com quem ele pode falar no momento é a, então ele pergunta a a se vai chover. No entanto, a, localizado em outra cidade, não tem conhecimento das condições climáticas da localização de f. Logo, f deve enviar, junto com sua consulta a a, algum conhecimento que possa ajudar a a raciocinar corretamente. Assim, f envia a consulta informando que há nuvens no céu e que está ventando no local em que se encontra. Quando a recebe a consulta, ele é capaz de concluir que vai chover na localização de f e responder corretamente.

Esta tese apresenta uma abordagem distribuída de raciocínio que busca modelar as características apresentadas.

A Seção 1.1 apresenta em mais detalhes os tipos de aplicação que podem se beneficiar da abordagem apresentada nesta tese, assim como um cenário de uso. A Seção 1.2 desenvolve, em mais detalhes, os pressupostos e desafios identificados com base no cenário descrito, e define a questão de pesquisa. A Seção 1.3 apresenta os objetivos da tese. A Seção 1.4 apresenta resumidamente as contribuições da tese, isto é, as soluções que estão sendo entregues à comunidade

científica para os problemas propostos. Enfim, a Seção 1.5 apresenta a estrutura da tese.

1.1 MOTIVAÇÃO

O raciocínio distribuído é um aspecto central exigido por muitas aplicações propostas nas últimas décadas, incluindo a Web Semântica (BERNERS-LEE *et al.*, 2001; ANTONIOU *et al.*, 2012), sistemas móveis e ubíquos, espaços inteligentes e sistemas de Inteligência Ambiente (BIKAKIS; ANTONIOU, 2010; RAKIB; UDDIN, 2019; MEKURIA *et al.*, 2019), e mais recentemente a Internet das Coisas (do Inglês, *Internet of Things*) (MAARALA *et al.*, 2017; SU *et al.*, 2018). Nesses tipos de sistemas, entidades computacionais distribuídas, geralmente chamadas de agentes inteligentes, são capazes de capturar ou receber conhecimento de seu ambiente e de outros agentes, e muitas vezes são capazes de derivar novos conhecimentos ou tomar decisões com base no conhecimento disponível com o objetivo de auxiliar ou fornecer informações úteis para usuários humanos. Tais usuários podem estar munidos de dispositivos móveis ou vestíveis, ou mesmo estar em um ambiente com dispositivos controlados por tais agentes inteligentes. A necessidade de comunicação e colaboração com outros agentes, possivelmente em localizações geograficamente distantes, pode surgir da necessidade de adquirir conhecimento que não pode ser obtido pelas capacidades sensoriais do dispositivo.

Outro tipo de aplicação que pode se beneficiar do raciocínio distribuído são os jogos sérios colaborativos (LORETO *et al.*, 2012; ANDREOLI *et al.*, 2017). Dentre eles, pode-se citar os jogos do estilo *Massive Multiplayer Online* (MMO) de cunho educativo (STEINKUEHLER, 2008), nos quais os aprendizes, por meio de avatares, se encontram imersos em um mundo virtual semelhante ao mundo real, podendo interagir com outros aprendizes e com recursos do ambiente a fim de alcançar seus objetivos – e.g., no caso de jogos educativos, o aprendizado de conceitos ou habilidades. Há também aplicações que mesclam o ambiente físico e virtual, como é o caso dos Jogos Baseados em Localização (do Inglês, *Location-Based Games*) (ERENLI, 2013), que muitas vezes incorporam a tecnologia de Realidade Aumentada para criar um ambiente imersivo que agrega conteúdos virtuais a lugares ou coisas no mundo real. Em tais jogos, usuários em diferentes localizações geográficas devem cumprir determinados objetivos, e, para isso, pode ser necessário que compartilhem informações entre si de maneira colaborativa.

A fim de ilustrar uma aplicação, que pode ser imaginada tanto em um contexto real como de um jogo, será apresentado um cenário em que coletores de cogumelos em um bosque são auxiliados por agentes inteligentes sendo executados em seus respectivos dispositivos móveis

para coletar cogumelos comestíveis. Esse mesmo cenário, que é uma atividade do tipo "caça ao tesouro", pode ser reimaginado em diferentes contextos em que se propõe a coleta de itens ou a verificação de situações ou objetos espalhados por um ambiente. Por exemplo, o mesmo tipo de cenário poderia ser imaginado em um contexto de resgate de pessoas em um desastre, o que envolve busca de vestígios e pessoas e a decisão sobre o que fazer em cada situação identificada (por exemplo, solicitar retroescavadeira de resgate quando for encontrado vestígio de local de soterramento de pessoas, ou solicitar helicóptero com escada para resgate de pessoas em local de difícil acesso por terra). O mesmo tipo de cenário também poderia ser imaginado no contexto de um jogo educacional, em que os aprendizes têm por objetivo coletar ou tirar foto de itens de um determinado tipo (por exemplo, cogumelos comestíveis) em um ambiente aberto, e assim construir um aprendizado sobre tais itens.

1.1.1 Cenário: Coletores de Cogumelos

Este cenário, baseado no proposto por Bikakis e Antoniou (2009), simula um ambiente de coletores de cogumelos.

O objetivo dessa atividade é coletar o máximo possível de cogumelos comestíveis e evitar cogumelos venenosos.

Cada pessoa possui um dispositivo móvel executando um *software* que registra exatamente o conhecimento que a pessoa tem sobre cogumelos. Tal conhecimento é então passado a um agente inteligente (ou agente do usuário) executando no mesmo dispositivo móvel, que realiza raciocínio sobre esse conhecimento e, se necessário, realiza consulta a outros agentes registrados na rede, de modo a auxiliar seu usuário com a melhor sugestão possível quanto à decisão de coletar ou não um cogumelo. Portanto, presume-se que os agentes do usuário estejam vinculados a uma mesma plataforma de agentes que provê um *message broker* que capacita os agentes a se comunicarem uns com os outros por meio da uma rede, um serviço de diretório e descoberta que permite aos agentes visualizarem outros agentes disponíveis (BELLIFEMINE *et al.*, 1999).

Supõe-se um cenário em que há 5 (cinco) coletores de cogumelos com seus respectivos agentes: Alice (agente a), Barb (agente b), Charles (agente c), Dennis (agente d) e Eric (agente e). A Figura 1 apresenta uma ilustração do cenário. A modelagem do conhecimento será apresentada no Capítulo 3, quando a formalização do *framework* proposto for apresentada. Portanto, o que segue é apenas uma descrição textual ilustrativa do cenário e das possíveis conclusões que podem

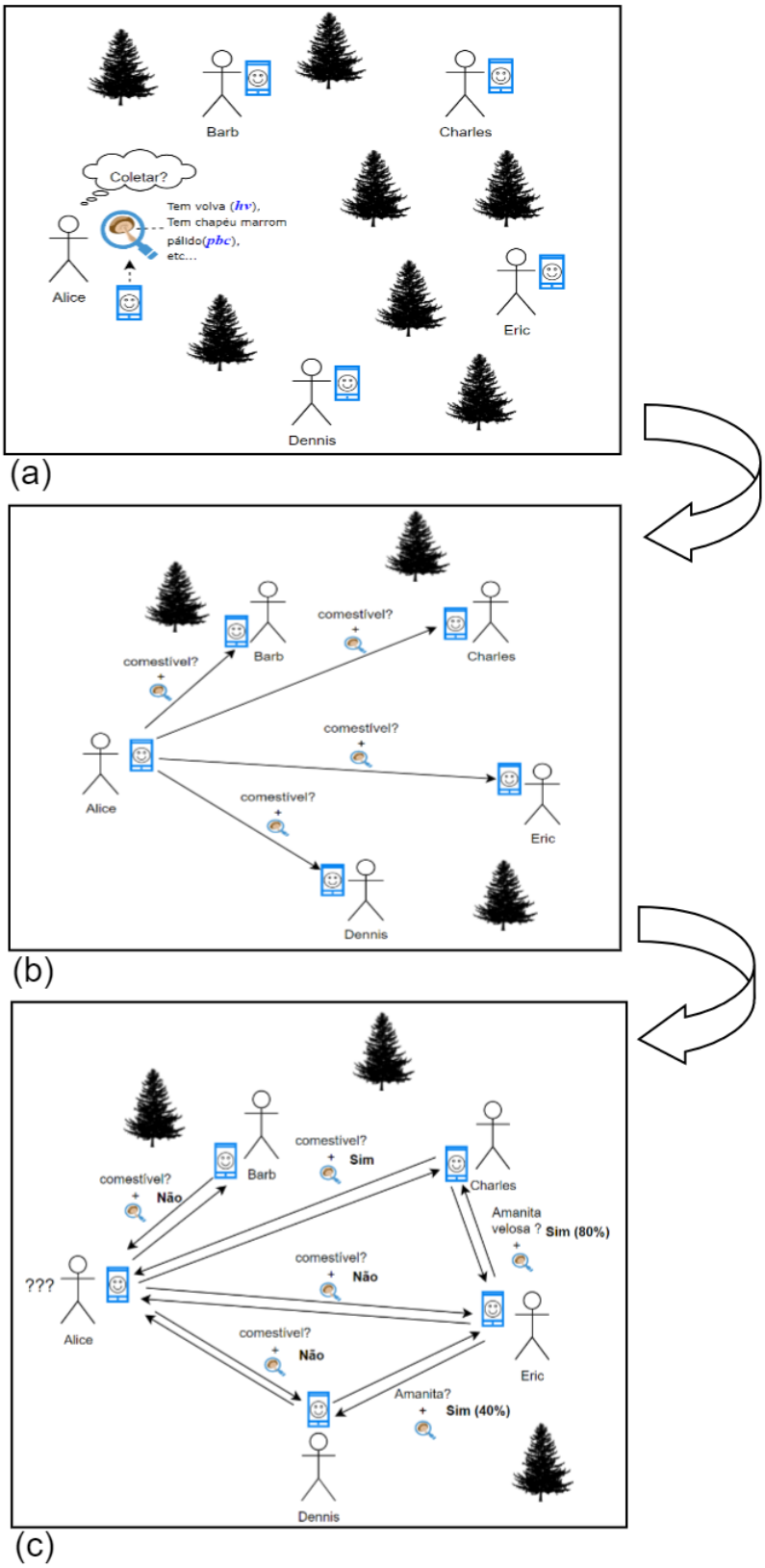
ser tiradas pelo agente de Alice.

Supõe-se também que os agentes possuem diferentes graus de confiança uns nos outros. Tal grau de confiança poderia ser derivado de algum mecanismo interno de aprendizagem do agente (por exemplo, um agente passa a confiar mais em um outro agente ao interagir com ele e obter conclusões que levam a decisões com resultados favoráveis, de acordo com alguma função de utilidade do agente) e/ou de fatores sociais (por exemplo, um agente ser considerado um especialista, ou com base na quantidade de tempo que um agente está no sistema, ou pela quantidade de participações do agente no alcance de conclusões), ou até mesmo uma confiança introjetada pelo próprio usuário. Salienta-se que os mecanismos de atribuição e atualização de graus de confiança estão fora do escopo deste trabalho, portanto considera-se apenas o estado dos graus de confiança em um dado momento. No caso deste cenário, considera-se que o agente de Alice (a) possui maior confiança no agente do Eric (e), seguido, em ordem, pelo agente de Charles (c), de Barb (b) e de Dennis (d). Barb possui confiança equivalente em todos os agentes. Charles possui maior confiança em Alice e Eric e menos confiança em Barb e Dennis. Dennis possui maior confiança em Charles e Eric e menos confiança em Alice e Barb. Finalmente, Eric possui maior confiança em Charles, seguido, em ordem, por Alice, Barb e Dennis. Uma representação numérica desses diferentes graus de confiança será apresentada na Seção 3.1.

Em um determinado momento, Alice encontra um cogumelo (denotado m_1) que possui as seguintes características: uma haste cuja parte de baixo é coberta por um saco proeminente, chamado de *volva*, um chapéu marrom pálido com manchas, enquanto que as margens do chapéu são proeminentemente alinhadas, e o cogumelo não tem um anel (*annulus*). Essas características são todas registradas no *software* específico de seu dispositivo móvel, que as codifica de modo a ser processado pelo agente.

Alice possui conhecimento sobre algumas espécies de cogumelo, tal como o *death cap*, que ela sabe ser venenoso. No entanto, a descrição do cogumelo que ela encontrou não se enquadra na descrição de nenhuma espécie conhecida por ela. Portanto, ela não é capaz, sozinha, de determinar se o cogumelo é ou não venenoso. O agente a de Alice, em antecipação à decisão de Alice, solicita ajuda a outros coletores de cogumelo no jogo. A consulta inclui a pergunta: "Este cogumelo m_1 é comestível?" e os fatos percebidos sobre o cogumelo encontrado – o foco do assunto. Isto é, além de enviar uma questão a ser respondida pelos demais agentes, é necessário que a também envie conhecimento relevante para que os demais agentes possam ajudá-lo a responder a questão. Neste caso, ele consegue falar com outros quatro agentes disponíveis.

Figura 1 – Ilustração do cenário dos coletores de cogumelo.



Legenda: (a) Alice encontra o cogumelo; (b) Alice envia a consulta aos demais agentes a fim de descobrir se o cogumelo é comestível ou não, juntamente com o conhecimento de foco referente às características do cogumelo encontrado; e (c) Charles e Dennis realizam consultas subsequentes sobre o cogumelo ser uma *Amanita velosa* ou uma *Amanita*, respectivamente, e os agentes enviam suas respostas a Alice.

Fonte: Autoria própria.

O primeiro é o agente de Barb (b), que acredita, de modo genérico e com base no seu conhecimento atual, que cogumelos com volva não são comestíveis. O segundo é o agente da Charles (c), que conhece nomes de espécies que não são tóxicas, dentre elas a *amanita velosa*, mas que não conhece as características distintas dessa espécie. O terceiro, d, de Dennis, acredita genericamente que amanitas são normalmente perigosas. Por fim, o agente e, de Eric, sabe descrever um cogumelo cujo nome é *amanita da primavera*, cuja descrição se enquadra perfeitamente na descrição do cogumelo em questão.

Supõe-se também que os agentes possuem uma capacidade de considerar a similaridade entre diferentes conhecimentos. Por exemplo, *amanita velosa* e *amanita da primavera* são dois nomes distintos para a mesma espécie de cogumelo, que é um cogumelo comestível do tipo *amanita*. Este, por sua vez, abrange várias espécies de cogumelo que, em geral, são venenosos. No entanto, os agentes não possuem certeza absoluta acerca da equivalência entre *amanita velosa* e *amanita da primavera*, de modo que possuem apenas 80% de certeza. De modo similar, os agentes possuem 40% de certeza acerca da equivalência entre *amanita* e *amanita velosa* e entre *amanita* e *amanita da primavera*, meramente por similaridade sintática. Pressupõe-se que tais graus de similaridade são provenientes de algum mecanismo cuja definição está fora do escopo deste trabalho. Ressalta-se que todos os agentes, por simplicidade, possuem os mesmos graus de certeza acerca dessas similaridades, mas que isso não necessariamente é um pré-requisito da abordagem proposta na tese, podendo os graus de certeza serem individuais para cada agente.

Neste cenário, Alice terá três opções: (1) usando como argumento o conhecimento de Barb, ela chegará à conclusão de que o cogumelo é venenoso e, portanto, o descartará; (2) usando como argumento o conhecimento de Dennis combinado ao conhecimento de Eric, ela chegará à mesma decisão; e (3) usando como argumento o conhecimento combinado de Charles e Eric, ela chegará à decisão contrária de coletar o cogumelo. A escolha de qual (ou quais) argumento(s) e conclusão adotar pode ser derivado de vários fatores, de acordo com uma estratégia de resolução de conflitos, que pode variar dependendo da aplicação. A estratégia padrão proposta neste trabalho considera tanto o grau de confiança entre os agentes quanto os graus de similaridade entre os conhecimentos envolvidos, os quais são agregados juntamente a estruturas baseadas em argumentos e, então, calculadas como sendo a força de tais argumentos. No caso deste cenário, pela estratégia padrão, Alice decidiria por acatar ao argumento baseado no conhecimento de Charles e Eric, levando-a a coletar o cogumelo, conforme a semântica descrita em detalhes na Seção 3.3 do Capítulo 3. Intuitivamente, tal decisão é tomada dado que: (1) Alice possui

maior confiança em Charles e Eric do que em Barb e Dennis; (2) Embora o argumento baseado no conhecimento de Dennis inclua conhecimento proveniente de Eric (o mais confiável), tal argumento tem força reduzida dado que o conhecimento de Dennis afirma que *amanitas* são venenosas, sendo que *amanita* possui baixo grau de similaridade com *amanita da primavera*, resultando em maior incerteza quanto à utilidade de tal conhecimento. Considera-se também que, embora haja uma confiança alta entre Alice e Eric, o fato do argumento gerado a partir de Charles depender indiretamente do conhecimento de Eric gera um tipo de confiança indireta entre Alice e Eric no que diz respeito a esse argumento. Então, embora o argumento de Charles ainda seja preferido em detrimentos dos demais, o fato de ser gerado a partir de uma dependência indireta deveria diminuir sua força, fazendo com que Alice fique em razoável dúvida entre acatar ao argumento de Barb, que depende só de seu próprio conhecimento, e de Charles. A intuição para isso é análoga à brincadeira do "telefone sem fio", na qual os participantes em uma sequência se comunicam transmitindo uma mensagem inicial de um para o outro, de modo que, quando a mensagem chega ao participante no final da linha, geralmente chega deturpada.

É importante notar que as conclusões derivadas do processo de raciocínio são baseadas no conhecimento dos agentes no momento em que cada agente recebe e processa a consulta, e no conhecimento de foco compartilhado pelo agente que emite a consulta. Se Eric não estivesse no sistema no momento da consulta, Alice acabaria decidindo descartar o cogumelo com base na crença de Barb. Como os agentes podem entrar e sair do ambiente, assim como ter suas crenças atualizadas, caso Alice precise realizar uma nova consulta aos demais agentes, ela terá que enviar mensagens aos agentes conhecidos disponíveis, que podem variar de momento a momento.

Em outras palavras, um agente pode não conhecer, *a priori*, qual outro agente disponível possui determinado conhecimento.

Caso seja pretendido, como requisito secundário derivado do raciocínio dos agentes, que os agentes aprendam ou revisem suas crenças com base nos argumentos que recebem, o agente de Alice (a), percebendo a confiabilidade do argumento gerado a partir do conhecimento de Charles (c) e Eric (e), poderia incorporar, em sua base de conhecimentos, novas regras envolvendo cogumelos do tipo *amanita da primavera* e suas características. Outro requisito secundário, que pode ser muito útil em ambientes educativos e também para que especialistas humanos possam auditar e ajustar o sistema, é a capacidade de extrair explicações dos agentes quanto aos motivos que os levaram a determinadas conclusões. Ambos os requisitos secundários, embora não sejam diretamente tratados neste trabalho, podem ser alcançados em parte por

uma estruturação adequada dos argumentos construídos pelos agentes durante o processo de raciocínio, uma vez que tais estruturas podem ser usadas como insumo para esses mecanismos.

1.2 PROBLEMAS

O cenário apresentado na Seção 1.1.1 ilustra uma configuração conceitual com os seguintes pressupostos:

- Existe um meio de comunicação disponível por meio do qual um agente pode se comunicar e trocar conhecimentos com outros agentes disponíveis.
- O ambiente é dinâmico e aberto, no sentido de que agentes entram e saem do sistema a todo o momento e podem ter seu conhecimento continuamente atualizado. Além disso, supõe-se que os agentes, assim que entram no sistema, são capazes de se comunicar com qualquer outro que também esteja no sistema.
- Não existe uma entidade central que fornece e executa as capacidades de raciocínio no sistema.
- Os agentes possuem algum mecanismo que permite a atribuição e atualização de graus de confiança entre si.
- Os agentes possuem algum mecanismo que permite atribuir graus de similaridade entre conhecimentos.

Os desafios de raciocínio no cenário descrito incluem:

- O conhecimento local de um agente pode ser *parcial*, o que significa que os agentes podem não possuir acesso a todo o conhecimento disponível no ambiente. Portanto, um agente deve ser capaz de consultar outros agentes em busca de *conhecimento faltante* para derivar conclusões. Por exemplo, é necessário que o agente de Alice consulte outros agentes a fim de saber se o cogumelo é ou não comestível.
- O conhecimento de diferentes agentes pode estar *inter-relacionado*, isto é, pode ser definido explicitamente que o conhecimento faltante deve ser adquirido por meio da interação com outros agentes, o que é chamado de *mapeamentos*. No entanto, pode não ser possível definir os *mapeamentos* com agentes específicos *a priori*, visto que o ambiente é aberto e

dinâmico. Logo, um agente deve ser capaz de representar e raciocinar sobre conhecimento faltante que pode ser adquirido de *agentes arbitrários*. Por exemplo, o agente de Alice não sabe exatamente quais agentes podem responder se o cogumelo é ou não comestível, portanto, deve consultar a todos.

- Ao receber uma consulta, um agente deve ser capaz de tornar-se *ciente* do conhecimento de *foco* da consulta do agente emissor, uma vez que tal conhecimento pode ser importante para produzir uma conclusão razoável e pode não ser conhecido *a priori* por todos os agentes. Por exemplo, é necessário que o agente de Alice envie as características do cogumelo aos agentes consultados, uma vez que os demais agentes não possuem qualquer conhecimento prévio sobre tal cogumelo.

- O conhecimento pode ser *inconsistente*, isto é, há casos em que um agente é capaz de gerar *conclusões mutuamente inconsistentes* a partir de conhecimentos adquiridos de dois ou mais agentes. Portanto, um agente deve ser capaz de escolher uma conclusão entre conclusões divergentes. Esta escolha deve ser realizada por meio de algum critério baseado nos *argumentos* que levam às conclusões divergentes. Considera-se que tal critério é baseado na *força desses argumentos*. Por exemplo, o agente de Alice recebe respostas contraditórias acerca do cogumelo ser comestível ou não para que possa tomar a decisão de coletar ou não o cogumelo. A conclusão que ela tomará deve ser baseada na força dos argumentos a favor e contra tal decisão.

- Os agentes podem ter diferentes *graus de confiança* uns nos outros, que devem ser considerados no cálculo da força dos argumentos baseados nos conhecimentos provenientes uns dos outros. Por exemplo, Alice tem maior tendência a aceitar um argumento construído por Charles do que um construído por Barb, visto que confia mais em Charles. Porém, é interessante também considerar um tratamento diferenciado quando o conhecimento utilizado é decorrente de uma confiança indireta entre os agentes. Por exemplo, mesmo que Alice confie mais em Charles, o argumento deste é baseado também no conhecimento de Eric, havendo assim uma dependência indireta entre Alice e Eric no que diz respeito ao argumento de Charles, o que poderia levar o argumento a não ser tão forte quanto seria se fosse baseado no conhecimento de um único agente.

- O conhecimento pode ser *incerto*, no sentido de que pode existir uma incerteza sobre a correspondência de conhecimentos recebidos de diferentes agentes. Portanto, o agente

deve ser capaz de considerar conhecimentos que não são exatamente iguais, no sentido de terem vocabulários heterogêneos, mas que possuem algum grau de *similaridade* entre eles. Tal similaridade deve também compor o cálculo de força dos argumentos que incluem tais conhecimentos, uma vez que argumentos que utilizam conhecimentos incertos deveriam ter força menor do que argumento baseados em conhecimentos com maior certeza. Por exemplo, existe uma incerteza acerca da equivalência entre amanitas e amanitas da primavera, o que faz com que Dennis construa um argumento baseado em uma incerteza maior.

- Os agentes devem gerar *argumentos estruturados* durante o processo de raciocínio, de modo que possam ser persistidos e posteriormente reutilizados como insumo para outras funcionalidades além do raciocínio dedutivo, tais como na revisão de crenças e na geração de explicações.

Com base nesses pressupostos e problemas, esta tese visa responder à seguinte questão de pesquisa:

> *"Como conceber um modelo geral para esse tipo de cenário que permita realizar o raciocínio de forma totalmente distribuída?"*

Conforme apresentado, em mais detalhes, no Capítulo 5, existem trabalhos que resolvem apenas alguns aspectos do problema. Um deles é a Lógica Derrotável Contextual (CDL, do Inglês *Contextual Defeasible Logic*) de Bikakis e Antoniou (2010), que propõe resolução de conflitos em um *framework* de argumentação baseado em Lógica Derrotável, cujos fundamentos são apresentados no Capítulo 2. No entanto, eles consideram apenas ambientes estáticos, nos quais cada agente conhece exatamente qual outro agente possui o conhecimento requerido. O cenário apresentado na Seção 1.1.1, por exemplo, não é possível de ser representado em CDL. Além disso, a solução operacional proposta por eles não gera argumentos estruturados durante o processo de raciocínio, que poderiam ser reutilizados em funcionalidades futuras. O trabalho de Dao-Tran *et al.* (2010), por outro lado, tem por objetivo permitir o raciocínio contextual distribuído em ambientes dinâmicos. No entanto, não é baseado em argumentação e não propõe formas de resolução de conflitos necessárias quando há conclusões contraditórias. García e Simari (2014), em seu trabalho sobre Programação Lógica Derrotável (DeLP, do Inglês *Defeasible Logic Programming*), apresentam um *framework* baseado em raciocínio derrotável que permite a

resolução de conflitos com conhecimento imperfeito, e que propõe uma ideia parecida com o compartilhamento de foco em consultas, as quais denominam *consultas contextuais*. No entanto, tal trabalho não apresenta uma abordagem totalmente distribuída e não provê a formalização de conhecimento inter-relacionado entre diferentes bases de conhecimento. Uma comparação mais detalhada desses e de outros trabalhos correlatos é apresentada no Capítulo 5.

1.3 OBJETIVOS

O objetivo principal deste trabalho é elaborar e avaliar um modelo de raciocínio distribuído com agentes com diferentes graus de confiança entre si, que leve em consideração a possibilidade de os agentes possuírem conhecimento imperfeito, isto é, conhecimento parcial, inconsistente e/ou incerto, e que permita aos agentes realizarem consultas compartilhando conhecimento relacionado ao seu foco. Uma vez que o ambiente é aberto e dinâmico, é necessário também que os agentes sejam capazes de realizar consultas a agentes arbitrários, por não saberem qual agente especificamente pode possuir determinado conhecimento.

O raciocínio deve também permitir a geração de estruturas baseadas em argumentos que sirvam de insumo, tanto para o raciocínio em si, quanto para preparar caminho para possíveis funcionalidades futuras relacionadas à revisão de crenças dos agentes e à geração de explicações.

Para isto, os seguintes objetivos secundários foram estabelecidos:

- Realizar uma revisão da literatura acerca de abordagens e conceitos existentes que atendam aos problemas levantados.
- Elaborar e avaliar a formalização do modelo, incluindo a arquitetura do sistema, a linguagem de representação de conhecimento e a estrutura e semântica de argumentação.
- Elaborar e avaliar um algoritmo distribuído que realize o modelo proposto e a construção de argumentos estruturados durante o processo de raciocínio.
- Realizar uma comparação do modelo proposto com abordagens existentes, propondo também desafios a serem tratados em trabalhos futuros.

1.4 CONTRIBUIÇÕES DA TESE

Com base nos problemas e requisitos apresentados, esta tese apresenta as seguintes contribuições:

- Abordagem totalmente distribuída de raciocínio baseada em argumentação com agentes com bases de conhecimento inter-relacionadas.
- Representação de conhecimento distribuído na forma de regras derrotáveis de mapeamento esquemáticas, que permite aos agentes adquirirem conhecimentos de fontes arbitrárias.
- Modelagem formal de um *framework* de argumentação estruturada que atende a requisitos relacionados à distribuição, imperfeição e heterogeneidade do conhecimento, correspondência de conhecimentos com base na similaridade, dinamicidade e abertura do ambiente, e confiança imperfeita entre os agentes, conforme apresentados na Seção 1.2.
- Definição de semântica de argumentação baseada em um cálculo de força de argumentos que leva em consideração o grau de confiança entre agentes e a similaridade entre os conhecimentos utilizados.
- Métodos alternativos de cálculo de força de argumentos que levam em consideração diferentes intuições acerca de atitudes dos agentes.
- Algoritmo distribuído de processamento de consultas baseado na modelagem formal proposta, que prevê os pontos em que deve ser executado com paralelismo, e que é capaz de construir estruturas de argumentos que, além de serem utilizados no raciocínio, também poderiam ser reaproveitadas para futuras finalidades, tais como revisão de crenças e geração de explicações.

1.5 ESTRUTURA DO DOCUMENTO

Capítulo 2: Fundamentação Teórica: apresenta os fundamentos teóricos sobre os quais se baseia o modelo proposto.

Capítulo 3: Modelo de Representação e Semântica de Argumentação: apresenta o modelo proposto, incluindo uma formalização da representação e da semântica de raciocínio baseada em argumentação, assim como a formalização do cenário apresentado no Capítulo 1.

Capítulo 4: Algoritmo Distribuído para Resposta a Consultas: apresenta o algoritmo de raciocínio distribuído, assim como análises das propriedades do algoritmo.

Capítulo 5: Trabalhos Relacionados: apresenta trabalhos relacionados e um estudo comparativo em relação ao modelo proposto nesta tese.

Conclusões e Trabalhos Futuros: são apresentadas as conclusões do trabalho, assim como discussão sobre trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados conceitos da literatura que servem de base para a solução do problema apresentado no Capítulo 1. Visto que este trabalho tem como ponto de partida a área de representação de conhecimento e raciocínio, na Seção 2.1 são apresentados alguns conceitos relacionados a essa área de pesquisa, juntamente com uma breve apresentação da lógica clássica e uma discussão do porquê de ela não ser suficiente para a representação e raciocínio neste trabalho. A Seção 2.2 apresenta então as abordagens de raciocínio derrotável e argumentação como uma forma de resolver o problema de lidar com conclusões conflitantes em bases de conhecimento inconsistentes. A Seção 2.3 apresenta conceitos relacionados a sistemas baseados em conhecimento distribuído, sendo discutidos: os sistemas multiagente (Seção 2.3.1); o conceito de raciocínio contextual (Seção 2.3.2); e os sistemas multicontexto (Seção 2.3.3). A Seção 2.4 apresenta conceitos relacionados à confiança em sistemas de raciocínio distribuídos, em especial em SMAs. A Seção 2.5 apresenta brevemente os conceitos relacionados à correspondência de conhecimentos ou informações heterogêneas por similaridade. A Seção 2.6 apresenta conceitos da literatura relacionados à ideia de foco e compartilhamento de foco em SMAs. Por fim, uma breve discussão na Seção 2.7 resume os conceitos apresentados tendo em vista o problema proposto neste trabalho.

2.1 REPRESENTAÇÃO DE CONHECIMENTO

Brachman e Levesque (2004) definem *representação de conhecimento* como "o campo de estudo relacionado ao uso de símbolos formais para representar uma coleção de proposições cridas por algum agente putativo". Um dos tipos mais comuns de raciocínio utilizado em Inteligência Artificial (IA) é a *inferência lógica*, na qual uma sentença final representa uma conclusão lógica das proposições representadas pelas sentenças iniciais, as premissas.

Existem muitas técnicas de representação do conhecimento. Este trabalho se preocupa especialmente com as abordagens baseadas em lógica. De acordo com Brachman e Levesque (2004), a lógica fornece uma semântica precisa e possibilita o raciocínio (inferência de novo conhecimento a partir do conhecimento existente). A lógica utilizada em representação de conhecimento e raciocínio possui suas origens e fundamentos na lógica matemática, também conhecida como lógica clássica. A linguagem da lógica clássica mais amplamente usada em

representação do conhecimento é a linguagem das fórmulas de primeira ordem (ou de predicados). Esta foi a representação usada por John McCarthy visando representar o conhecimento declarativo em seu artigo Advice Taker (MCCARTHY, 1959), e que Alan Robinson propôs realizar a prova automática usando resolução (ROBINSON, 1965). A lógica proposicional é um subconjunto da lógica de primeira ordem, e é particularmente interessante do ponto de vista computacional por existirem algoritmos de satisfatibilidade eficientes para essa lógica.

No entanto, a lógica clássica possui algumas limitações, especialmente no que diz respeito à representação de conhecimento imperfeito. Um dos pressupostos da lógica clássica é a monotonicidade, e que impossibilita seu uso em cenários nos quais existe a possibilidade de que conclusões antes consideradas válidas devam ser retraídas, ou revogadas, ante a adição de novas informações. A lógica clássica é monotônica no seguinte sentido: sempre que uma sentença A é uma consequência lógica de um conjunto de sentenças T, então A também é uma consequência de um superconjunto arbitrário de T. Em outras palavras, adicionar informações nunca invalida nenhuma conclusão já existente (BREWKA *et al.*, 2008). Esse problema é resolvido por meio de formalismos de lógicas não-monotônicas, como é o caso dos formalismos de raciocínio derrotável apresentados na Seção 2.2.

Um exemplo de como a lógica clássica é incapaz de modelar certas situações é dada a seguir. Suponha que se queira representar conhecimento sobre aves e suas características em uma base de conhecimento KB que utiliza lógica clássica. Em determinado momento, o sistema é alimentado com o conhecimento de que toda ave voa e acerca de uma ave chamada "tweety" por meio das seguintes regras:

$$r_1 : voa(P) \leftarrow ave(P)$$

$$r_2 : ave(tweety) \leftarrow$$

Neste trabalho, convenciona-se chamar os elementos de uma regra de *antecedentes*, ou *corpo* da regra, que é um conjunto de fórmulas ou literais que indicam as condições para que a regra possa ser utilizada para derivar seu *consequente*, ou *cabeça* da regra. Portanto, neste caso, o conjunto de antecedentes de r_1 é composto por um único elemento $ave(P)$ e o consequente é $voa(P)$, enquanto o conjunto de antecedentes de r_2 é vazio e o consequente é $ave(tweety)$, de forma que $ave(tweety)$ sempre será derivada. Um mecanismo de raciocínio automático, por inferência sobre essas regras, pode chegar à conclusão de que "tweety" voa, representado pela fórmula fechada $voa(tweety)$. Diz-se que $voa(tweety)$ é uma consequência lógica de KB,

denotado $KB \models voa(tweety)$. Agora suponha que, em determinado momento, seja introduzida a regra de que pinguins não voam, e de que *tweety* é, na realidade, um pinguim, de modo que as seguintes regras são adicionadas:

$$r_3 : \neg voa(P) \leftarrow pinguim(P)$$

$$r_4 : pinguim(tweety) \leftarrow$$

Onde $\neg$ denota a negação da informação, de modo que $\neg voa(P)$ é um predicado que indica que "P não voa", onde P é uma variável livre que deve ser substituída por uma constante (no caso, *tweety*). Neste caso, obtém-se uma BC inconsistente, pois o mesmo mecanismo de raciocínio irá utilizar as regras r_3 e r_4 para chegar à conclusão de que "tweety" não voa, isto é, $KB \models \neg voa(tweety)$, o que contradiz a consequência lógica anterior. A fim de que não haja tal inconsistência nas consequências lógicas, é necessária alguma forma de suprimir uma das conclusões. Isso pode ser feito por meio de uma relação de superioridade de uma regra sobre a outra, de modo que, quando duas regras (ou cadeias de regras, que também podem ser chamadas de argumentos) podem ser utilizadas para apoiar conclusões contraditórias, apenas uma delas é aceita. Neste exemplo, se for considerado que r_3 é uma regra mais forte que r_1, apenas a conclusão $\neg voa(tweety)$ será aceita, e $voa(tweety)$ será rejeitada. Dessa forma, o raciocínio torna-se não-monotônico, pois uma informação que antes era uma consequência lógica do sistema pode ser retraída ao se adicionar novas informações.

Voltando ao cenário motivador apresentado na Seção 1.1.1, nota-se que ocorre algo similar: o agente de Alice (a) deve ser capaz de decidir entre conclusões conflitantes derivadas pelos demais agentes. Neste caso, considera-se a superioridade entre os argumentos em termos da confiança entre os agentes envolvidos e do grau de certeza no que diz respeito à similaridade entre as informações utilizadas. Como uma abordagem de raciocínio não-monotônico, a lógica derrotável e sua relação com sistemas argumentativos estruturados e baseados em regras é interessante, pois permite definir explicitamente formas de lidar com conflitos, e é capaz de descrever e formalizar de forma adequada o problema aqui apresentado.

2.2 RACIOCÍNIO DERROTÁVEL E SISTEMAS ARGUMENTATIVOS

O raciocínio derrotável é um raciocínio baseado em regras, onde as regras podem ser derrotadas por outras regras que apoiam uma conclusão contrária (GOVERNATORI *et al.*, 2004).

O conceito de derrota está no cerne do raciocínio derrotável. Quando surgem conflitos entre as regras, uma relação de preferência/superioridade pode ser usada para resolver esses conflitos. O raciocínio derrotável foi desenvolvido como uma forma de raciocínio não-monotônico, tendo já sido utilizado em diversas aplicações, tais como regulamentos executáveis, contratos, regras de negócios, *e-commerce*, negociação automatizada, raciocínio jurídico e sistemas multiagente (GOVERNATORI *et al.*, 2004).

O conhecimento em uma teoria derrotável, em geral, é composto por um conjunto de regras e uma relação de superioridade entre as regras, sendo as regras divididas em dois tipos: regras estritas e regras derrotáveis.

- **Regras estritas:** são regras no sentido da lógica clássica, onde, sempre que os antecedentes são conclusões lógicas do sistema, então necessariamente o consequente é uma conclusão lógica (por exemplo, "um pinguim é uma ave"). Se uma regra estrita não tiver antecedentes, então trata-se de um fato estrito (por exemplo, "tweety é um pinguim", em um contexto onde se tem absoluta certeza sobre este fato).

- **Regras derrotáveis:** são regras que podem ser derrotadas por evidências contrárias (por exemplo, "uma ave geralmente voa"). Se uma regra derrotável não tiver antecedentes, então trata-se de um fato derrotável, que pode resultar de uma ideia ou percepção imperfeita sobre o mundo (por exemplo, "tweety é uma torda-mergulheira" do ponto de vista de um agente com uma visão imperfeita sobre "tweety").

- **Relação de superioridade:** é uma relação binária entre regras que define se uma regra é superior a outra, e é usada no caso de a aplicação das regras levar a conclusões contraditórias. Por exemplo, a regra que afirma que "um pinguim não voa" pode ser considerada superior à regra que afirma que "uma ave geralmente voa", se for levada em conta a especificidade das regras.

É importante frisar que algumas lógicas baseadas em raciocínio derrotável também fazem distinção entre regras e fatos (ANTONIOU *et al.*, 2000; MODGIL; PRAKKEN, 2014), mas neste trabalho utiliza-se o modelo simplificado de considerar fatos como regras sem antecedentes.

A lógica derrotável é uma lógica não-monotônica introduzida por Nute (2001) como uma forma de formalizar o raciocínio derrotável. A partir de Pollock (1987), a argumentação foi identificada como uma forma de entender o raciocínio derrotável, conforme estudado na filosofia. Desde então, diversos estudos sobre argumentação estruturada baseada em raciocínio derrotável

têm sido publicados (NUTE, 2001; PRAKKEN; VREESWIJK, 2001; GOVERNATORI *et al.*, 2004; BIKAKIS; ANTONIOU, 2010; GARCÍA; SIMARI, 2014; MODGIL; PRAKKEN, 2014). Além disso, *frameworks* de argumentação abstratas foram desenvolvidas (DUNG, 1995; AMGOUD; CAYROL, 2002; BENCH-CAPON, 2003) para apoiar a caracterização de raciocínio não-monotônico em geral em termos teóricos da argumentação. Os elementos básicos destes *frameworks* são as noções de argumentos e *aceitabilidade* de um argumento. Resumidamente, um argumento é aceitável se for possível mostrar que não é possível refutá-lo com mais argumentos.

Um dos trabalhos mais influentes na área de argumentação é a argumentação abstrata de Dung (1995), em que a estrutura interna (premissas e conclusão) dos argumentos individuais é abstraída. O foco da argumentação abstrata está na relação de derrota entre os argumentos, e nos conjuntos de argumentos que defendem seus membros, representando aqueles que, dado um conjunto de argumentos, são aceitáveis. Como a maioria dos trabalhos encontrados na literatura fazem referência a propriedades e conceitos definidos por Dung (1995), a Seção 2.2.1 descreve a argumentação abstrata. Em seguida, na Seção 2.2.2, são discutidos os trabalhos em argumentação estruturada, especialmente os assim chamados *frameworks* de argumentação baseados em regras, nos quais se baseia em grande parte este trabalho.

2.2.1 Argumentação Abstrata

Dung (1995) mostrou que a argumentação pode ser estudada sem considerar a estrutura interna dos argumentos individuais, o que ficou conhecido como argumentação abstrata. Em seu trabalho, os argumentos são nós em um grafo e as arestas representam relações de derrotas entre argumentos. Formalmente, um *framework* de argumentação é um par $AF = (Args, \mathcal{D})$ onde $Args$ é um conjunto finito de argumentos e $\mathcal{D} \subseteq Args \times Args$ é uma relação de derrota. Um argumento A derrota um argumento B se $(A, B) \in \mathcal{D}$ (ou $B\mathcal{D}A$).

Um exemplo simples é mostrado na Figura 2, onde B e D derrotam A; além disso, C derrota B e E derrota D.

Avaliar um argumento consiste em verificar se este argumento é aceitável ou não dados os outros argumentos e a relação de derrota. A aceitabilidade dos argumentos é definida por uma semântica lógica, onde é considerado como um argumento interage com os outros argumentos. Para isso, é necessário primeiramente definir a noção de conjunto de argumentos livre de conflitos, no qual nenhum argumento no conjunto derrota outro argumento pertencente ao conjunto, e de conjunto de argumentos que defendem um argumento, isto é, tal que seus

argumentos derrotam todos os derrotadores de um argumento. Formalmente, seja $(Args, \mathcal{D})$ um *framework* de argumentação e S um conjunto de argumentos ($S \subseteq Args$). S é *livre de conflitos* sse, $\forall A,B \in S,(A,B) \notin \mathcal{D}$. S *defende* um argumento A sse, $\forall (B,A) \in \mathcal{D}$, $\exists C \in S$ t.q. $(C,B) \in \mathcal{D}$.

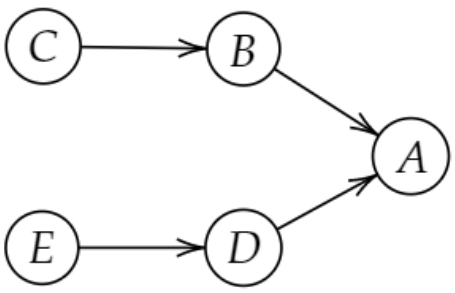
Um conjunto de argumentos ser *livre de conflitos* é importante quando se pensa em agentes racionais. Agentes racionais devem ter uma posição livre de conflitos ao tomar uma decisão. Por exemplo, seria irracional um agente considerar o conjunto $\{A,B\}$ sabendo da derrota entre B e A. Quanto ao conceito de defesa, tem-se, por exemplo, na Figura 2, que o conjunto de argumentos $\{C, E\}$ defende A. Dessa forma, é possível caracterizar a admissibilidade coletiva de um conjunto de argumentos da seguinte forma: Seja S um conjunto de argumentos livre de conflitos em um *framework* AF. S é *admissível* se for livre de conflito e defender todos os elementos em S.

Na Figura 2, os conjuntos: $\emptyset$, $\{C\}$, $\{E\}$, $\{C, E\}$ e $\{A, C, E\}$ são todos admissíveis. Desse modo, um agente possui uma posição consistente ao usar qualquer um desses conjuntos de argumentos admissíveis. Por exemplo, seria irracional para um agente considerar o conjunto $\{A, E\}$, dado que conhece a existência da derrota entre B e A. Assim, considerando o *framework* de argumentação representado na Figura 2, a única maneira racional de considerar o argumento A também deve considerar C e E, dado que eles são necessários para defender A contra B e D, respectivamente.

Um conjunto admissível S é chamado de uma extensão *completa* se, e somente se, todos os argumentos defendidos por S também estão em S. No exemplo da Figura 2, a única extensão completa é o conjunto $\{A, C, E\}$. A extensão completa tem alguns refinamentos. Por exemplo, seja S um conjunto de argumentos:

- S é uma extensão *aterrada* (do Inglês, *grounded*) se for a extensão completa mínima.

Figura 2 – Exemplo de *framework* de argumentação abstrato.



Fonte: Autoria própria.

- S é uma extensão *preferida* se for uma extensão completa máxima.
- S é uma extensão *estável* se S for livre de conflitos, e, $\forall A \in Args \setminus S$, $\exists B \in S$ tal que $B \mathcal{D} A$, ou seja, todo argumento não pertencente a S é derrotado por algum argumento em S.

A extensão aterrada é única e contém todos os argumentos que não são derrotados, bem como os argumentos que são defendidos direta ou indiretamente por argumentos não derrotados.

Quanto à definição da aceitabilidade de conjuntos de argumentos, pode-se definir o *status* de argumentos individuais da seguinte forma: Seja $(Args, \mathcal{D})$ um sistema de argumentação, e $\mathcal{E}_1, ..., \mathcal{E}_n$ suas extensões sob uma determinada semântica. Seja $A \in Args$ um argumento individual, então:

- A é ceticamente aceito sse $\forall \mathcal{E}_i, A \in \mathcal{E}_i$, onde $i = 1, ..., n$.
- A é credulamente aceito sse $\exists \mathcal{E}_i, A \in \mathcal{E}_i$, onde $i = 1, ..., n$.
- A é rejeitado sse $\nexists \mathcal{E}_i$ tal que $A \in \mathcal{E}_i$.

2.2.2 Argumentação Estruturada

Segundo Caminada e Amgoud (2007), a argumentação pode ser vista como um processo de raciocínio que consiste nas quatro etapas a seguir:

1. Construir argumentos (a favor/contra uma "sentença") a partir de uma base de conhecimento.
2. Determinar os diferentes conflitos entre os argumentos.
3. Avaliar a aceitabilidade dos diferentes argumentos.
4. Concluir ou definir as conclusões justificadas.

Embora a argumentação abstrata forneça uma abordagem clara e precisa para formalizar os aspectos da argumentação referentes aos passos 2 a 4, os argumentos são tratados de forma abstraída, ou seja, o conteúdo de tais argumentos não é formalizado. *Frameworks* de argumentação estruturados permitem definir a construção de argumentos a partir de uma linguagem lógica subjacente. Tais argumentos podem também ter uma semântica própria, o que inclui a definição

de diferentes tipos de ataque/derrota, tais como *rebutting* (ataque/derrota à conclusão de um argumento) e *undercutting* (ataque/derrota às premissas de um argumento).

Em particular, este trabalho concentra-se em formalismos de argumentação baseados em regras, que, além de definir semânticas, enfatizam como os argumentos são construídos. Em uma abordagem baseada em regras, os argumentos são formados pelo encadeamento de aplicações de regras na forma de árvores ou grafos de inferência (GARCÍA *et al.*, 2020).

Em especial, este trabalho toma como base a Lógica Derrotável (DL, do Inglês *Defeasible Logic*) apresentada por Antoniou, Billington, Governatori e Maher em diversos trabalhos (ANTONIOU *et al.*, 2000; ANTONIOU *et al.*, 2001; MAHER, 2001; GOVERNATORI *et al.*, 2004), que será melhor detalhada na Seção 2.2.2.1. Em seguida, a Seção 2.2.2.2 apresenta uma breve apresentação sobre duas outras abordagens que convém mencionar: a Programação Lógica Derrotável (DeLP, do Inglês *Defeasible Logic Programming*) (GARCÍA; SIMARI, 2004), e o *framework* ASPIC+ (MODGIL; PRAKKEN, 2014).

2.2.2.1 Lógica Derrotável (DL) de Antoniou, Billington, Governatori e Maher

A Lógica Derrotável (DL), proposta em (ANTONIOU *et al.*, 2000) e (GOVERNATORI *et al.*, 2004), é um formalismo não-monotônico simples, eficiente e flexível, capaz de lidar com muitas intuições diferentes de raciocínio não-monotônico. Dentre algumas características da DL, incluem-se: (1) ser facilmente implementável, possuindo complexidade linear (MAHER, 2001); (2) suportar diferentes variantes de lógica derrotável, podendo ser adaptada a fim de obter uma lógica com propriedades desejadas, tais como com bloqueio ou com propagação de ambiguidade; e (3) possuir um formalismo baseado em argumentação com noções de aceitabilidade semelhantes à noção de aceitabilidade de Dung. A semântica padrão da DL é a com bloqueio de ambiguidade, que se assemelha à extensão aterrada de Dung, porém sendo um pouco menos cética em alguns casos. É possível adaptar tal semântica, fazendo pequenas alterações, para se obter a DL com propagação de ambiguidade, que foi demonstrada ser equivalente à extensão aterrada de Dung. Como o escopo deste trabalho não inclui estudar as diferenças entre essas semânticas, serão apresentadas somente as definições referentes à semântica com bloqueio de ambiguidade, que é a semântica padrão da DL.

Uma BC em DL contém os elementos distintos do raciocínio derrotável conforme já apresentado: regras estritas, regras derrotáveis, e uma relação de superioridade. No entanto, a DL também suporta um tipo específico de regra chamada derrotadora (do Inglês, *defeater*), que

é uma regra que não pode ser utilizada para derivar uma conclusão, mas apenas para evitar que se alcance uma conclusão. Em outras palavras, elas são usadas apenas para derrotar regras derrotáveis ao produzir evidências ao contrário. Embora a ideia de derrotadoras seja interessante, por simplicidade a existência desse tipo de regra será ignorada, uma vez que não contribuem para com o poder de expressividade da DL, podendo inclusive serem simuladas por meio de regras estritas e derrotáveis (ANTONIOU *et al.*, 2001).

Formalmente, uma *regra* $r : Body(r) \hookrightarrow Head(r)$ consiste em *antecedentes* (ou *corpo*), denotado $Body(r)$, que é um conjunto finito de literais, uma flecha, e seu *consequente* (ou *cabeça*) $Head(r)$, que é um literal. Há dois tipos de flechas: $\rightarrow$ e $\Rightarrow$, que correspondem, respectivamente, a uma regra estrita e uma regra derrotável. Dado um conjunto de regras R, R_s denota o conjunto de todas as regras estritas, R_{sd} o conjunto de regras estritas e derrotáveis e R_d o conjunto de regras derrotáveis. $R[q]$ denota o conjunto de regras em R cuja cabeça é q. Se q é um literal, $\sim q$ denota o literal complementar (isto é, se q é um literal positivo p, então $\sim q$ é $\neg p$; e se q é $\neg p$, então $\sim q$ é p).

Uma teoria derrotável $\mathcal{DL} = (R, >)$ é dado por um conjunto finito de regras R e uma relação de superioridade entre as regras $>\subset R \times R$. O mecanismo de raciocínio padrão em DL, proposto em (ANTONIOU *et al.*, 2000), é baseado em uma teoria de *prova* (ou *derivação*), que consiste em gerar uma sequência finita de literais etiquetados (do Inglês, *tagged literals*), representando as conclusões possíveis no que diz respeito a cada literal. Na DL com bloqueio de ambiguidade, há duas etiquetas: ∂ e Δ, que podem ter uma polaridade positiva ou negativa:

- $+\Delta q$, significando que q é definitivamente provável em $\mathcal{DL}$ (i.e., usando apenas regras estritas).
- $-\Delta q$, significando que q não é definitivamente provável em $\mathcal{DL}$.
- $+\partial q$, significando que q é derrotavelmente provável em $\mathcal{DL}$.
- $-\partial q$, significando que q não é derrotavelmente provável em $\mathcal{DL}$.

A provabilidade é baseada no conceito de derivação em $\mathcal{DL}$, que é uma sequência finita $P = (P(1),...,P(n))$ de literais etiquetados. Seja $\#$ uma etiqueta, uma regra é $\#$*-aplicável* em uma dada derivação $(i+1)$-ésima se todos os seus antecedentes foram provados em $P(1..i)$, i.e., $\forall l \in Body(r), +\#l \in P(1..i)$, onde $P(1..i)$ denota a parte inicial da sequência P com tamanho

i. Similarmente, uma regra é *#-descartável* se um de seus antecedentes foi demonstrado não ser provável em $P(1..i)$, i.e., $\exists l \in Body(r)$ t.q. $-\#l \in P(1..i)$.

As seguintes formulações apresentam as condições para Δ:

$+\Delta$:

Se $P(i+1) = +\Delta q$ então

$\exists r \in R_s[q]$ t.q. r é Δ-aplicável

$-\Delta$:

Se $P(i+1) = -\Delta q$ então

$\forall r \in R_s[q]$, r é Δ-descartável

A definição de Δ nada mais é do que um encadeamento para frente (do Inglês, *forward chaining*) das regras estritas. Para que um literal q seja definitivamente provável, é necessário encontrar uma regra estrita com cabeça q Δ-aplicável, isto é, tal que todos os antecedentes em seu corpo foram definitivamente provados anteriormente. Para estabelecer que q não pode ser definitivamente provado, deve-se estabelecer que toda regra estrita com cabeça q é Δ-descartável, isto é, existe pelo menos um antecedente que foi mostrado ser não definitivamente provável.

A seguir são apresentadas as condições para as provabilidades derrotáveis (∂).

$+\partial$:

Se $P(i+1) = +\partial q$ então

(1) $+\Delta q \in P(1..i)$, ou

(2) (2.1)$\exists r \in R_{sd}[q]$ t.q. r é ∂-aplicável, e

(2.2) $-\Delta \sim q \in P(1..i)$ e

(2.3)$\forall s \in R[\sim q]$

(2.3.1)s é ∂-descartável, ou

(2.3.2)$\exists t \in R_{sd}[q]$ t.q.

t é ∂-aplicável e $s < t$

$-\partial$:

Se $P(i+1) = -\partial q$ então

(1) $-\Delta q \in P(1..i)$, e

(2) (2.1)$\forall r \in R_{sd}[q]$, r é ∂-descartável, ou

(2.2) $+\Delta \sim q \in P(1..i)$ ou

(2.3)$\exists s \in R[\sim q]$ t.q.

(2.3.1)s é ∂-aplicável, e

(2.3.2)$\forall t \in R_{sd}[q]$,

t é ∂-descartável ou $s \not< t$

Para demonstrar que q é derrotavelmente provável há duas possibilidades: ou (1) demonstra-se que q já foi definitivamente provado; ou (2) argumenta-se usando a parte derrotável da $\mathcal{DL}$. Especificamente, é necessário que exista uma regra aplicável para q (2.1). No entanto, é preciso também considerar possíveis "ataques", isto é, cadeias de raciocínio que apoiam $\sim q$. Mais especificamente: para provar q derrotavelmente, é necessário estabelecer que $\sim q$ não é definitivamente provável (2.2). Além disso, deve-se considerar a existência de regras que possuem cabeça $\sim q$ (2.3). Essencialmente, cada uma dessas regras s ataca a conclusão q. Para que q seja provável, cada regra s, ou deve ser demonstrada ser descartável (2.3.1), ou deve ser contra-atacada por uma regra t com cabeça q com as seguintes propriedades: (i) t deve ser

aplicável, e (ii) t deve ser mais forte do que s (2.3.2). A intuição para $-\partial$ segue um raciocínio similar, demonstrando que, para toda regra aplicável para um literal q atacada por uma regra s, não existe regra t que contra-ataca s.

Exemplos da aplicação dessa forma de inferência, assim como propriedades, podem ser encontrados em (ANTONIOU *et al.*, 2001).

Antoniou *et al.* (2000) demonstram que a DL satisfaz as propriedades de coerência e consistência. Uma teoria é *coerente* se não houver p tal que $\mathcal{DL} \vdash +\partial p$ e $\mathcal{DL} \vdash -\partial p$, ou $\mathcal{DL} \vdash +\Delta p$ e $\mathcal{DL} \vdash -\Delta p$. Uma teoria é *consistente* se para cada p tal que $\mathcal{DL} \vdash +\partial p$ e $\mathcal{DL} \vdash +\partial \neg p$, também $\mathcal{DL} \vdash +\Delta p$ e $\mathcal{DL} \vdash +\Delta \neg p$. Intuitivamente, a coerência diz que nenhum literal é simultaneamente provável e não provável. A consistência diz que um literal e sua negação podem ser ambos derrotavelmente prováveis somente quando ele e sua negação são ambos definitivamente prováveis (o que é impossível devido à consistência das regras estritas); dessa forma, a inferência derrotável não deriva conclusões inconsistentes.

Com base no mesmo formalismo de representação de BCs com regras derrotáveis, Governatori *et al.* (2004) apresentam um *framework* de argumentação estruturado, assim como definições de aceitabilidade que geram resultados equivalentes ao uso da teoria de prova apresentada acima.

Argumentos são definidos como árvores de prova. No entanto, a DL requer uma noção mais geral de árvore de prova que admite árvores infinitas, de modo que se possa distinguir entre cadeias de raciocínio não-refutadas, mas infinitas (i.e., falácias de argumentação circular), e cadeias realmente refutadas (GOVERNATORI *et al.*, 2004). Um *argumento* para um literal p baseado em um conjunto de regras R é uma árvore (possivelmente infinita) cujos nós são rotulados por literais, a raiz é rotulada por p, e para todo nó com rótulo h:

1. Se b_1,...,b_n rotulam os filhos de h, então existe uma regra em R com corpo b_1,...,b_n e cabeça h.

2. As arestas de uma árvore de prova são rotulados pelas regras usadas para obtê-las.

Dada uma teoria derrotável $\mathcal{DL}$, o conjunto de argumentos que pode ser gerado é denotado $Args_{\mathcal{DL}}$. Um literal é chamado de *a conclusão* de um argumento se este rotula a raiz do argumento. Um *subargumento* (*próprio*) de um argumento A é uma subárvore (própria)[1] da

[1] O conceito de subárvore (própria) é similar ao conceito de subconjunto (próprio). Uma subárvore (não própria) pode ser descrita como uma relação reflexiva e transitiva, isto é, toda árvore é subárvore dela própria, e toda

árvore de prova que representa A. Se apenas regras estritas são usadas, trata-se de um *argumento estrito*. Caso contrário, é chamado de um *argumento derrotável*.

Um exemplo simples é dado a seguir. Considere a teoria derrotável $\mathcal{DL}$ com as seguintes regras:

$$r_1 :\Rightarrow d \qquad r_4 : a,\neg b \Rightarrow c$$

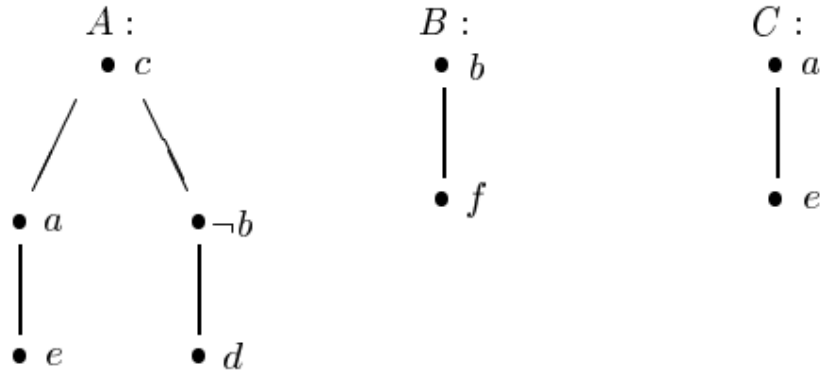
$$r_2 :\rightarrow e \qquad r_5 : e \rightarrow a$$

$$r_3 :\Rightarrow f \qquad r_6 : f \Rightarrow b$$

$$r_7 : d \Rightarrow \neg b$$

A partir dessas regras, os argumentos representados na Figura 3 podem ser construídos (os rótulos das arestas foram ocultados para simplificar a representação). Neste caso, A e B são argumentos derrotáveis, e C é um argumento estrito.

Figura 3 – Exemplos de argumentos construídos para DL.



Fonte: Governatori *et al.* (2004, p. 9)

Quanto aos relacionamentos entre argumentos, um argumento A *ataca* um argumento B se uma conclusão de A é o complemento de uma conclusão de B. Um argumento A *derrota* um argumento derrotável B em q se A ataca B, e A é baseado em r_A e B é baseado em r_B tal que $r_A \not< r_B$. Por exemplo, os argumentos A e B da Figura 3 atacam um ao outro. Desse modo, se $r_7 > r_6$, então A derrota B e B não derrota A. Se $r_6 > r_7$, então B derrota A e A não derrota B. Se não houver relação de preferência entre r_6 e r_7, então A e B se derrotam mutuamente.

Embora seja intuitiva a necessidade de haver uma relação de superioridade entre as regras, é possível definir uma teoria DL que não possua essa relação, porém que possivelmente

subárvore de uma subárvore de uma árvore também é sua subárvore. Por outro lado, uma subárvore própria é uma relação apenas transitiva, denotando todas as subárvores de uma árvore exceto ela mesma. Neste trabalho, usa-se os dois conceitos.

simule essa relação por meio dos outros ingredientes da lógica – as regras estritas e derrotáveis (GOVERNATORI *et al.*, 2004). Dessa forma, a relação de derrota entre argumentos não precisa ser baseada na relação de superioridade entre regras, podendo-se usar outros critérios, tal como o cálculo de força de argumentos apresentado neste trabalho, ou não usar nenhum critério, em cujo caso a relação de derrota equivale à relação de ataque.

A seguir são apresentadas as definições de *apoio* e *undercut*. Ambas são representadas pela relação de um argumento a um conjunto de argumentos, e são fundamentais para a definição de aceitabilidade de argumentos dada em seguida.

Um argumento derrotável A é *apoiado* por um conjunto de argumentos S se todo subargumento próprio de A está em S. Um argumento A ser apoiado por um conjunto S significa que esse conjunto contém todas as premissas que formam a base para se concluir o argumento. Um argumento derrotável A é *undercut* por um conjunto de argumentos S se S apoia um argumento B que derrota um subargumento próprio de A. Um argumento ser *undercut* por S significa que se pode demonstrar que algumas premissas de A não podem ser provados se os argumentos em S forem aceitos. Esta definição de *undercut* é muito importante pois considera que derrotas vindas de argumentos aceitáveis contra argumentos em qualquer ponto da cadeia de raciocínio invalida um argumento.

O cerne da semântica de argumentação é a noção de *argumento aceitável*. Baseado nesse conceito, é possível definir *argumentos justificados* e *conclusões justificadas*, isto é, conclusões que podem ser tomadas após serem considerados os conflitos existentes. Assim, a aceitabilidade é definida da seguinte forma: Um argumento A é *aceitável* no que diz respeito a um conjunto de argumentos S se A é finito, e

1. A é estrito, ou

2. a) A é apoiado por S, e

   b) todo argumento que derrota A é *undercut* por S.

Baseado nisso, é possível definir a construção do conjunto de argumentos justificados de forma incremental. Seja $\mathcal{DL}$ uma teoria derrotável. $J_i^{\mathcal{DL}}$ é um conjunto parcial de argumentos justificados, definido como segue.

- $J_0^{\mathcal{DL}} = \emptyset$;
- $J_{i+1}^{\mathcal{DL}} = \{a \in Args_{\mathcal{DL}} \mid a \text{ é aceitável no que diz respeito a } J_i^{\mathcal{DL}}\}$

Dessa forma, o conjunto de *argumentos justificados* em uma teoria derrotável $\mathcal{DL}$ é $JArgs^{\mathcal{DL}} = \bigcup_{i=1}^{\infty} J_i^{\mathcal{DL}}$. Um literal p é *justificado* se é a conclusão de um argumento em $JArgs^{\mathcal{DL}}$. A construção do conjunto de *argumentos rejeitados* segue um raciocínio similar, e será apresentada em mais detalhes na Seção 3.3, visto ser equivalente à que é utilizada neste trabalho.

No exemplo dado, caso $r_7 > r_6$, tem-se a seguinte sequência de construção do conjunto de argumentos justificados:

- $J_0^{\mathcal{DL}} = \emptyset$;
- $J_1^{\mathcal{DL}} = \{\rightarrow e; \rightarrow e \rightarrow a; \Rightarrow f; \Rightarrow d\}$
- $J_2^{\mathcal{DL}} = J_1^{\mathcal{DL}} \cup \{d \Rightarrow \neg b\}$;
- $J_3^{\mathcal{DL}} = J_2^{\mathcal{DL}} \cup \{a, \neg b \Rightarrow c\}$
- $JArgs^{\mathcal{DL}} = J_3^{\mathcal{DL}}$

Governatori *et al.* (2004) demonstraram que essa semântica de argumentação é equivalente à semântica baseada em derivação da seguinte forma:

- $\mathcal{DL} \vdash +\Delta p$ sse existe um argumento estrito para p em $Args_{\mathcal{DL}}$.
- $\mathcal{DL} \vdash -\Delta p$ sse não existe argumento estrito para p em $Args_{\mathcal{DL}}$.
- $\mathcal{DL} \vdash +\partial p$ sse p é justificado.
- $\mathcal{DL} \vdash -\partial p$ sse p é rejeitado.

A abordagem de argumentação adotada neste trabalho é baseada nesse *framework* de argumentação para DL. Uma das principais vantagens é a representação de argumentos como árvores, o que permite definir a comparação entre argumentos tirando proveito de propriedades relativas a essa estrutura de árvores. Também é interessante o uso das relações de *apoio* e *undercut* na definição de aceitabilidade, que consideram toda a cadeia de raciocínio para definir a aceitabilidade dos argumentos. Isto é especialmente útil ao se considerar argumentos gerados por diferentes agentes que fornecem apoio a argumentos gerados por outros agentes, como proposto neste trabalho.

2.2.2.2 Outras Abordagens de Argumentação Estruturada e Discussões

A Programação Lógica Derrotável, ou DeLP (GARCÍA; SIMARI, 2004; GARCÍA; SIMARI, 2014), fornece um sistema de raciocínio computacional que usa um motor de argumentação para obter respostas de uma BC representada por meio de uma linguagem de programação lógica estendida com regras derrotáveis. A DeLP pode ser vista como uma formalização de raciocínio derrotável em que os resultados da programação lógica e da argumentação são combinados.

A DeLP funciona por meio de consultas. Quando deseja-se saber se um literal é garantido – isto é, derivável – a partir de um programa P, é realizada uma consulta. Existem quatro respostas possíveis para uma consulta: `YES`, se o literal for garantido a partir de P; `NO`, se o complemento do literal for garantido a partir de P; `UNDECIDED`, se nem o literal nem seu complemento são garantidos a partir de P; ou `UNKNOWN`, se o literal não existe no programa P. Informalmente, um literal será garantido se houver pelo menos um argumento apoiando o literal que prevalece após passar por um processo dialético considerando todos os seus contra-argumentos. Um argumento em DeLP é simplesmente uma tupla $\langle L, \mathcal{A} \rangle$ onde L é o literal que é a conclusão do argumento, e $\mathcal{A}$ é um conjunto de regras derrotáveis a partir das quais é possível derivar L.

A DeLP também permite definir preferências entre os argumentos, e os autores argumentam que tais preferências são modulares no sistema de argumentação, podendo ser definidas por diferentes critérios estabelecido sobre o conjunto de argumentos.

A aceitabilidade de um argumento em DeLP é dada por uma árvore dialética, onde o nó raiz é o argumento para o literal consultado. Cada nó filho derrota o nó raiz, e os filhos dos filhos derrotam os filhos, e assim por diante. Desse modo, um literal é garantido se existir um argumento para tal literal, e todos os derrotadores para tal argumento são derrotados por outros argumentos, e estes não são derrotados por nenhum outro.

O *framework* ASPIC+ foi originado do *European ASPIC Project*, tendo sido projetado para incluir inferências estritas e derrotáveis. Diferente da DL e da DeLP, no entanto, o ASPIC+ não define semânticas de argumentação próprias. Em vez disso, eles propõem a conversão de um sistema de argumentação estruturado para um *framework* de argumentação abstrato, a partir do qual as semânticas baseadas em extensões de Dung podem ser aplicadas. O ASPIC+ também se propõe como um *framework* capaz de acomodar diferentes lógicas, a partir das quais deriva-se,

em outro nível de abstração, regras estritas e regras derrotáveis.

Assim, um *sistema de argumentação* AS é composto de: uma linguagem lógica $\mathcal{L}$ fechada sob negação ($\neg$); um conjunto de regras $\mathcal{R} = \mathcal{R}_s \cup \mathcal{R}_d$, composto por regras estritas e regras derrotáveis da forma $\varphi_1, ..., \varphi_n \rightarrow \varphi$ e $\varphi_1, ..., \varphi_n \Rightarrow \varphi$, respectivamente (onde φ_i, φ são meta-variáveis correspondentes a fórmulas em $\mathcal{L}$); e uma função parcial $n : \mathcal{R}_d \rightarrow \mathcal{L}$, que mapeia quais fórmulas em $\mathcal{L}$ correspondem a quais regras derrotáveis em $\mathcal{R}_d$.

Além disso, é definida uma base de conhecimento $\mathcal{K}$, que contém as assim chamadas premissas, que são fatos, isto é, fórmulas similares às regras sem corpo na DL, que podem ser estritas ou derrotáveis. Define-se então uma *teoria de argumentação* como a junção de um sistema de argumentação AS a uma base de conhecimento $\mathcal{K}$, a partir da qual argumentos podem ser gerados de forma recursiva sobre as regras em AS e as premissas em K, similarmente à DL.

Uma característica singular do ASPIC+ é a relação de *undercut*, que, diferente da definição de *undercut* da DL, é tida como um ataque contra a aplicação de uma regra, ou passo de inferência, e não contra uma premissa de um argumento, como em DL. Além do *undercut*, há duas outras relações de ataque: *rebut*, que é o ataque à conclusão do argumento, similarmente à definição de ataque em DL; e *undermine*, que consiste no ataque a uma premissa, similarmente ao ataque a um argumento que não possui subargumentos próprios em DL. Considerando essas relações de ataque e uma ordem de preferência $\succeq$ entre os argumentos, define-se a relação de derrota, a partir da qual é possível construir um *framework* de argumentação abstrata de Dung, permitindo que a aceitabilidade dos argumentos em ASPIC+ possa ser definida por meio das semânticas de extensão.

Uma comparação entre DL e ASPIC+ foi realizada por Lam *et al.* (2016), que demonstraram ser possível instanciar o ASPIC+ em DL. No entanto, os autores não deixam claro como o *undercut* da DL poderia ser traduzido para o *undercut* em ASPIC+ (e vice-versa), visto que ambas as definições possuem significados diferentes. Além disso, diferente do ASPIC+, a DL não tem por objetivo gerar *frameworks* de argumentação abstrata de Dung, embora tenha sido provado que a variante com propagação de ambiguidade da DL gere resultados equivalentes à semântica aterrada de Dung (ANTONIOU *et al.*, 2001). O mesmo ocorre com a DeLP, que também possui semântica própria independente das semânticas baseadas em extensão de Dung, embora haja uma variante chamada $DeLP_{(GR)}$ que gera resultados equivalentes à semântica aterrada (GARCÍA *et al.*, 2020).

Parsons *et al.* (2018) e García *et al.* (2020) realizaram comparações entre ASPIC+ e

DeLP, as quais não serão detalhadas por estarem fora do escopo do trabalho. Nenhum trabalho foi encontrado que realiza uma comparação entre DL e DeLP, o que pode ser uma oportunidade para trabalhos futuros. No entanto, é possível identificar duas diferenças chave entre as duas abordagens: (1) a DL define argumentos como árvores, permitindo analisar precisamente as relações de ataque entre argumentos e subargumentos, enquanto a DeLP define argumentos como tuplas da forma $\langle conclusão, premissas \rangle$, onde as premissas consistem simplesmente no conjunto de regras que colaboram no atingimento da conclusão; e (2) a semântica em DL é definida como a construção incremental de um conjunto de argumentos justificados, analisando-se as relações de ataque existentes entre eles; enquanto a DeLP realiza a verificação (consulta) da aceitabilidade (garantia) de um único argumento com base em um processo dialético. Importa mencionar que a árvore dialética gerada em DeLP não possui qualquer relação com a estrutura de árvore dos argumentos em DL, sendo aquela parte do processo dialético de garantia. Como será apresentado no decorrer desta tese, a representação de argumentos como árvores é fundamental para o cálculo de força de argumentos proposto neste trabalho, sendo mais um motivo para a adoção da DL como base, em detrimento da DeLP.

2.3 SISTEMAS BASEADOS EM CONHECIMENTO DISTRIBUÍDO E RACIOCÍNIO CONTEXTUAL

Esta seção apresenta conceitos relacionados à representação e raciocínio em uma configuração na qual diferentes agentes, com bases de conhecimento (BCs) individuais, distintas e parciais, devem realizar tarefas de raciocínio. Para isso, é necessário que haja uma cooperação entre os agentes de modo a permitir que um agente adquira informações necessárias para o atingimento de conclusões e possivelmente a tomada de decisões. A Seção 2.3.1 apresenta conceitos básicos relacionados a agentes e sistemas multiagente. Em seguida, a Seção 2.3.2 apresenta a ideia de raciocínio contextual, a partir da qual foram propostos os principais trabalhos que formalizam o problema do raciocínio com bases de conhecimento distribuídas e inter-relacionadas, em especial os sistemas multicontexto apresentados na Seção 2.3.3.

2.3.1 Sistemas Multiagente

Há muitas definições de agentes na literatura, mas a maioria parece concordar de que agentes são componentes de software autônomos e fracamente acoplados que exploram diferentes

técnicas de inteligência artificial. De acordo com Ferber (1999) *apud.* Gandon (2002), é possível distinguir duas tendências principais de pesquisa no domínio dos SMAs.

- *Inteligência Artificial Distribuída:* motivada pela resolução distribuída de problemas, geralmente baseada na manipulação de símbolos representando conhecimento, sendo influenciada pelo ramo da inteligência artificial simbólica, seguindo assim um paradigma cognitivista a fim de criar inteligência artificial baseada na manipulação de símbolos. Os trabalhos no paradigma BDI (*belief-desire-intention*) (RAO; GEORGEFF, 1995; SHOHAM, 1993) são um grande exemplo dessa tendência, na qual os agentes são chamados de agentes deliberativos. Eles compreendem modelos de estados mentais internos complexos (suas crenças, seus objetivos e seus planos para alcançá-los), bem como uma arquitetura de raciocínio sofisticada, geralmente composta por diferentes módulos que operam de forma assíncrona (SYCARA, 1998), a fim de derivar as ações futuras e o estado mental da situação atual.

- *Vida Artificial:* trata da simulação de vida (geralmente vida de enxames). Segue uma abordagem subcognitiva minimalista de agentes, onde os comportamentos são baseados em estímulo-resposta, sem representação interna complexa, com comunicação simples baseada em sinais. Eles são chamados de agentes reativos e têm suas raízes no trabalho de Brooks (1991) e Minsky (1988). A inteligência é criada por meio de um processo evolutivo e baseada apenas na combinação emergente de simples interações dos agentes com seu ambiente. O principal critério de evolução é a viabilidade geralmente aplicada por uma simulação de um processo de seleção natural. Sua simplicidade e alta redundância de funções os tornam robustos e altamente tolerantes a falhas. Os agentes reativos são geralmente usados para fins de simulação.

É evidente que este trabalho se enquadra no primeiro grupo, visto que tem como requisito formalizar e concretizar o raciocínio derrotável no contexto de agentes com conhecimento imperfeito.

Algumas características principais de agentes incluem:

- Localidade (do Inglês, *situatedness*): talvez o mais relevante aspecto tendo em vista o caráter distribuído dos agentes. O agente é capaz de receber informações sensoriais do ambiente em que está situado, e é capaz de realizar ações que modificam o ambiente de alguma forma (SYCARA, 1998; JENNINGS, 2000);

- Autonomia: o agente é capaz de operar sem intervenção direta de humanos ou de outras entidades, e tem algum tipo de controle sobre suas ações e seu estado interno (WOOLDRIDGE; JENNINGS, 1995; JENNINGS, 2000; SYCARA, 1998);
- Sociabilidade: o agente é capaz de interagir com outros agentes (e possivelmente com agentes humanos) por meio de algum tipo de linguagem de comunicação (WOOLDRIDGE; JENNINGS, 1995; SYCARA, 1998).

Outras características incluem proatividade, adaptabilidade, mobilidade, etc. Para o escopo deste trabalho, uma caracterização exaustiva sobre os atributos de um agente não é necessária, visto tratar apenas de alguns aspectos do funcionamento de um agente, e não o desenvolvimento de uma arquitetura de agentes como um todo.

De acordo com Gandon (2002), uma definição para SMA, que estende a definição de solucionadores de problemas distribuídos de Durfee e Lesser (1989), é a seguinte: um SMA é definido como uma rede fracamente acoplada de agentes que trabalham juntos como uma sociedade com o objetivo de resolver problemas que geralmente estariam fora do alcance de um agente individual.

De acordo com Sycara (1998), as seguintes são as características de um SMA:

- cada agente possui *informações ou capacidades incompletas* para resolver o problema geral com o qual o sistema lida e, portanto, cada agente possui um ponto de vista limitado.
- *não existe um sistema de controle global*: o comportamento coletivo é resultado de regras e interações sociais, e não de uma autoridade de supervisão central.
- *os recursos são descentralizados*: os recursos necessários para a realização das tarefas atribuídas ao sistema são divididos e distribuídos.

Zambonelli *et al.* (2001) distinguem entre duas classes principais de SMAs:

- *sistemas distribuídos de resolução de problemas* nos quais os agentes são explicitamente projetados para alcançar um determinado objetivo de maneira cooperativa e benevolente.
- *sistemas abertos* nos quais os agentes não são necessariamente coprojetados para compartilhar um objetivo comum, e que podem sair e entrar dinamicamente no sistema.

No primeiro caso, todos os agentes são conhecidos *a priori* e benevolentes uns com os outros e, portanto, podem confiar uns nos outros durante as interações. No último caso, deve-se

levar em consideração a chegada dinâmica de agentes desconhecidos, bem como a possibilidade de os agentes possuírem interesses próprios (isto é, que se preocupa com o atingimento de seus próprios objetivos, e não somente com objetivos coletivos) no curso das interações. Este último caso é exatamente a configuração que este trabalho visa tratar, como já foi explicitado na Seção 1.2 do Capítulo 1.

De acordo com Ferrando e Onaindia (2013), a motivação para introduzir o raciocínio distribuído em um ambiente multiagente é tripla. Em primeiro lugar, os SMAs podem ser benéficos em muitos domínios, particularmente quando um sistema é composto de várias entidades que são distribuídas funcionalmente ou espacialmente. Em segundo lugar, um SMA permite a concepção de vários agentes vinculados a diferentes dispositivos que leem dados de contexto brutos e executam suas próprias inferências contextuais com base em suas percepções, bem como melhoram a capacidade de se detectar mais rapidamente alterações de informações de contexto. Terceiro, a execução distribuída promove a eficiência do processamento paralelo e a robustez do sistema para lidar com problemas complexos, além da simplicidade de uma construção incremental em uma rede de agentes interconectados, evitando assim as falhas críticas e limitações de recursos de sistemas centralizados.

Embora existam diversos trabalhos e diferentes caracterizações e aplicações de agentes e SMAs, poucos trabalhos formalizam especificamente uma espécie de raciocínio totalmente distribuído e descentralizado tal que os agentes possuem conhecimentos inter-relacionados, o que permitiria que eles alcançassem conclusões por meio do fluxo de informações entre eles, de forma colaborativa. Tais formalizações são, no entanto, providas pelos trabalhos baseados no conceito de raciocínio contextual, especialmente os sistemas multicontexto (SMCs).

2.3.2 Raciocínio com Conhecimento Distribuído Inter-Relacionado e Raciocínio Contextual

O método de estruturar sistemas complexos baseados em conhecimento em um conjunto de módulos autônomos tornou-se prática comum em várias áreas, como Web Semântica, Banco de Dados, Dados Vinculados, Ontologias e Sistemas Par-a-Par (P2P, do Inglês, *peer-to-peer*). Nessas aplicações, o conhecimento é frequentemente estruturado em múltiplas fontes e sistemas que interagem entre si, doravante indicados como bases de conhecimento locais ou simplesmente bases de conhecimento (BCs). Esforços têm sido dedicados a fornecer uma base teórica bem fundamentada para a capacidade de representar e raciocinar sobre conhecimento distribuído inter-relacionado, destacando-se os trabalhos que buscam formalizar o raciocínio contextual

(MCCARTHY; BUVAC, 1994; GHIDINI; GIUNCHIGLIA, 2001) e trabalhos em áreas mais recentes da Web Semântica, como correspondência de ontologias (EUZENAT *et al.*, 2007), integração e modularização de ontologias (KUTZ *et al.*, 2004; SERAFINI *et al.*, 2005; BAO *et al.*, 2009), dados vinculados (*linked data*) (HARTIG, 2012), e em sistemas par-a-par (ADJIMAN *et al.*, 2006; CHATALIC *et al.*, 2006; BINAS; MCILRAITH, 2008).

A ideia de raciocínio com BCs inter-relacionadas confunde-se, em grande parte, na literatura, com a noção de raciocínio contextual. McCarthy (1987) introduziu as noções de *contexto* e *raciocínio contextual* na área de Inteligência Artificial como uma abordagem para o problema da *generalidade*. De acordo com ele, cada pedaço de conhecimento parece ser verdadeiro apenas dentro de um domínio particular. Muitas afirmações são verdadeiras apenas em um determinado contexto. No mesmo artigo, McCarthy argumentou que a combinação de raciocínio não-monotônico com raciocínio contextual poderia constituir uma solução adequada para esse problema. A principal solução proposta por McCarthy foi considerar os contextos como objetos de primeira classe; sua proposta consistia em um predicado especial com o qual se poderia especificar axiomas como $ist(c, \forall x.person(x) \rightarrow smart(x))$, que intuitivamente significa que "toda pessoa no âmbito do contexto c é inteligente", sendo ist um símbolo para *"is true"*, isto é, $ist(c,X)$ significa que "X é verdadeiro no contexto c". Regras de levantamento (do Inglês, *lifting*) são usadas para importar/transmitir axiomas entre diferentes contextos. Tais ideias e conceitos foram mais tarde utilizados na formalização da *lógica proposicional de contexto* (PLC, do Inglês, *Propositional Logic of Context*) (MCCARTHY; BUVAC, 1994). Uma formalização similar foi também utilizada por Guha (1992) no sistema CYC, que consistia em uma coleção de "microteorias'" parciais inter-relacionadas.

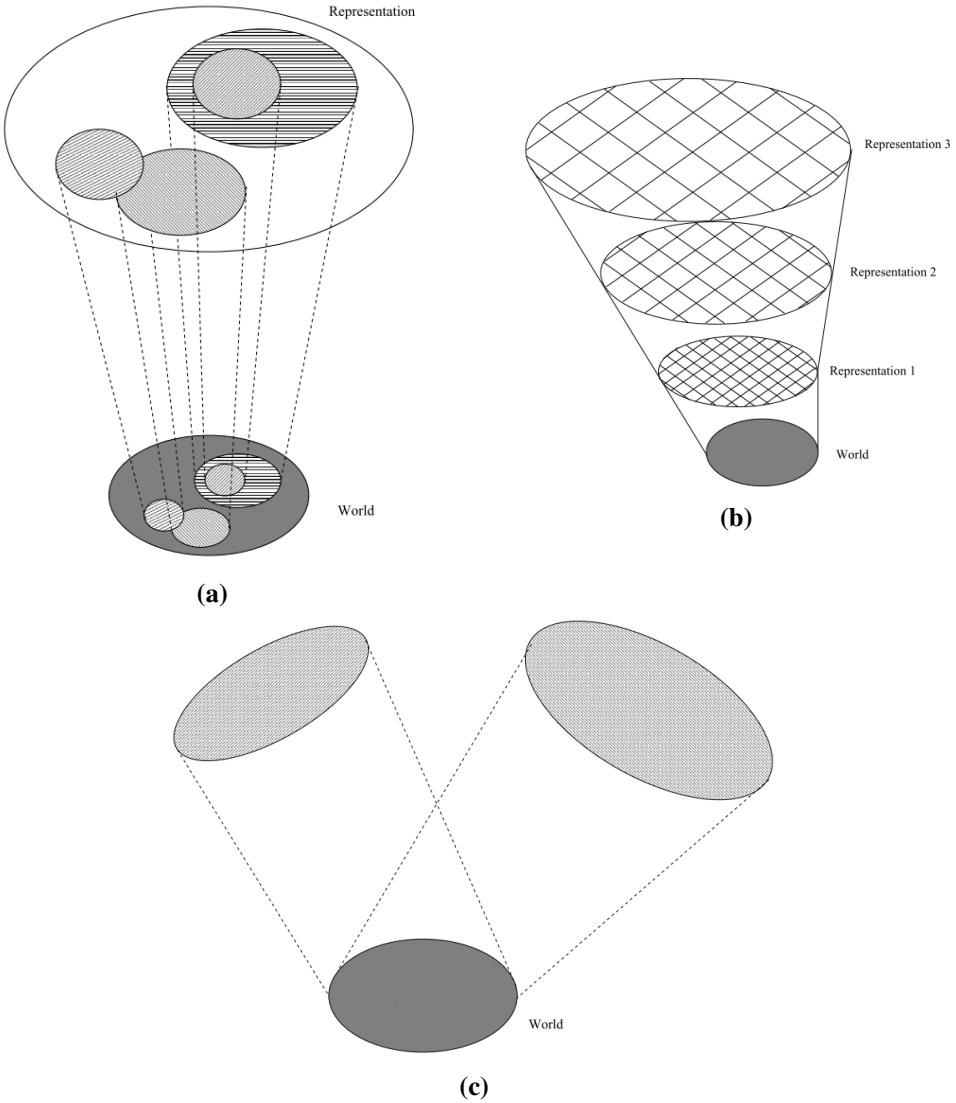
Benerecetti *et al.* (2000) definem contexto como tendo três propriedades, ou dimensões, fundamentais:

- *Parcialidade*: Um contexto é parcial – descreve apenas um subconjunto de um estado de coisas mais abrangente. Essa ideia é ilustrada na Figura 4(a). O círculo inferior representa um estado de coisas: "o mundo", ou todo o conhecimento expressável. Os círculos acima são representações parciais desse estado de coisas.

- *Aproximação*: Um contexto é aproximado – é, em possíveis diferentes níveis, uma abstração do mundo que descreve. Essa intuição é ilustrada na Figura 4(b). Os círculos superiores correspondem a representações possíveis do mundo, em diferentes níveis de aproximação.

- *Perspectiva*: Um contexto é visto em perspectiva – ele reflete um ponto de vista. Um determinado estado de coisas pode, em geral, ser visto de várias perspectivas independentes, conforme ilustrado na Figura 4(c).

Figura 4 – Ilustração das dimensões de (a) parcialidade; (b) aproximação; e (c) perspectiva.



Fonte: Adaptado de Benerecetti *et al.* (2000, p. 10-12).

De acordo com Serafini e Bouquet (2004), as duas principais formalizações que foram propostas com o objetivo de formalizar o raciocínio contextual são a PLC (BUVAC; MASON, 1993; MCCARTHY; BUVAC, 1994), e os SMCs introduzidos em (GIUNCHIGLIA; SERAFINI, 1994) . Serafini e Bouquet (2004) demonstraram que ambos os formalismos são, em grande parte, equivalentes. No entanto os SMCs têm se mostrado os mais adequados no que diz respeito às três propriedades do raciocínio contextual formuladas por Benerecetti *et al.* (2000) (*parcialidade*, *aproximação* e *perspectiva*) e têm se mostrado tecnicamente mais gerais do que a PLC. Desde

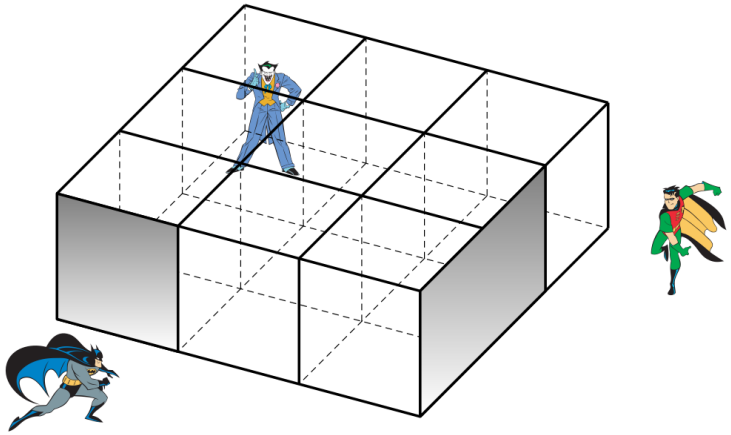
então, diversos trabalhos baseados em SMCs têm sido propostos. A Seção 2.3.3 apresenta em mais detalhes a abordagem de SMCs.

2.3.3 Sistemas Multicontexto

Uma proposta de formalização de uma abordagem totalmente distribuída de raciocínio contextual são os SMCs, que fornecem um formalismo para representar trocas de conhecimento entre BCs, denominadas contextos, heterogêneas e possivelmente não-monotônicas (GIUNCHIGLIA, 1993; BREWKA; EITER, 2007). As informações dos diferentes contextos são vinculadas por meio de *regras de ponte*, e cada contexto é capaz de trabalhar com linguagens e motores de inferência distintas.

Uma ilustração simples das intuições subjacentes aos SMCs é dada pelo assim chamado exemplo da "*caixa mágica*" (GHIDINI; GIUNCHIGLIA, 2001), adaptado por Dao-Tran *et al.* (2015), apresentado na Figura 5.

Figura 5 – A caixa mágica.



Fonte: Dao-Tran *et al.* (2015, p. 544).

Suponha que, em um jogo de computador, os jogadores Batman e Robin perseguiram o jogador Coringa até uma área parcialmente fechada, como mostrado na Figura 5. Robin está ferido e não consegue perceber a sua distância em relação a outros objetos. Nem Batman nem Robin podem dizer a posição exata do Coringa na caixa 3×3: Batman só sabe dizer que ele não está nas colunas 2 e 3, enquanto Robin só sabe dizer que ele está na linha 1. No entanto, se eles trocarem seus conhecimentos parciais, eles podem acabar concluindo que o Coringa está na linha 1 e coluna 1. Pode-se modelar Batman e Robin como contextos que incluem suas informações sobre a posição do Coringa.

As informações são trocadas entre eles usando regras de ponte. Por exemplo, uma regra "$(1 : na\_linha(X)) \leftarrow (2 : na\_linha(X))$" definida pelo Batman, cuja intuição é de que o Batman (contexto 1) "importa" o conhecimento do Robin (contexto 2) sobre algo estar em uma linha, ou que o Batman mantém a posição de que um objeto está em determinada linha se Robin mantém a mesma posição. Os elementos de uma regra, portanto, não são meramente fórmulas ou literais, como na lógica proposicional e lógica de primeira-ordem, mas sim *fórmulas/literais rotulados* da forma $(c_i : x_i)$, onde $c_i \in \{1,..,n\}$ dado um SMC $M = \{C_1,...,C_n\}$, e x_i é uma fórmula/literal em uma dada lógica, implicando, intuitivamente, que c_i define x_i em sua BC local.

As intuições subjacentes aos SMCs podem ser resumidas da seguinte forma:

- Um *contexto* é um subconjunto de um estado global de um indivíduo, ou – mais formalmente – uma teoria parcial e aproximada do mundo da perspectiva de um indivíduo (GIUNCHIGLIA *et al.*, 1993).
- O raciocínio com múltiplos contextos é uma combinação de:
  - *raciocínio local*, que é realizado usando conhecimento local sobre um único contexto. As conclusões locais de um dado contexto derivam de um conjunto de axiomas e regras de inferência que modelam o conhecimento contextual local, e que constituem apenas um subconjunto do conhecimento global.
  - *raciocínio distribuído*, que também leva em consideração as possíveis relações entre contextos locais. Essas relações resultam do fato de que diferentes contextos constituem diferentes representações do mesmo mundo. Elas são modeladas como regras de inferência (conhecidas como regras de *mapeamento*, ou de *ponte*) cujos antecedentes (corpo da regra) podem conter literais que são definidos em outros contextos, o que permite o fluxo de informações entre contextos relacionados.

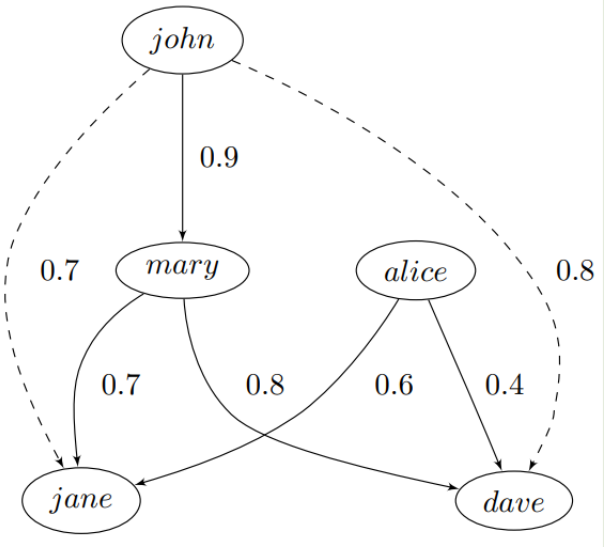
Os problemas mais críticos de raciocínio em SMCs são: (a) a *heterogeneidade* de teorias de contexto locais no que diz respeito à linguagem e ao sistema de inferência usados; e (b) os potenciais conflitos, por conta de informações contraditórias, que podem surgir da interação entre contextos através de seus mapeamentos. Este trabalho trata apenas do problema (b), propondo formas explícitas de resolução de conflitos resultantes da geração de argumentos a favor e contra um dado literal. O problema (a) será ignorado neste trabalho, visto que uma única linguagem para a representação de conhecimento é adotada, sendo baseada em lógica derrotável.

Uma generalização de SMCs não-monotônicos, que estabeleceu as bases para diversos trabalhos posteriores, é a proposta de Brewka e Eiter (2007), cujo foco é a capacidade de lidar com sistemas de raciocínio heterogêneos e distribuídos, ou seja, sistemas que adotam lógicas diferentes nos diferentes contextos. A semântica desta versão do SMC é chamada de semântica baseada em equilíbrios. Dadas as semânticas de um conjunto de contextos (que podem ser lógicas monotônicas ou não-monotônicas), a semântica baseada em equilíbrios é obtida pela composição das semânticas locais com uma metodologia inspirada no paradigma de programação de conjunto de resposta. Cada equilíbrio pode ser visto como um conjunto aceitável de literais, similar ao conceito de extensões de Dung e de modelos em semânticas baseadas em modelos (RUSSELL; NORVIG, 2016). A semântica baseada em equilíbrios não propõe resolução de conflitos por meio de preferências explícitas, além de não ser baseada em argumentação, como proposto neste trabalho.

Bikakis e Antoniou (2010) apresentam a Lógica Derrotável Contextual (CDL, do Inglês *Contextual Defeasible Logic*), uma abordagem baseada em SMC cuja semântica de raciocínio é baseada na DL de (GOVERNATORI *et al.*, 2004), descrita na Seção 2.2.2.1. As características desse trabalho serão melhor descritas no Capítulo 5, que trata dos trabalhos relacionados, quando será comparado à abordagem apresentada neste trabalho. De fato, a CDL cumpre vários dos requisitos para o problema apresentado na Seção 1.2. No entanto, ainda não possui recursos suficientes para atender a todos os requisitos, como o fato de não suportar raciocínio em ambientes dinâmicos, não lidar com conhecimento incerto baseado no grau de similaridade entre conhecimentos heterogêneos, e não lidar com a questão do compartilhamento de foco em consultas.

Outro trabalho que atende a alguns requisitos do problema apresentado nesta tese é o Sistema Multicontexto Distribuído e Dinâmico (DDMCS, do Inglês *Distributed Dynamic Multi-Context System*) de Dao-Tran *et al.* (2010), que suporta ambientes dinâmicos e realiza a vinculação de literais de diferentes contextos por meio de um grau de similaridade. No entanto, a abordagem é baseada na semântica de equilíbrios, carecendo de um raciocínio que permita a resolução de conflitos, além de não ser baseada em argumentação. Também requer um tipo de pré-vinculação (ou configuração) dos contextos a cada alteração no sistema, o que pode gerar uma sobrecarga em ambientes altamente dinâmicos. Alguns detalhes relevantes e comparações também serão apresentados no Capítulo 5, quando serão discutidos os trabalhos relacionados.

Figura 6 – Exemplo de rede de confiança.



Fonte: Parsons *et al.* (2011, p. 880).

2.4 CONFIANÇA EM SISTEMAS DE RACIOCÍNIO DISTRIBUÍDO

A confiança é um mecanismo útil em sistemas distribuídos, onde os agentes podem usar um nível de confiança associado às fontes de informações contraditórias para derivar uma conclusão. Existem muitas abordagens diferentes sobre confiança na literatura (CASTELFRANCHI; FALCONE, 2010; PARSONS *et al.*, 2011; TANG *et al.*, 2012; PINYOL; SABATER-MIR, 2013; PANISSON *et al.*, 2016). Um conceito relevante apresentado por Parsons *et al.* (2011), Tang *et al.* (2012), Panisson *et al.* (2016) é o de relações de confiança entre agentes. Dado um conjunto de agentes $Ags = \{Ag_1,...,Ag_n\}$, uma relação de confiança $\tau \subseteq Ags \times Ags$, onde a existência da relação indica que um agente atribui algum nível de confiança a outro. Por exemplo, $\tau(Ag_i, Ag_j)$ significa que o agente Ag_i tem pelo menos alguma confiança em Ag_j. É importante perceber que esta não é uma relação simétrica, então se $\tau(Ag_i, Ag_j)$ é válido, isso não significa que $\tau(Ag_j, Ag_i)$ é válido. Uma rede de confiança é um grafo direcionado que representa as múltiplas relações de confiança entre os agentes. Uma rede de confiança pode ser definida como: $\Gamma = (Ags, \tau)$, onde Ags é o conjunto de nós no grafo, representando os agentes da rede de confiança, e τ é o conjunto de arestas, onde cada aresta é uma relação de confiança de pares entre agentes de Ags. Um exemplo de uma rede de confiança é apresentado na Figura 6.

As linhas sólidas são relações de confiança diretas, as linhas pontilhadas são relações indiretas derivados das relações diretas. Assim, o agente $john$ confia em $jane$ e $dave$ porque confia em $mary$, que confia em $jane$ e $dave$. No entanto, $john$ não confia, nem mesmo indiretamente,

em $alice$. Logo, a relação τ é definida como uma relação transitiva.

Para medir a confiança, uma função $tr : Ags \times Ags \rightarrow [0,1]$ é usada, que retorna um valor entre 0 e 1, representando o quanto um agente confia em outro. Uma característica interessante é que o valor de confiança de uma relação indireta entre dois agentes Ag_i e Ag_j pode ser derivada dos valores de confiança das relações diretas que formam o caminho no grafo entre Ag_i e Ag_j. Para isso, um operador geral $\otimes^{tr}$ com $tr(Ag_i, Ag_j) = tr(Ag_0, Ag_1) \otimes^{tr} ... \otimes^{tr} tr(Ag_{n1}, Ag_n)$ é usado como o valor de confiança que Ag_i tem em Ag_j de acordo com o caminho Ag_0,...,Ag_n que existe entre Ag_i e Ag_j. Um exemplo de operador que segue uma abordagem mais cética, utilizado na Figura 6, é o min, isto é, o valor da confiança indireta entre dois agentes Ag_i e Ag_j é determinado pelo menor valor de confiança direta no caminho entre Ag_i e Ag_j: $tr(Ag_i, Ag_j) = min(tr(Ag_0, Ag_1), ..., tr(Ag_{n1}, Ag_n))$. Outros operadores podem ser propostos, como o que utiliza a média (avg) entre os valores de confiança, que é um pouco mais otimista ou crédula que o min.

Panisson *et al.* (2016) também introduzem o conceito de confiança aplicada a crenças (ou informações), baseada no valor de confiança aplicado às fontes dessas crenças, que também é um conceito relevante para este trabalho, que propõe um cálculo de força de argumentos tal que um dos parâmetros utilizados nesse cálculo é a confiança do agente que gera o argumento pelos agentes que cooperaram com conhecimento que apoia o argumento. Os autores propõem a ideia de um agente cético que considera o número de fontes das quais recebeu as informações e o valor de confiança de cada uma dessas fontes, a fim de obter uma forma de valor de confiança social. Assim, para um agente Ag_i, eles propõem uma fórmula que soma os valores de confiança das fontes que uma informação φ possui, determinando um valor de confiança social do seguinte modo:

$$trb_i(\varphi) = \frac{\sum_{s \in S^+_\varphi} tr(Ag_i,s)}{|S^+_\varphi| + |S^-_\varphi|}$$

onde $S^+_\varphi = \{s_1,...,s_n\}$ é o conjunto das diferentes fontes (incluindo agentes), que definem φ, e S^-_φ é o conjunto das fontes que definem $\neg\varphi$.

A partir dessa definição, é possível definir também um valor de confiança dos argumentos gerados a partir de informações derivadas de múltiplos agentes. Panisson *et al.* (2016) definem um argumento como uma tupla $\langle S,c \rangle$, onde $S = \{\varphi_1,...,\varphi_n\}$ é o conjunto de apoio do argumento, isto é, as fórmulas a partir das quais é possível derivar c. Dessa forma, o valor de confiança de um argumento, dado um agente Ag_i, é calculado como $tra(\langle S,c \rangle) = trb_i(\varphi_1) \otimes^{tra} ... \otimes^{tra} trb_i(\varphi_n)$.

O operador $\otimes^{tra}$ pode ser definido como max para agentes mais crédulos ou otimistas, min para agentes mais céticos ou pessimistas, e avg para um meio-termo. A partir do valor de confiança para um argumento, é possível utilizá-lo para definir a relação de derrota entre argumentos. Por exemplo, pode-se definir que um argumento Arg_i derrota Arg_j sse $tra(Arg_i) \geq tra(Arg_j)$.

É interessante também mencionar a ideia de prioridades e preferências entre pares em sistemas de inferência par-a-par e entre contextos em SMCs, conforme proposto em (BINAS; MCILRAITH, 2008; BIKAKIS; ANTONIOU, 2010; BIKAKIS; ANTONIOU, 2011), que é similar à ideia de confiança entre agentes. Binas e McIlraith (2008) propõem a ideia de pares distribuídos e interligados para os quais são atribuídos valores distintos de prioridade. Tais valores são então utilizados para definir uma classificação (do Inglês, *rank*) entre os argumentos gerados utilizando-se informações provenientes de diferentes pares, a partir da qual uma relação de ordem é obtida, que é utilizada na definição de derrota entre argumentos. Uma diferença com relação à abordagem de confiança apresentada é que os valores de prioridade têm escopo global, isto é, cada par não possui uma confiança individual sobre os outros pares.

Bikakis e Antoniou (2010), por outro lado, propõem uma abordagem baseada em SMCs que utiliza o conceito de uma ordem de preferência total estrita da forma $T_i = [C_1,C_2,...,C_n]$ para cada contexto C_i no sistema. Assim, similarmente à abordagem de Binas e McIlraith (2008), é proposta a definição de uma classificação entre os argumentos gerados a partir das informações provenientes de diferentes contextos, a partir da qual é definida a relação de derrota entre argumentos. Em (BIKAKIS; ANTONIOU, 2011), uma abordagem alternativa substitui a ordem de preferência total estrita por uma ordem de preferência parcial T_i, modelada como um grafo direcionado cujos vértices representam os contextos do sistema e cujas arestas representam as relações de preferência entre os contextos que rotulam os vértices conectados, de forma que um contexto C_j é preferido por C_i a um contexto C_k, denotado $C_j >^i C_k$, se existe um caminho a partir do vértice C_k até o vértice C_j em T_i.

2.5 CORRESPONDÊNCIA ENTRE INFORMAÇÕES HETEROGÊNEAS POR SIMILARIDADE

Esta seção apresenta alguns conceitos relacionados à correspondência entre informações detidas por diferentes agentes ou bases de conhecimento. Na área de SMAs, o problema de *matchmaking* já foi extensivamente considerado em (SYCARA *et al.*, 2002) e (OGSTON; VASSILIADIS, 2001), e consiste em mecanismos e cálculos de similaridade que permitem

que agentes encontrem informações e serviços providos por outros agentes. Neste trabalho, são considerados apenas mecanismos que fornecem graus de similaridade entre informações atômicas (i.e., literais) detidas por diferentes agentes, seja por meio de algum tipo de similaridade sintática ou semântica. Por exemplo, Sycara *et al.* (2002) apresenta o seguinte cálculo de similaridade sintática entre duas expressões E_i e E_j, que podem ser declarações de variáveis ou restrições, tal que $S(E)$ é o conjunto de palavras em E:

$$Sim(E_i,E_j) = 1 - \left(\frac{\sum_{(u,v)\in S(E_i)\times S(E_j)} d_w(u,v)}{|S(E_i) \times S(E_j)|}\right)$$

Onde $d(u,v)$ denota a distância entre duas palavras u e v. Um relato mais abrangente sobre cálculos de distância entre sequências de caracteres (*strings*) pode ser encontrado em (NAVARRO, 2001).

Outra área em que a correspondência é utilizada é em ontologias. Uma das possibilidades é o uso de ontologias que relacionam conceitos como base para se definir a similaridade entre sentenças em uma aplicação. Um exemplo é dado por Pedersen *et al.* (2004), que propõem medidas de similaridade utilizando a ontologia léxica de língua inglesa WordNet (MILLER, 1995), que organiza substantivos e verbos de forma hierárquica por meio de relações *is-a* ("é um"), o que permite quantificar o quanto um conceito A é similar a um conceito B. Por exemplo, é possível definir uma medida de similaridade que mostra que *automobile* (automóvel) é mais similar a *boat* (barco) do que a *tree* (árvore), devido ao fato de que *automobile* e *boat* compartilham do mesmo ancestral *vehicle* (veículo) na hierarquia do WordNet.

Outra linha de pesquisa que envolve o cálculo de similaridade entre conceitos é a chamada correspondência de ontologias (do Inglês, *ontology matching*) (EUZENAT *et al.*, 2007), cujo objetivo principal é realizar a mescla ou alinhamento entre ontologias. De acordo com Cross (2003), uma vez que a representação sintática de ontologias não é capaz de descrever completamente a semântica de diferentes ontologias, a correspondência automática de ontologias traz consigo um grau de incerteza. Por exemplo, se apenas correspondência sintática for realizada, como é o caso de correspondência baseada em nome, sem o uso de um dicionário de sinônimos, podem ocorrer erros de correspondência. Assim, cada correspondência realizada acaba por ter um grau de certeza, que muitos sistemas especificam por meio de um grau de similaridade entre as informações. Euzenat *et al.* (2007) definem 4 (quatro) tipos de heterogeneidade em ontologias, as quais são resolvidas por meio de correspondência: *sintática*, *terminológica*, *conceitual* e *semiótica*. Um detalhamento mais aprofundado está fora do escopo deste trabalho, que apenas

utiliza esse tipo de recurso como oráculo.

2.6 FOCO E CONSULTAS CONTEXTUALIZADAS

Outra característica importante de se alcançar em agentes dotados de capacidade de raciocínio contextual e distribuído é a consideração do *foco* atual de um agente, que pode envolver informações relevantes sobre sua situação atual, localização, atividade, objeto(s) de interesse ou objetivos.

O conceito de foco adotado neste trabalho possui estreita relação com definições de *contexto* dadas na literatura. Na teoria sobre raciocínio contextual, Sperber e Wilson (1986) definem contexto como "um construto psicológico, um subconjunto das suposições de um indivíduo sobre o mundo", e Giunchiglia *et al.* (1993) definem contexto como "o subconjunto do estado completo de um indivíduo que é utilizado para raciocinar sobre um dado objetivo".

Em sistemas sensíveis ao contexto em geral, Vieira *et al.* (2011) definem contexto como qualquer coisa que cerca, ou que gira em torno, de uma situação, em um dado momento, e que permite identificar o que é e o que não é relevante para a interpretação e compreensão de tal situação.

A ideia de foco também está estritamente relacionada à dimensão de *aproximação* em raciocínio contextual, conforme definida por Benerecetti *et al.* (2000): quanto mais "próximo" se está de um objeto (ou conjunto de objetos) de interesse, maior o foco no que é relevante sobre aquele(s) objeto(s), isto é, mais fina é a granularidade do conhecimento acerca desse(s) objeto(s), e mais abstraídas as características que não são relevantes naquele momento. Portanto, a ideia de contexto, assim como de foco, está estritamente relacionada a uma situação específica no sistema, relacionada a um indivíduo ou entidade, e tal situação pode ser definida como uma ênfase em um determinado subconjunto das informações contidas no sistema.

Um dos primeiros trabalhos a considerar a importância do foco em sistemas computacionais foi o apresentado em (GROSZ, 1977), no qual discute-se a representação e o uso de foco na compreensão de diálogos em sistemas de compreensão de linguagem natural. Especialmente em Ambientes Virtuais Colaborativos, tais como os baseados em Realidade Virtual (RV), a ideia de foco baseado na proximidade e relevância dos objetos é um conceito importante a ser modelado, estando diretamente relacionado ao conceito de *awareness* (RODDEN, 1996; MCEWAN; GREENBERG, 2005).

Trabalhos na área de visão cognitiva em robótica também apresentam conceitos rela-

cionados ao conceito de foco, o qual tem relação com a ideia de *atenção* (ITTI; KOCH, 2001; FRINTROP, 2011; VEMULA *et al.*, 2018). Assim como ocorre com um ser humano, a visão de um robô é capaz de enxergar muito mais do que seu "cérebro" é capaz de processar. Por esse motivo, é necessário que apenas porções do que é percebido seja processado preferencialmente, o que se reflete na atenção do robô em algo, por exemplo, em um objeto existente no ambiente.

Neste trabalho, adota-se o conceito de foco como um conjunto de conhecimentos que é relevante para um objetivo de raciocínio, isto é, para uma consulta realizada por um agente acerca do valor-verdade de uma dada proposição. Tais conhecimentos podem estar relacionados à atual situação, atividade, objeto(s) de interesse, localização ou objetivos nos quais o agente que emite a consulta possa estar engajado. Esse conhecimento pode não necessariamente estar limitado às crenças do agente, podendo ser referente a percepções atuais do ambiente que porventura não tenham sido agregadas pelo agente na forma de crenças, como ocorre, por exemplo, no modelo de revisão de crenças orientado a dados (DBR, do Inglês, *Data-oriented Belief Revision*) apresentado por Paglieri e Castelfranchi (2004). Considerando que a consulta possa ser respondida por meio da colaboração de múltiplos agentes, é necessário que o agente que emite a consulta compartilhe esse conhecimento de foco com os demais agentes, e que esse conhecimento seja também propagado para todos os agentes que venham a se envolver em tal processo de raciocínio.

Um trabalho relevante sobre SMAs e que lida com a questão do foco é o CArtAgO (do Inglês, *Common ARTifact infrastructure for AGents Open environments*), um *framework*/infraestrutura de uso geral que permite programar e executar ambientes virtuais, ou ambientes de *software*, para SMAs (RICCI *et al.*, 2011). Especificamente, esse *framework* permite a criação de artefatos do ambiente, e, dentre outras funcionalidades, permite que um agente se foque em um artefato, de modo que as propriedades observáveis desse artefato sejam mapeadas diretamente para as crenças do agente, permitindo também que o agente atualize suas crenças quando ocorrem alterações no artefato.

O presente trabalho concentra-se especificamente na possibilidade de os agentes compartilharem conhecimento de foco ao realizarem raciocínio distribuído por meio da passagem de mensagens contendo tais conhecimentos. Isto se deve ao fato de que se pressupõe que os agentes em um sistema possam estar geograficamente distantes entre si, possivelmente impedindo que todos os agentes colaboradores no raciocínio tenham acesso direto ao conhecimento de foco ligado a um artefato específico. Este problema específico não é resolvido pelo CArtAgO, mas,

pressupondo que todos os agentes tenham acesso virtual a todos os artefatos em um sistema que utiliza o CArtAgO, seria possível realizar uma forma de integração com a solução proposta neste trabalho: em vez de se compartilhar o conhecimento de foco diretamente no conteúdo da mensagem, poder-se-ia enviar, na mensagem, a referência ao(s) artefato(s) CArtAgO que contêm o conhecimento de foco, de modo que os demais agentes possam se focar nos mesmos artefatos enquanto estiverem processando o raciocínio. Tal integração pode ser um interessante tema para um trabalho futuro.

Por fim, nenhum trabalho foi encontrado na literatura que tratasse especificamente da ideia de compartilhamento de foco. No entanto, García e Simari (2014) apresentam a ideia de *consultas contextuais*, nos quais diferentes agentes podem realizar consultas a serviços especializados em raciocínio chamados de *DeLP-Servers*. Juntamente a essas consultas, os agentes podem enviar conhecimento "privado", também chamado pelos autores de *informações contextuais*, que podem ser integradas a um programa DeLP temporariamente, apenas no contexto da consulta realizada. Isso possibilita que um *DeLP-Server* responda à consulta considerando, não somente o conhecimento nele armazenado, mas também conhecimento contextual específico providenciado pelo agente que emite a consulta, o que se assemelha muito à abordagem de compartilhamento de conhecimento de foco proposta neste trabalho. No entanto, tal abordagem não consiste em uma forma de raciocínio distribuído com conhecimentos inter-relacionados de diferentes agentes.

2.7 DISCUSSÃO

Este capítulo apresentou os principais conceitos encontrados na literatura que dão base à abordagem proposta neste trabalho. Foi demonstrado que a representação de conhecimento usando lógica clássica não é suficiente para lidar com o problema proposto, visto que não admite a não-monotonicidade, ou derrotabilidade das consequências lógicas. Isto é resolvido por uma abordagem de raciocínio não-monotônico, enfatizando-se a lógica derrotável, juntamente com um raciocínio baseado em argumentação, como um formalismo adequado para lidar com esse problema.

Em seguida, apresenta-se os principais conceitos relacionados a sistemas com BCs distribuídas. Os SMAs são apresentados como a definição mais abrangente e estudada na literatura acerca de entidades autônomas baseadas em lógica e capazes de realizar raciocínio, além de contar com características importantes no que diz respeito à localidade (*situatedness*) e

sociabilidade de um agente, isto é, sua capacidade de interação com o ambiente e com outros agentes, e definições acerca da natureza do ambiente, que pode ser aberto e composto de agentes com interesses próprios.

No entanto, os trabalhos em SMAs não formalizam uma solução para o problema específico do raciocínio distribuído quando suas bases de conhecimento são inter-relacionadas. Isto é resolvido pelos trabalhos na área de raciocínio contextual, especialmente os assim chamados SMCs, que definem regras especiais chamadas *regras de ponte*, que conectam informações definidas em diferentes BCs, permitindo assim uma forma de raciocínio totalmente distribuído baseado no fluxo de informações entre as BCs. No entanto, não há trabalhos sobre SMCs que propõem soluções para todos os aspectos levantados na Seção 1.2 do Capítulo 1.

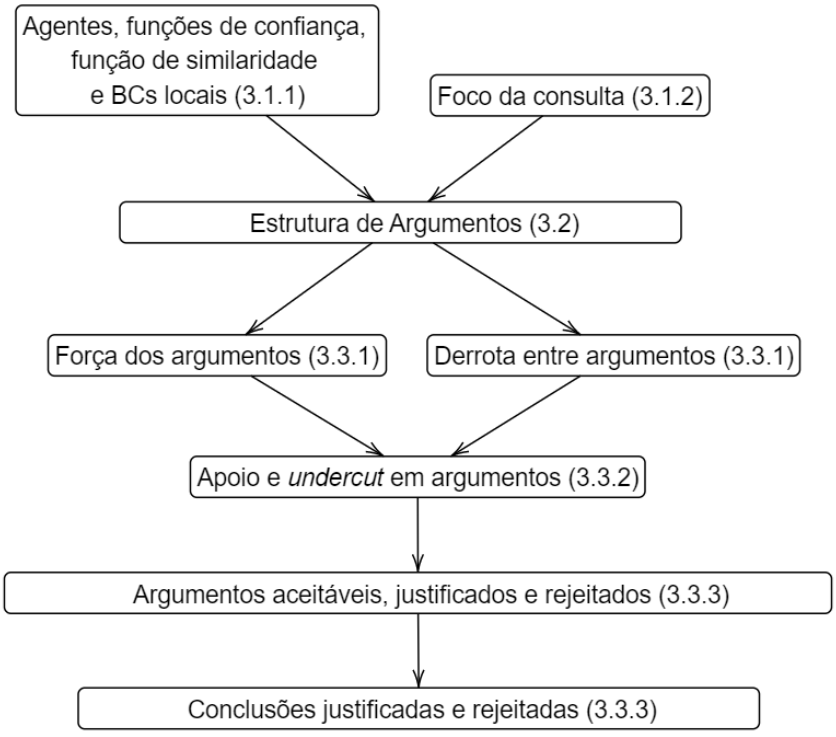
Também são apresentados outros conceitos relevantes para este trabalho, que serão trabalhados como ingredientes na solução proposta: a confiança em SMAs; a correspondência entre informações heterogêneas; e o conceito de foco e compartilhamento de foco.

3 UM FRAMEWORK DE ARGUMENTAÇÃO BASEADO EM REGRAS PARA RACIOCÍNIO DISTRIBUÍDO

Este capítulo apresenta o *framework* proposto neste trabalho com foco em suas definições formais e exemplos. A Seção 3.1 apresenta a formalização da arquitetura, incluindo os agentes, seu formalismo de representação de conhecimento, sua função de confiança, a formalização do foco de consulta e a formalização do problema. A Seção 3.2 apresenta os argumentos e como eles são derivados das BCs dos agentes no sistema. A Seção 3.3 apresenta a semântica de argumentação. A Seção 3.4 apresenta a conversão do formalismo apresentado para uma teoria DL centralizada, demonstrando, assim, que o formalismo herda as propriedades da DL, tais como consistência e coerência.

A Figura 7 apresenta um resumo dos conceitos apresentados.

Figura 7 – Resumo dos conceitos apresentados no Capítulo 3.



Fonte: Autoria própria.

3.1 FORMALIZAÇÃO DA ARQUITETURA E DO PROBLEMA

Esta seção apresenta uma formalização do SMA e da representação de conhecimento baseada em regras (Seção 3.1.1), dos focos de consulta e bases de conhecimento de foco (Seção

3.1.2), e, por fim, a formalização do problema (Seção 3.1.3).

3.1.1 Arquitetura Multiagente e Representação de Conhecimento Baseado em Regras

A seguir é apresentada a definição de um SMA, denominado Sistema Multiagente baseado em Raciocínio Derrotável Distribuído, ou DDRMAS (do Inglês, *Distributed Defeasible Reasoning-based Multi-Agent System*).

Definição 1 (DDRMAS). *Um DDRMAS é uma tupla $\mathcal{S} = (Ags, F_Q, \Theta, st)$, tal que $Ags = \{a, b,...,z\}$ é um conjunto finito de agentes situados no ambiente em um determinado momento, $F_Q = \{\alpha, \beta,...,\omega\}$ é um conjunto finito de focos de consulta em um determinado momento, Θ é uma função de similaridade e st é um limite de similaridade.*

Um agente na abordagem proposta é definido conforme a Definição 2.

Definição 2 (Agente). *Um agente $a \in Ags$ é definido como uma tupla da forma (KB_a, P_a), tal que:*

- *KB_a é a base de conhecimento do agente;*
- *$P_a : Ags \rightarrow [0,1]$ é uma função de confiança, tal que, dados dois agentes b e c em Ags, se $P_a(b) > P_a(c)$, então a confia em b mais do que em c. Se $P_a(b) = P_a(c)$, então eles são igualmente confiáveis para a. $P_a(a) = 1$ por padrão, ou seja, o agente tem total confiança em si mesmo.*

A função de confiança é útil para a resolução de conflitos que podem surgir da interação entre os agentes, e sua definição específica em cada agente pode resultar de mecanismos diversos, os quais definimos como oráculo, não sendo abordados neste trabalho. Portanto, a definição real de tal função é deixada em aberto para o desenvolvedor da aplicação específica. Por exemplo, poder-se-ia considerar que os agentes constroem uma relação de confiança entre si à medida que interagem uns com os outros. Uma abordagem é a proposta por (CASTELFRANCHI; FALCONE, 2010), que considera a expectativa de um agente de que o outro produza um determinado resultado que inclua algum objetivo do agente. Tal confiança entre os agentes pode ser aumentada a cada vez que um agente alcança adequadamente um objetivo com base no auxílio prestado por outro agente, e diminuída caso esse auxílio se mostre prejudicial.

Pressupõe-se também que um valor de confiança padrão é atribuído a cada $P_a(b)$, para $b \neq a$. Tal valor padrão permite que todos os agentes possam considerar argumentos vindos

de outros agentes sem que eles precisem se conhecer e/ou previamente terem-lhes atribuído um valor de confiança específico. Este valor pode ser também configurado pelo desenvolver da aplicação a fim de simular diferentes intuições. Por exemplo, um valor padrão 0, juntamente à fórmula de força de argumentos apresentada na Seção 3.3.1, faz com que um agente nunca considere argumentos provenientes de agentes para os quais ele ainda não tenha atribuído um valor de confiança específico, enquanto um valor padrão $0{,}5$ é capaz de considerá-los, mesmo que de forma mais precavida.

A Definição 3 apresenta o conceito de uma base de conhecimento de um agente.

Definição 3 (Base de Conhecimento)**.** *Uma base de conhecimento KB_a é um conjunto de regras na forma*

$$r_{ai} : \langle a, x\rangle \hookleftarrow Body(r)$$

onde r_{ai} é o identificador da regra, $a \in Ags$, $i \in \mathbb{N}^+$, $\langle a, x\rangle$ é um cr-literal que é o consequente (cabeça) da regra, e $Body(r)$ é um conjunto de r-literais, também denominado corpo, que representa os antecedentes da regra, isto é, uma conjunção de r-literais. O símbolo $\hookleftarrow$ deve ser substituído por $\leftarrow$, o que significa que a regra é uma regra estrita, ou por $\Leftarrow$, o que significa que a regra é uma regra derrotável.

Por conveniência de notação, as chaves $\{...\}$ em torno do corpo de uma regra são omitidas no decorrer da tese, e uma regra tal que $Body(r) = \emptyset$ é denotada $r_{ai} : \langle a, x\rangle \hookleftarrow$.

As bases de conhecimento KB_a são conjuntos de regras compostas pelos assim chamados *literais rotulados* (referenciados no decorrer do texto como *r-literais*) p da forma $\langle \mathbb{D}, x\rangle$, onde $\mathbb{D}$ é um rótulo chamado de *definidor* do *r-literal*, e x é um literal. O definidor de um *r-literal* p pode ser referenciado pela função $D(p)$, podendo ser uma referência direta a um agente $a \in Ags$ (neste caso, o *r-literal* é chamado de *literal rotulado concreto*, ou *cr-literal*) ou um símbolo $@$, significando que qualquer agente em Ags pode definir o literal rotulado, que neste caso específico é chamado de *literal rotulado esquemático* (ou *er-literal*). Outro tipo de *r-literal* é o *literal rotulado de foco*, ou *fr-literal*, cujo formato é $\langle \mathcal{F}, x\rangle$, onde $\mathcal{F}$ é um símbolo que marca o *r-literal* como um *fr-literal*. Estes não são usados nas bases de conhecimento locais dos agentes, mas sim em bases de conhecimento de foco, conforme apresentado na Seção 3.1.2. Por fim, existe um tipo de *r-literal* que é gerado e usado exclusivamente na estrutura de argumentos, que

são os *r-literais* instanciados, ou *ir-literais*, os quais são apresentados quando da definição de um argumento (Definição 7) na Seção 3.2.

O literal enclausurado por um *r-literal* p pode ser referenciado por meio da função $L(p)$. Contudo, nos exemplos apresentados, é possível que, em vez do literal, seja utilizada uma fórmula com variáveis livres. Neste caso, pressupõe-se a existência de um mecanismo de instanciação dessas fórmulas para fórmulas fechadas com constantes, ou unificação, similarmente ao proposto por García e Simari (2014). Portanto, sempre que for apresentada uma regra com *r-literais* enclausurando fórmulas com variáveis livres, o leitor deve interpretá-las como o conjunto de regras que podem ser geradas pela instanciação dessas fórmulas usando as constantes definidas no próprio exemplo (e.g., no Exemplo 1.1, m_1 é uma constante que substitui a variável M nas fórmulas).

Um literal, portanto, representa informação atômica (x) ou a negação da informação atômica ($\neg x$). Dado um literal x, seu literal complementar é um literal correspondente à negação de x, que pode ser denotado como $\sim x$. Mais precisamente, $\sim x \equiv \neg x$ e $\sim (\neg x) \equiv x$. Para qualquer *r-literal* $p = \langle \mathbb{D}, x \rangle$, seu *r-literal* complementar é denotado $\sim p = \langle \mathbb{D}, \sim x \rangle$. O conjunto de literais em uma base de conhecimento KB é denotado $x \in V$.

O conjunto de *r-literais* em uma base de conhecimento genérica KB é chamado de vocabulário, denotado $V^R = V^{CR} \cup V^{ER}$, onde V^{CR} é o conjunto de *cr-literais* e V^{ER} é o conjunto de *er-literais*, tal que V^{CR} e V^{ER} são disjuntos, isto é $V^{CR} \cap V^{ER} = \emptyset$. Considera-se também que tais conjuntos são fechados sob negação. Isto é, sempre que um *r-literal* $\langle \mathbb{D}, x \rangle \in V^R$, o *r-literal* $\langle \mathbb{D}, \sim x \rangle \in V^R$.

Uma base de conhecimento $KB_a = KB_a^s \cup KB_a^d \cup KB_a^m \cup KB_a^@$, tal que KB_a^s, KB_a^d, KB_a^m e $KB_a^@$ são mutuamente disjuntos (*pair-wise disjoint*), pode ser composta por quatro tipos diferentes de regras:

- *Regras estritas locais* (KB_a^s), chamadas simplesmente de *regras estritas* no decorrer do texto;
- *Regras derrotáveis locais* (KB_a^d), chamadas simplesmente de *regras derrotáveis* no decorrer do texto;
- *Regras derrotáveis de mapeamento* (KB_a^m), ou *regras de mapeamento* no decorrer do texto;

- *Regras derrotáveis de mapeamento esquemáticas* ($KB_a^{@}$), ou *regras esquemáticas* no decorrer do texto.

As regras dos três últimos tipos são todas derrotáveis, portanto, são marcados com a flecha $\Leftarrow$ no lugar do $\hookleftarrow$, enquanto as estritas são marcadas com $\leftarrow$. Por conveniência, o *cr-literal* na cabeça de uma regra r também pode ser referenciado como $Head(r)$, e o tipo da regra (estrita ou derrotável) pode ser referenciado por $Type(r) \in \{strict, defeasible\}$.

Uma regra estrita é uma regra $r_{ai} : \langle a, x\rangle \leftarrow Body(r)$ tal que, para cada *r-literal* $\langle \mathbb{D}, y\rangle \in Body(r)$, $\mathbb{D} = a$, ou seja, todos os *r-literais* são *cr-literais* locais, uma vez que são definidos localmente por a. Da mesma forma, uma regra derrotável local é uma regra $r_{ai} : \langle a, x\rangle \Leftarrow Body(r)$ tal que para cada *cr-literal* $\langle \mathbb{D}, y\rangle \in Body(r)$, $\mathbb{D} = a$. A diferença é que as regras estritas não podem ser derrotadas, ou seja, elas representam verdades inquestionáveis no sistema e são interpretadas pela lógica clássica: sempre que todos os *cr-literais* no corpo de uma regra estrita são consequências lógicas de KB_a^s, então o *cr-literal* consequente da regra também é uma consequência lógica de KB_a^s, e, portanto, de KB_a. Uma regra local derrotável, por outro lado, não pode ser aplicável[1] para apoiar seu consequente se houver evidência contrária adequada (não inferior), ou seja, é possível que exista uma regra com um consequente contrário que impediria a aceitação do consequente de outra regra.

Um requisito importante é que KB_a^s não pode ter ciclos nem inconsistências. A manutenção de tal consistência é deixada para um especialista humano ou por algum mecanismo de revisão/atualização de crenças, que está fora do escopo deste trabalho. A intuição é que regras estritas são conhecimentos consolidados no âmbito de um sistema. Eles podem ser propriedades intrínsecas que sempre se mantêm, como a declaração "Um pinguim é uma ave" ou a crença de um agente sobre sua própria existência. Por outro lado, o conjunto de regras derrotáveis tolera regras contraditórias ou cadeias de regras que levam a linhas de raciocínio circulares ou autodestrutivas. No caso de regras contraditórias (ou seja, regras com consequentes complementares), sendo ambas aplicáveis, algum critério é necessário para escolher entre apenas um dos consequentes, ou seja, o argumento para uma conclusão sempre derrota seu contra-argumento, de modo que apenas um deles é justificado. No caso de cadeias de regras circulares (ciclos), os argumentos baseados nessas regras não podem ser usados para apoiar uma conclusão, mas podem ser usados para atacar um contra-argumento, de modo que a conclusão não seja totalmente negada,

[1] Uma regra aplicável é tal que todos os seus antecedentes são provados como consequência lógica do sistema, seguindo a mesma noção usada em DL e apresentada na Seção 2.2.2.1.

mas seja marcada como $undec$, como em (GARCÍA; SIMARI, 2004; BIKAKIS; ANTONIOU, 2010). O mesmo ocorre com as linhas de raciocínio autodestrutivas, em que o complemento de um *r-literal* já aparece anteriormente na cadeia de regras. A estrutura de argumentação para ambos os últimos casos será discutida na Seção 3.2.

Uma regra de mapeamento é uma regra $r_{ai} : \langle a, x\rangle \Leftarrow Body(r)$ tal que para cada $\langle \mathbb{D}, y\rangle \in Body(r)$, $\mathbb{D} \in Ags$ e existe pelo menos um $\langle \mathbb{D}, y\rangle \in Body(r)$ tal que $\mathbb{D} \neq a$, ou seja, *cr-literais* no corpo de uma regra também podem ser definidos por outros agentes específicos, caso em que são chamados de *cr-literais estrangeiros*. Isso é o que permite que os agentes funcionem com conhecimento parcial, exigindo a colaboração de outros agentes para chegar a conclusões. A intuição de uma regra de mapeamento denotada por $r_{ai} : \langle a, x\rangle \Leftarrow \langle b, y\rangle$ é a seguinte: "Se a sabe que o agente b conclui y, então a o considera uma premissa válida para concluir x se não houver qualquer evidência contrária adequada".

Uma regra esquemática é uma regra $r_{ai} : \langle a, x\rangle \Leftarrow Body(r)$ tal que para cada $\langle \mathbb{D}, y\rangle \in Body(r), \mathbb{D} \in Ags \cup \{@\}$, e existe pelo menos um $\langle \mathbb{D}, y\rangle \in Body(r)$ tal que $\mathbb{D} = @$, ou seja, os *r-literais* podem ser definidos pelo próprio agente (*cr-literais* locais), outros agentes (*cr-literais* estrangeiros) ou por um agente arbitrário (*er-literais*). A intuição de uma regra esquemática denotada $r_{ai} : \langle a, x\rangle \Leftarrow \langle @, y\rangle$, é a seguinte: "Se a sabe que algum agente conclui um literal *similar o suficiente* a y, então a o considera uma premissa válida para concluir x se não houver nenhuma evidência contrária adequada".

Regras esquemáticas são fundamentais para permitir que os agentes usem conhecimento de agentes arbitrários sem saber *a priori* se tais agentes têm o conhecimento necessário. Este é um dos principais recursos para permitir o raciocínio distribuído em um ambiente dinâmico e aberto onde o conhecimento dos agentes sobre o conhecimento de outros agentes é parcial ou inexistente. Por exemplo, suponha um agente a que tem uma regra com um antecedente x que não é uma consequência lógica de seu conhecimento local, e suponha que, alguns momentos antes, um agente b que sabe que x é verdadeiro entrou no sistema. Neste caso, sem regras esquemáticas, a não perguntaria a b sobre x e, portanto, não conseguiria concluir que x é verdadeiro.

Considera-se também a possibilidade de *er-literais* serem vinculados (ou instanciados) a qualquer *cr-literal* que lhes seja similar o suficiente. Isso permite suportar uma gama ainda mais ampla de cenários, especialmente aqueles que envolvem raciocínio baseado em informações com vocabulários heterogêneos. Por exemplo, suponha um cenário no qual o conhecimento é derivado de uma ontologia OWL (BECHHOFER *et al.*, 2004). Neste caso, uma função de

similaridade poderia ser baseada em axiomas que usam a propriedade `owl:sameAs`, que indica que dois indivíduos com URIs distintas representam o mesmo conceito ou objeto no mundo real. Outros exemplos podem incluir similaridade sintática, como o WordNet (MILLER, 1995), que pode ser usada em cenários envolvendo reconhecimento de voz ou texto, e outros processos de correspondência sintática e semântica, como o Larks (SYCARA *et al.*, 2002). A Seção 2.5 apresenta mais alguns exemplos de trabalhos envolvendo correspondência entre informações heterogêneas.

Portanto, define-se uma *função de similaridade* entre *r-literais*, bem como os conceitos de *limite de similaridade* e *similar o suficiente*, por meio da Definição 4.

Definição 4 (Função de Similaridade, Limite de Similaridade, e Similar o Suficiente)**.** *A* ***função de similaridade*** $\Theta : V^R \times V^R \rightarrow [0,1]$ *é tal que maior similaridade entre* p *e* q *resulta em um valor* $\Theta(p,q)$ *maior. Dado um* ***limite de similaridade*** $st \in [0,1]$*, dois r-literais* p *e* q *são* ***similares o suficiente*** *se* $\Theta(p,q) \geq st$*. Dados dois r-literais idênticos, a similaridade entre eles sempre será* 1*, i.e.,* $\forall p \in V^R, \Theta(p,p) = 1$.

Exemplo 1. Este exemplo apresenta a codificação do cenário motivador apresentado no Capítulo 1, Seção 1.1.1. Nele, há 5 (cinco) agentes: o agente de Alice (a), o agente de Barb (b), o agente de Charles (c), o agente de Dennis (d) e o agente de Eric (e). O conhecimento dos agentes e suas funções de confiança são apresentadas no Quadro 1.

Quadro 1 – Definições dos agentes para o Exemplo 1.

<table>
<tr>
<td>
Alice: a
$\mathbf{KB_a} = \{r_{a1} : \langle a, \neg ed(M)\rangle \Leftarrow \langle a, dc(M)\rangle$
$r_{a2} : \langle a, col(M)\rangle \Leftarrow \langle @, ed(M)\rangle$
$r_{a3} : \langle a, \neg col(M)\rangle \Leftarrow \langle @, \neg ed(M)\rangle \}$
$P_a(b) = 0,4; P_a(c) = 0,6; P_a(d) = 0,2; P_a(e) = 0,8$
</td>
<td>
Barb: b
$\mathbf{KB_b} = \{r_{b1} : \langle b, \neg ed(M)\rangle \Leftarrow \langle b, hv(M)\rangle$
$r_{b2} : \langle b, col(M)\rangle \Leftarrow \langle @, ed(M)\rangle$
$r_{b3} : \langle b, \neg col(M)\rangle \Leftarrow \langle @, \neg ed(M)\rangle \}$
$P_b(a) = P_b(c) = P_b(d) = P_b(e) = 0,5$
</td>
</tr>
<tr>
<td>
Charles: c
$\mathbf{KB_c} = \{r_{c1} : \langle c, ed(M)\rangle \Leftarrow \langle @, avl(M)\rangle \}$
$P_c(a) = P_c(e) = 1; P_c(b) = P_c(d) = 0,4$
</td>
<td>
Dennis: d
$\mathbf{KB_d} = \{r_{d1} : \langle d, \neg ed(M)\rangle \Leftarrow \langle @, am(M)\rangle \}$
$P_d(c) = P_d(e) = 1; P_d(a) = P_d(b) = 0,4$
</td>
</tr>
<tr>
<td colspan="2">
Eric: e
$\mathbf{KB_e} = \{r_{e1} : \langle e, spa(M)\rangle \Leftarrow \langle e, hv(M)\rangle, \langle e, pbc(M)\rangle \}$
$P_e(a) = 0,4; P_e(b) = 0,6; P_e(c) = 0,2; P_e(d) = 0,8$
</td>
</tr>
</table>

Fonte: Autoria própria.

Alice tem algum conhecimento sobre algumas espécies, como o *death cap* (representado por $dc(M)$), que ela sabe não ser comestível ($\neg ed(M)$, regra r_{a1}). Suponha também que Alice e Barb compartilhem o mesmo conhecimento de que, se um cogumelo for comestível ($ed(M)$), eles devem coletá-lo ($col(M)$) e, se não for comestível ($\neg ed(M)$), eles não devem coletá-lo

($\neg col(M)$). Eles também estão dispostos a considerar a opinião de qualquer agente conhecido sobre o fato de um cogumelo ser comestível ou não, ou seja, se algum agente afirma que um cogumelo é comestível, então eles estão dispostos a aceitar que seja verdade se não houver evidência contrária adequada. Tal conhecimento é assim apresentado em a (regras r_{a2} e r_{a3}) e b (regras r_{b2} e r_{b3}). Barb também acredita que um objeto não é comestível se tiver uma *volva* ($hv(M)$, regra r_{b1}).

Charles acredita que um cogumelo é comestível se for uma *amanita velosa* ($avl(M)$, regra r_{c1}). No entanto, ele não consegue descrever um cogumelo desta espécie; assim, a regra r_{c1} é uma regra esquemática, o que significa que ele está disposto a aceitar que o cogumelo é uma *amanita velosa* se qualquer outro agente declarar que um cogumelo é similar o suficiente a *amanita velosa*.

Dennis acredita que um objeto não é comestível se for uma *amanita* ($am(M)$, regra r_{d1}). No entanto, ele não sabe nada sobre as características de amanitas; assim, um *er-literal* é usado no corpo da regra.

Finalmente, Eric acredita que um cogumelo é uma amanita da primavera ($spa(M)$) se tiver algumas propriedades, como ter um volva ($hv(M)$) e uma capa acastanhada ($pbc(M)$, regra r_{e1}). Amanita da primavera é o mesmo tipo de cogumelo que *amanita velosa*, mas assume-se que os agentes no sistema não possuem tanta certeza sobre isso. Portanto, os agentes consideram que $\Theta(\langle\mathbb{D}, spa(M)\rangle,\langle\mathbb{D}, avl(M)\rangle) = 0{,}8$. Da mesma forma, a amanita da primavera é um tipo de amanita, e os agentes consideram $\Theta(\langle\mathbb{D}, spa(M)\rangle,\langle\mathbb{D}, am(M)\rangle) = 0{,}5$. Portanto, considera-se que a função de similaridade tem o valor 1 para todos os literais lexicamente idênticos e, especificamente, o valor $0{,}8$ entre a amanita da primavera e amanita velosa e $0{,}5$ entre a amanita e qualquer tipo específico de amanita. A função de similaridade entre todos os demais literais tem automaticamente o valor 0. Suponha também que o limite de similaridade st para todo o sistema seja de $0{,}4$.

Cada agente tem diferentes níveis de confiança entre si, conforme também mostrado na Figura 1. Por exemplo, Alice confia mais em Eric ($P_a(d) = 0{,}8$), seguido por Charles ($P_a(c) = 0{,}6$), Barb ($P_a(b) = 0{,}4$) e Dennis ($P_a(d) = 0{,}2$). Em contraste, Barb confia em todos os agentes igualmente.

□

3.1.2 Foco de Consulta e Base de Conhecimento de Foco

Um agente $a \in Ags$ (chamado de *agente emissor*) é capaz de emitir consultas para si mesmo ou para outros agentes. Quando um agente emite uma *consulta inicial*, que resulta em *consultas subsequentes* a outros agentes (chamados de *agentes colaboradores*), é necessário que cada um desses agentes fique ciente de conhecimentos específicos relacionados ao foco da consulta inicial. A Figura 8 ilustra essas consultas e papéis para o Exemplo 1.1.

Este é um recurso fundamental para permitir um raciocínio eficaz em um contexto distribuído, porque nutre a capacidade de raciocínio dos agentes colaboradores com a possibilidade de considerar informações relevantes – e possivelmente não conhecidas *a priori* por esses agentes – para a consulta do agente emissor, que podem estar relacionadas ao seu atual objeto de interesse, atividade, situação ou localização. Portanto, o foco é tido como uma base de conhecimento, i.e., um conjunto de regras, que pode conter um tipo especial de regra, chamada de *regra de foco*. A intuição de uma regra de foco é que ela seja considerada, por cada agente que recebe a consulta, como definida localmente pelo próprio agente de forma temporária e limitada ao processamento da consulta em questão. Isto permite que o agente raciocine a partir da união de suas próprias regras locais com as regras de foco recebidas, de modo que, assim, ele possa alcançar conclusões que ele possivelmente não conseguiria alcançar se não tivesse recebido as regras de foco.

As definições a seguir definem uma estrutura chamada de *foco de consulta*, bem como o conceito de *base de conhecimento de foco*. O foco da consulta é importante pois, além de encapsular uma BC de foco, possui um identificador único e contém, como metainformações, o *r-literal* que inicialmente foi consultado e o agente emissor que inicialmente emitiu a consulta e criou o foco. Tais informações permitem uma forma de rastreabilidade do foco de consulta, visto que ele deverá ser propagado de um agente para o outro durante o raciocínio.

Definição 5 (Foco de Consulta)**.** *Um foco de consulta para um r-literal p é uma tupla $\alpha = (p, a, KB_\alpha^{\mathcal{F}})$, de modo que α é um identificador único do foco de consulta, a é o agente que criou a consulta e $KB_\alpha^{\mathcal{F}}$ é a base de conhecimento de foco de α.*

Definição 6 (Base de Conhecimento de Foco)**.** *Uma* base de conhecimento de foco $KB_\alpha^{\mathcal{F}}$ *é um conjunto de regras derrotáveis (Definição 3) e* regras de foco *da forma $r_{\alpha i}^{\mathcal{F}} : Head(r_{\alpha i}^{\mathcal{F}}) \Leftarrow Body(r_{\alpha i}^{\mathcal{F}})$, tal que $\exists p \in \{Head(r_{\alpha i}^{\mathcal{F}})\} \cup Body(r_{\alpha i}^{\mathcal{F}})$ t.q. $p = \langle \mathcal{F}, x \rangle$, i.e., pelo menos um dos r-literais que compõem a regra é um literal rotulado de foco, ou* fr-literal.

Um *fr-literal*, quando recebido por um agente, deve ser interpretado temporariamente como se fosse uma *cr-literal* definido localmente. Por exemplo, se um agente b recebe uma consulta com a regra $r^{\mathcal{F}}_{\alpha 1} : \langle \mathcal{F}, y_1 \rangle \Leftarrow$, ele deve interpretar a regra como se fosse $r^{\mathcal{F}}_{b1} : \langle b, y_1 \rangle \Leftarrow$. Esse processo é chamado de *localização* dos *fr-literais*, e ocorre na derivação de argumentos, como apresentado na Seção 3.2. Quando o processamento da consulta é finalizado, a regra é descartada se originalmente já não pertencia à base de conhecimento local de b.

Vale notar, também, que as BCs de foco podem ter apenas regras derrotáveis[2]. Isso ocorre porque a introdução de regras estritas exigiria um mecanismo para evitar a introdução de ciclos e inconsistências ao subconjunto de regras estritas de uma base de conhecimento, mesmo que temporariamente. Além disso, parece contraintuitivo para um agente receber uma regra estrita de outro agente e prontamente considerá-la também uma regra estrita. Regras estritas em um sistema provavelmente serão definidas por um especialista humano ou serão derivadas de um processo rigoroso de revisão de crenças.

3.1.3 Formalização do Problema

O problema que se busca resolver já foi descrito textualmente no Capítulo 1, Seção 1.2. No entanto, agora que foram formalizadas as estruturas a serem manipuladas para que se resolva o problema, pode-se também formalizar o problema em si.

Primeiramente, por conveniência, algumas definições adicionais são apresentadas. A base de conhecimento global $KB_{\mathcal{S}}$ de um sistema $\mathcal{S}$ é a união das bases de conhecimento de todos os agentes do sistema, ou seja, $KB_{\mathcal{S}} = \bigcup_a^{Ags} KB_a$. Uma base de conhecimento global estendida $KB_{\mathcal{S}\alpha} = KB_{\mathcal{S}} \cup KB^{\mathcal{F}}_{\alpha}$ inclui o conhecimento de foco de um foco de consulta α. A base de conhecimento estendida local de um agente a, denotada $KB_{a\alpha} = KB_a \cup KB^{\mathcal{F}}_{\alpha}$ representa o conhecimento local de um único agente aumentado com o conhecimento de foco de um foco de consulta α. Similarmente, $V^R_{\mathcal{S}\alpha}$ e $V^R_{a\alpha}$, respectivamente, denotam o vocabulário global incluindo o foco de consulta α e o vocabulário local de um agente a incluindo α.

Dadas essas definições, é possível formalizar o problema da seguinte forma: "Dado um agente a com um foco de consulta $\alpha = (p, a, KB^{\mathcal{F}}_{\alpha})$ para um *r-literal* p em um sistema multiagente $\mathcal{S}$, a deseja saber se o *cr-literal* p é uma consequência lógica de $KB_{\mathcal{S}\alpha}$ ou não". Este é um subproblema do seguinte problema: "Dado um sistema multiagente $\mathcal{S}$ considerando um

[2] As próprias regras de foco são sempre derrotáveis, mas uma BC de foco pode também possuir regras derrotáveis comuns, conforme a Definição 3, que também podem ser regras de mapeamento e regras esquemáticas.

foco de consulta $\alpha = (p, a, KB^{\mathcal{F}}_{\alpha})$, quais *r-literais* em $KB_{\mathcal{S}\alpha}$ são suas consequências lógicas?". Como é explicado mais detalhadamente na Seção 3.2, é possível que um *r-literal* p em $KB_{\mathcal{S}\alpha}$ possa ser uma consequência lógica ($KB_{\mathcal{S}\alpha} \models p$) – isto é, p tem valor-verdade $true$ – ou não ser uma consequência lógica ($KB_{\mathcal{S}\alpha} \not\models p$) – isto é, p tem valor-verdade $false$. No entanto, pode não ser possível afirmar que p é uma consequência lógica ou não, o que ocorre especialmente quando existem argumentos falaciosos (linhas de raciocínio circulares ou autodestrutivas). Nesses casos, um valor-verdade $undec$ é atribuído a p, como será apresentado no Capítulo 4.

É importante observar que o uso da definição de uma BC global estendida diz respeito apenas à formalização do modelo e do problema: não é necessário que os agentes realizem operacionalmente uma união de suas BCs. A solução, como apresentada na derivação de argumentos na Seção 3.2 e no Capítulo 4, consiste em cada agente colaborar para a construção de argumentos baseada em sua própria BC estendida local, e então enviar tais argumentos a outros agentes a fim de gerar argumentos maiores que permitam responder ao problema apresentado.

Exemplo 1.1. Suponha que, em algum momento, Alice encontre um cogumelo, denominado internamente com a constante m_1, que possui uma volva e uma capa acastanhada pálida. Assim, o agente de Alice emite a si mesmo uma consulta com o seguinte foco:

$$p = \langle a, col(m_1)\rangle \quad \alpha = (p, a, KB^{\mathcal{F}}_{\alpha}) \quad KB^{\mathcal{F}}_{\alpha} = \{r^{\mathcal{F}}_{\alpha 1}, r^{\mathcal{F}}_{\alpha 2}\}$$

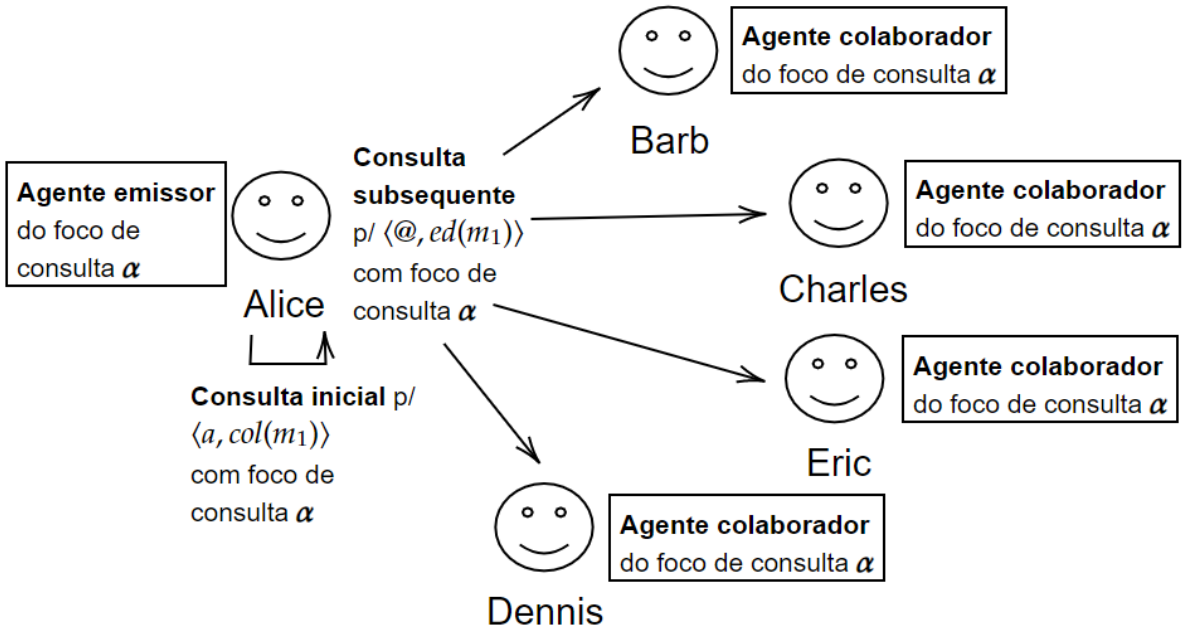
$$r^{\mathcal{F}}_{\alpha 1} : \langle \mathcal{F}, hv(m_1)\rangle \Leftarrow \qquad r^{\mathcal{F}}_{\alpha 2} : \langle \mathcal{F}, pbc(m_1)\rangle \Leftarrow$$

onde α é um novo foco de consulta cuja BC de foco contém regras que representam as características do cogumelo. A Figura 8 ilustra essa consulta inicial, as consultas subsequentes e os papéis dos agentes.

Ao receber sua própria consulta, o agente a não consegue chegar a uma conclusão baseado apenas no conhecimento disponível localmente. Assim, ele deve enviar consultas subsequentes para outros agentes, incluindo, em cada consulta, o mesmo α criado anteriormente para permitir que os agentes raciocinem efetivamente sobre o cogumelo percebido. Neste caso, Alice conclui que ela pode coletar o cogumelo m_1 por causa do argumento gerado por Charles com a ajuda de Eric (r_{c1} e r_{e1}), embora Barb sozinha (r_{b1}) e Dennis com Eric (r_{d1} e r_{e1}) gerem argumentos contrários. O motivo desta decisão é baseado na estratégia padrão de cálculo de força de argumentos proposta na Seção 3.2.

□

Figura 8 – Ilustração das consultas e papéis dos agentes para o Exemplo 1.1.



Fonte: Autoria própria.

Exemplo 1.2. Suponha que, simultaneamente ao foco de consulta α do Exemplo 1.1, Barb encontre um cogumelo (m_2) que ela sabe ser um *death cap* e deseja saber se deve coletá-lo ou não. Portanto, ele emite a si mesmo uma consulta com o seguinte foco:

$$p : \langle b, col(m_2)\rangle \quad \beta = (p, b, KB^{\mathcal{F}}_{\beta}) \quad KB^{\mathcal{F}}_{\beta} = \{r^{\mathcal{F}}_{\beta 1}\}$$

$$r^{\mathcal{F}}_{\beta 1} : \langle \mathcal{F}, dc(m_2)\rangle \Leftarrow$$

Nesse caso, um novo foco de consulta, β, é originado de Barb. É importante notar que, neste caso, não se inclui o conhecimento de foco vinculado ao foco de consulta α. Conforme apresentado na Seção 3.3, Barb decidirá não coletar m_2, visto que o agente de Alice responderá à consulta informando que m_2 não é comestível.

□

Exemplo 1.3. Suponha que, posteriormente, Eric saia do sistema e Alice encontre outro cogumelo m_3 com as mesmas características de m_1. Suponha também que o agente de Alice não aprendeu as razões que a levaram a coletar m_1 na consulta α (tal habilidade será estudada em trabalhos futuros). Nesse caso, ao emitir uma nova consulta com foco γ, que é idêntico a α, ele só recebe respostas de Barb e Dennis. Barb ainda afirma que o cogumelo não é comestível com base em suas regras locais e regras de foco. Dennis responde negativamente à consulta para $\neg ed(m_3)$, pois depende de que algum agente responda positivamente acerca de $am(m_3)$, o que

não ocorre dada a ausência de Eric. De qualquer forma, uma resposta positiva para $\neg ed(m_3)$ é retornada por Barb, o que fará com que o agente de Alice tome a decisão de não coletar m_3. Este exemplo demonstra como o sistema oferece suporte ao raciocínio em ambientes dinâmicos em que os agentes saem e entram no ambiente continuamente. □

3.2 ESTRUTURAS DE ARGUMENTOS

Esta seção apresenta como os argumentos são derivados das regras de cada agente, juntamente com o conhecimento de foco de um determinado foco de consulta.

Um *argumento* $A \in Args$, tal que $Args$ é o conjunto de argumentos que pode ser gerado a partir de uma base de conhecimento KB, é uma árvore n-ária derivada do encadeamento de regras de KB.

A Definição 7 apresenta formalmente um argumento, juntamente com algumas funções que capturam elementos e propriedades importantes de um argumento. É importante notar que os tipos de *r-literais* que compõem os argumentos não são os mesmos dos que compõem as regras, pois ocorre a *localização* dos *fr-literais* (caso (I) da definição) e a *instanciação* dos *cr-literais* estrangeiros (caso (II)) e dos *er-literais* (caso (III)). A localização dos *fr-literais* consiste em criar *cr-literais* locais a partir de *fr-literais*, pois, como foi discutido na Seção 3.1.2, os *fr-literais* devem ser tidos pelos agentes como se fossem informações definidas localmente. A instanciação de *cr-literais* estrangeiros e *er-literais* consiste em explicitar as interdependências entre agentes, incluindo-se o grau de similaridade entre os *r-literais* tidos como similares o suficiente, gerando assim *ir-literais*. Ressalta-se também que a localização e instanciação ocorrem somente na derivação de argumentos, não incorrendo em nenhuma alteração nas bases de conhecimentos em si. A Figura 9, referente ao Exemplo 2, ilustra graficamente como um argumento é derivado das bases de conhecimento dos agentes juntamente com um foco de consulta.

Um *r-literal instanciado*, ou *ir-literal*, tem a forma $p_{inst} = \langle b, x, \theta \rangle$, onde $b \in Ags$, x é um literal e θ é o *grau de similaridade* entre os *r-literais* utilizados para gerar p_{inst}. Um *ir-literal* p_{inst} pode ser derivado de um *er-literal* p, tal que existe uma regra cuja cabeça é um *cr-literal* similar o suficiente p', sendo o grau de similaridade $\theta = \Theta(p,p')$. A fim de manter uma uniformidade e facilitar o cálculo de força dos argumentos mais adiante, *ir-literais* também são criados a partir de *cr-literais* estrangeiros, porém com o valor de θ fixado em 1, visto que um *cr-literal* estrangeiro no corpo de uma regra sempre será idêntico ao *cr-literal* na cabeça da regra correspondente. Os *ir-literais* em um argumento são peças fundamentais para

o cálculo de força de argumentos proposto neste trabalho, visto sinalizarem em que pontos o argumento apresenta dependências de argumentos definidos por outros agentes, e em que pontos foi necessário instanciar um *er-literal* com um *cr-literal* similar o suficiente, que pode agregar graus de similaridade variáveis que terão impacto na força do argumento.

Portanto, dado um argumento A baseado em uma regra r, para cada *r-literal* q no corpo de r é criado:

- um *cr-literal*, no caso de q ser um *cr-literal* local, um *fr-literal* ou um *er-literal* tal que existe uma regra local cuja cabeça é um literal idêntico (caso (I) da Definição 7); ou
- um *ir-literal*, no caso de se tratar de:
  - um *r-literal* estrangeiro (caso (II)), ou
  - um *er-literal* tal que existe uma regra definida por outro agente cuja cabeça possui um *cr-literal* similar o suficiente, de acordo com a função de similaridade (caso (III)).

Em qualquer um dos casos, esse *cr-literal*/*ir-literal* será uma das premissas do argumento, sendo também, portanto, a conclusão de um subargumento B de A.

Conforme o caso (IV) da definição, um argumento também pode conter um *nó-folha falacioso*, rotulado com $q!$, onde q é o *r-literal* que gera a falácia. Há dois tipos de falácias detectadas: argumentação circular e argumentação autodestrutiva. A primeira ocorre quando, na derivação de um argumento, surge um *r-literal* que já se encontra em algum nó ancestral. Um argumento autodestrutivo ocorre quando surge o complemento de um *r-literal* que já se encontra em algum nó ancestral. Como apresentado na Definição 8, um nó-folha falacioso é considerado um tipo específico de argumento.

Definição 7 (Argumento)**.** *Dada uma base de conhecimento KB, um argumento $A \in Args$ baseado em uma regra $r : p \hookleftarrow Body(r) \in KB$ tal que $p = \langle a,x \rangle$ é uma árvore n-ária cujo nó raiz é p. Dado $n = |Body(r)|$, se $n = 0$, então a árvore tem um único nó-filho rotulado com o símbolo $\top$, que é um nó-folha. Se $n > 0$, então a árvore tem n nós-filhos, cada um correspondendo a um r-literal $q = \langle \mathbb{D}, y \rangle$ tal que $q \in Body(r)$, e cada nó-filho é:*

(I) um cr-literal local $q_{loc} = \langle a, y \rangle$, se $\mathbb{D} \in \{a, \mathcal{F}, @\}$ (ou seja, q é um cr-literal local, um fr-literal ou um er-literal), e existe uma regra r' t.q. $Head(r') = \langle \mathbb{D}', y \rangle$ e $\mathbb{D}' \in \{a, \mathcal{F}\}$, a partir da qual um subargumento B pode ser derivado, ou

(II) *um ir-literal $q_{inst} = \langle b, y, 1\rangle$, com $b \in Ags \setminus \{a\}$, no caso de $\mathbb{D} = b$ (ou seja, q é um cr-literal estrangeiro), e existe uma regra r' t.q. $Head(r') = \langle \mathbb{D}', y\rangle$ t.q. $\mathbb{D}' \in \{b, \mathcal{F}\}$, a partir da qual um subargumento B pode ser derivado, ou*

(III) *um ir-literal $q_{inst} = \langle b, y', \theta\rangle$, com $b \in Ags$, no caso de $\mathbb{D} = @$ (ou seja, q é um er-literal) e existe uma regra r' t.q. $Head(r') = q' = \langle b, y'\rangle$ e $\theta = \Theta(q, q') \geq st$ (ou seja, q e q' são similares o suficiente), a partir da qual um subargumento B pode ser derivado, ou*

(IV) *um nó-folha falacioso $q_{fall} = \langle b, y', \theta\rangle!$, com $b \in Ags$, no caso de $\mathbb{D} \in \{b, \mathcal{F}, @\}$ e existir uma regra r' t.q. $Head(r') = q' = \langle \mathbb{D}', y'\rangle$ t.q. $\mathbb{D}' \in \{b, \mathcal{F}\}$ e $\theta = \Theta(q, q') \geq st$, e existe um nó q_{orig} ancestral de q_{fall} na árvore, tal que $D(q_{fall}) = D(q_{orig})$ e: ou $L(q_{fall}) = L(q_{orig})$, ou $L(q_{fall}) = \sim L(q_{orig})$.*

*Cada aresta da árvore é rotulada com o identificador da **regra** usada para derivar o argumento, e pode ser referenciada por meio da função $Rule(A) = r$.*

*$Conc(A)$ referencia a **conclusão** de um argumento A, que é o cr-literal que rotula seu nó-raiz.*

*$ArgD(A)$ referencia o **definidor** de um argumento A, que é o agente que define a conclusão de A, i.e., $ArgD(A) = D(Conc(A))$.*

*$SA(A)$ referencia o conjunto de **subargumentos próprios** de um argumento A. Um subargumento próprio de um argumento A, denotado A', é uma subárvore própria da árvore que representa o argumento A. Subargumentos próprios serão chamados simplesmente de **subargumentos** no decorrer do texto.*

*$ISA(A) \subseteq SA(A)$ referencia o conjunto de **subargumentos instanciados** de um argumento A, isto é, os subargumentos de A cujas conclusões são ir-literais. Formalmente: $B \in ISA(A)$ sse $B \in SA(A)$ e $Conc(B) = \langle b, x, \theta\rangle$, t.q. $b \in Ags$, $x \in V$ e $\theta \in [0,1]$.*

*$DISA(A) \subseteq ISA(A)$ referencia o conjunto de **subargumentos instanciados diretos** de um argumento A, que são os subargumentos instanciados de A que não são subargumentos instanciados de outro subargumento instanciado de A. Formalmente, $B \in DISA(A)$ sse $B \in ISA(A)$ e $\exists C \in ISA(A)$ t.q. $B \in ISA(C)$.*

*$Prem(A)$ referencia o conjunto de **premissas** de um argumento A, que são todos os nós em um argumento, exceto sua raiz e os nós rotulados com $\top$. Uma premissa pode ser um cr-literal local, um ir-literal ou um nó-folha falacioso. Se $Prem(A) = \emptyset$, então A é um **argumento***

***base**. Pode-se expressar as premissas como as conclusões dos subargumentos próprios de um argumento, i.e.,* $Prem(A) = \{q \mid q = Conc(B), B \in SA(A)\}$.

$IRLs(A) \subseteq Prem(A)$ *referencia o conjunto de premissas de um argumento* A *que são **ir-literais** . Se* $IRLs(A) = \emptyset$ *ou se* $\forall q_{inst} \in IRLs(A)$, $D(q_{inst}) = a$, *então* A *é um **argumento local**. Caso contrário, é um **argumento distribuído**. Pode-se expressar esse conjunto como as conclusões dos subargumentos instanciados de um argumento, i.e.,* $IRLs(A) = \{q \mid q = Conc(B), B \in ISA(B)\}$.

$DIRLs \subseteq IRLs(A)$ *referencia o conjunto de **ir-literais diretos** de um argumento* A, *que são as conclusões dos argumentos instanciados diretos de* A. *Esse conjunto define a fronteira entre o que é baseado apenas nas regras de* KB_a *(a BC do agente que define o argumento) e o que é derivado das BCs de outros agentes. Formalmente,* $DIRLs = \{q \mid q = Conc(B), B \in DISA(B)\}$.

$Fall(A) \in \{true, false\}$ *indica se um argumento é **falacioso** (*$true$*) ou não (*$false$*). Um argumento não falacioso é chamado de argumento **válido**. Um argumento é falacioso se tiver uma premissa que é um nó-folha falacioso. Formalmente,* $Fall(A) = true$ *sse* $q! \in Prem(A)$ *t.q.* $q \in V^{CR}$.

$Type(A) \in \{strict, defeasible\}$ *indica se o argumento é **estrito** ou **derrotável**. Um argumento estrito é tal que o próprio argumento é formado a partir de uma regra estrita, e todo subargumento também é estrito. Um argumento derrotável é baseado em uma regra derrotável ou algum de seus subargumentos é derrotável. Formalmente,* $Type(A) = strict$ *sse* $Type(Rule(A)) = strict$ *e* $\forall B \in SA(A), Type(Rule(B)) = strict$; *e* $Type(A) = defeasible$ *sse* $Type(Rule(A)) = defeasible$ *ou* $\exists B \in SA(A)$ *t.q.* $Type(Rule(B)) = defeasible$.

Definição 8 (Nó-Folha Falacioso como Argumento)**.** *Um nó-folha falacioso é um argumento da forma* $q! \in Args$, *tal que* $q \in V^{CR}$. *Um nó-folha falacioso apresenta as seguintes propriedades:*

- $Rule(A) = \perp$, *pois podem existir várias regras com cabeça* q
- $Conc(A) = q$
- $ArgD(A) = D(q)$
- $SA(A) = ISA(A) = DISA(A) = Prem(A) = IRLs(A) = DIRLs(A) = \emptyset$
- $Fall(A) = true$

Convenciona-se nomear um argumento a partir do agente que o define, e o foco de consulta também é utilizado como sobrescrito dada a possibilidade de haver múltiplos focos de consulta. Por exemplo, se a conclusão de um argumento é definida por um agente a em um foco de consulta α, então o argumento será chamado de A_i^α, com $i \in \mathbb{N}^+$. Se for um argumento definido por um agente b em um foco de consulta β, será chamado B_i^β. Quando se referencia explicitamente um sistema $\mathcal{S}$ e um foco de consulta α, denota-se o conjunto de argumentos como $Args_{\mathcal{S}\alpha}$, similarmente ao que é feito no caso de $KB_{\mathcal{S}\alpha}$.

Uma representação gráfica é demonstrada nos exemplos dados a seguir (ver Figuras 9, 10 e 12). Optou-se por utilizar arestas direcionadas na árvore, no sentido da conclusão do argumento, visto representar de maneira mais natural o conceito de derivação e a correspondência dos argumentos com as regras nos quais são baseados. Também são diferenciadas arestas com linha duplicada para argumentos baseados em regras derrotáveis, e arestas com linha simples para argumentos baseados em regras estritas. A representação "em linha" segue uma notação similar à notação de parênteses para árvores, porém com e uma flecha $\leftharpoondown$ indicando a separação entre conclusão e premissas: Um argumento é denotado $(...)$. Um argumento base com conclusão p é $(p \leftharpoondown \top)$. Um argumento com conclusão p com dois subargumentos é $(p \leftharpoondown (...), (...))$.

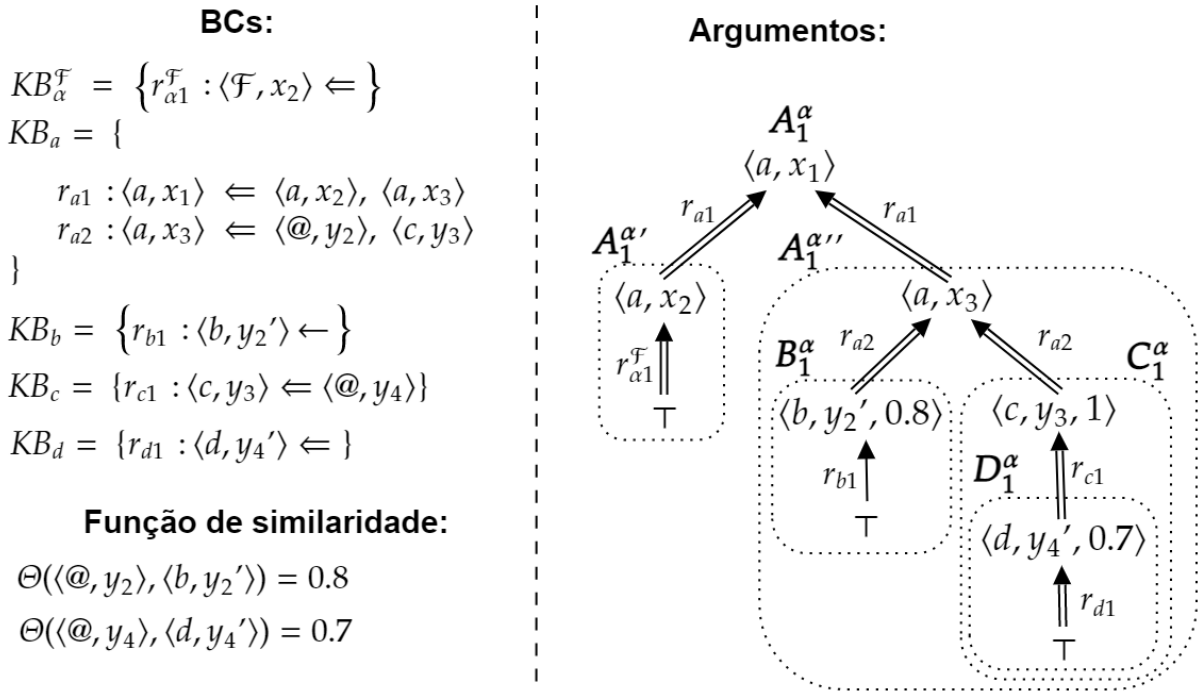
Exemplo 2. A Figura 9 ilustra um argumento que contempla todos os casos da Definição 7, exceto a existência de argumentos falaciosos, que são ilustrados no Exemplo 3. Nela, há 4 (quatro) agentes, a, b, c e d, e suas respectivas bases de conhecimento KB_a, KB_b, KB_c e KB_d. A BC de foco de um foco de consulta α qualquer, denotado $KB_\alpha^\mathcal{F}$, contém uma única regra de foco $r_{\alpha 1}^\mathcal{F}$. Logo, os argumentos são gerados a partir da base de conhecimento estendida $KB_{\mathcal{S}\alpha} = KB_a \cup KB_b \cup KB_c \cup KB_d \cup KB_\alpha^\mathcal{F}$.

A partir da regra $r_{a1} \in KB_a$, é possível derivar o argumento A_1^α. Para o primeiro antecedente da regra, $\langle a, x_2\rangle$, é possível derivar um subargumento $A_1^{\alpha\prime}$ com conclusão $\langle a, x_2\rangle$, baseado na regra de foco $r_{\alpha 1}^\mathcal{F}$ (caso I da Definição 7). Este, por sua vez, é um argumento base, visto ter corpo vazio. Para o segundo antecedente da regra, $\langle a, x_3\rangle$, é possível derivar um subargumento $A_1^{\alpha\prime\prime}$ com conclusão $\langle a, x_3\rangle$, baseado na regra r_{a2} (caso I da Definição 7).

A regra r_{a2} possui dois antecedentes. Para o primeiro, $\langle @, y_2\rangle$, é possível derivar um subargumento B_1^α com conclusão $\langle b, y_2'$,0.8$\rangle$[3], baseado na regra r_{b1}, dado que $\Theta(\langle @, y_2\rangle, \langle b, y_2'\rangle) =$ 0,8 (caso III da Definição 7). Este, por sua vez, é um argumento base, visto que r_{b1} tem corpo

[3] O padrão numérico para números decimais $X.Y$ (onde X é a parte inteira e Y é a parte fracionária do número) será utilizado em vez de X,Y quando estiver dentro de uma tupla, visto que a representação de tuplas também usa vírgulas.

Figura 9 – Ilustração da concepção de argumentos do Exemplo 2.



Fonte: Autoria própria.

vazio. Para o segundo antecedente de r_{a2}, $\langle c, y_3 \rangle$, é possível derivar um subargumento C_1^α com conclusão $\langle c, y_3, 1 \rangle$, baseado na regra r_{c1} (caso II da Definição 7). A regra r_{c1} possui apenas um antecedente, $\langle @, y_4 \rangle$, para o qual é possível derivar um subargumento D_1^α com conclusão $\langle d, y_4', 0.7 \rangle$, baseado na regra r_{d1}, dado que $\Theta(\langle @, y_2 \rangle, \langle b, y_2' \rangle) = 0{,}7$ (caso III da Definição 7).

Apenas B_1^α é um argumento estrito.

Representações "em linha":

- $A_1^\alpha = (\langle a, x_1 \rangle \leftharpoonup A_1^{\alpha\prime}, A_1^{\alpha\prime\prime})$
- $A_1^{\alpha\prime} = (\langle a, x_2 \rangle \leftharpoonup \top)$
- $A_1^{\alpha\prime\prime} = (\langle a, x_3 \rangle \leftharpoonup B_1^\alpha, C_1^\alpha)$
- $B_1^\alpha = (\langle b, y_2', 0.8 \rangle \leftharpoonup \top)$
- $C_1^\alpha = (\langle c, y_3, 1 \rangle \leftharpoonup D_1^\alpha)$
- $D_1^\alpha = (\langle d, y_4', 0.7 \rangle \leftharpoonup \top)$

Quanto às funções, os seguintes valores para A_1^α podem ser definidos:

- $Rule(A_1^\alpha) = r_{a1}$

- $Conc(A_1^\alpha) = \langle a, x_1 \rangle$
- $ArgD(A_1^\alpha) = a$
- $SA(A_1^\alpha) = \{A_1^{\alpha\prime}, A_1^{\alpha\prime\prime}, B_1^\alpha, C_1^\alpha, D_1^\alpha\}$
- $ISA(A_1^\alpha) = \{B_1^\alpha, C_1^\alpha, D_1^\alpha\}$
- $DISA(A_1^\alpha) = \{B_1^\alpha, C_1^\alpha\}$
- $Prem(A_1^\alpha) = \{\langle a, x_2 \rangle, \langle a, x_3 \rangle, \langle b, y_2', 0.8 \rangle, \langle c, y_3, 1 \rangle, \langle d, y_4', 0.7 \rangle\}$
- $IRLs = \{\langle b, y_2', 0.8 \rangle, \langle c, y_3, 1 \rangle, \langle d, y_4', 0.7 \rangle\}$
- $DIRLs = \{\langle b, y_2', 0.8 \rangle, \langle c, y_3, 1 \rangle\}$
- $Fall(A_1^\alpha) = false$
- $Type(A_1^\alpha) = defeasible$

□

Continuação do Exemplo 1.1. (Veja o Exemplo 1 na página 68 e o Exemplo 1.1 na página 72) A Figura 10 mostra os argumentos derivados que concluem $\langle a, col(m_1) \rangle$ e $\langle a, \neg col(m_1) \rangle$. Além desses, também podem ser derivados os argumentos baseados nas regras r_{b2} e r_{b3}, mas estes serão omitidos por serem idênticos aos argumentos A_1^α, A_2^α e A_3^α, com a única diferença de terem como conclusões $\langle b, col(m_1) \rangle$ e $\langle b, \neg col(m_1) \rangle$. Os rótulos das regras nas arestas também foram omitidos, uma vez que é fácil associar quais argumentos foram derivados de quais regras, como apresentadas na Figura 1.

Observe que, a partir da regra esquemática r_{a2} o argumento A_1^α é formado pela instanciação do *er-literal* $\langle @, ed(m_1) \rangle$ com a conclusão de C_1^α. Da mesma forma, A_2^α e A_3^α são formados pela regra r_{a3}, tendo seus *er-literais* instanciados com as conclusões dos subargumentos B_1^α e D_1^α, respectivamente. Os argumentos B_1^α e E_1^α são simplesmente originados de suas regras locais juntamente com as regras de foco.

O argumento C_1^α é baseado na regra r_{c1} e formado com base na instanciação do *er-literal* $\langle @, avl(m_1) \rangle$ com a cabeça da regra r_{e1}, a partir do qual é criado o subargumento E_1^α. O *ir-literal* $\langle e, spa(m_1), 0.8 \rangle$, neste caso, possui grau de similaridade $0{,}8$. O mesmo ocorre com D_1^α, que é baseado na regra r_{d1} e formado com base na instanciação do *er-literal* $\langle @, am(m_1) \rangle$

com a conclusão do argumento E_1^{α}, porém com grau de similaridade $0{,}4$, gerando-se o *ir-literal* $\langle e, spa(m_1), 0.4\rangle$.

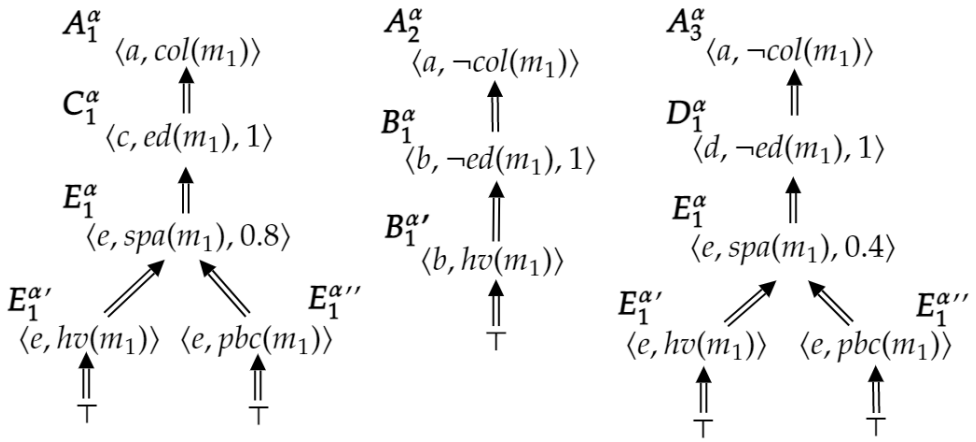
Observe que os *fr-literais* originados das regras de foco recebidos na consulta são considerados localizados em cada agente – por exemplo, o *fr-literal* $\langle \mathcal{F}, hv(m_1)\rangle$ no escopo do agente b torna-se a conclusão $\langle b, hv(m_1)\rangle$ de $B_1^{\alpha\prime}$; e ambos $\langle \mathcal{F}, hv(m_1)\rangle$ e $\langle \mathcal{F}, pbc(m_1)\rangle$ em E_1^{α} no escopo do agente e tornam-se as conclusões $\langle e, hv(m_1)\rangle$ e $\langle e, pbc(m_1)\rangle$ de $E_1^{\alpha\prime}$ e $E_1^{\alpha\prime\prime}$, respectivamente. Importa notar, portanto, que sem o compartilhamento do conhecimento de foco da consulta α, nenhum desses argumentos poderia ser derivado, visto que b e e não estariam cientes do cogumelo m_1 e suas características.

□

Continuação do Exemplo 1.2. (Veja o Exemplo 1 na página 68 e o Exemplo 1.2 na página 72) A partir do foco de consulta β, os argumentos gerados são mostrados na Figura 11.

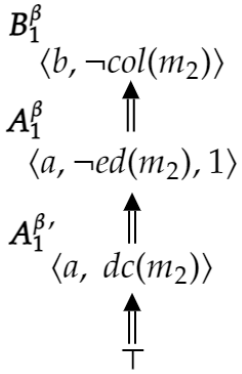
Neste caso, o argumento B_1^{β} é formado a partir da regra r_{b1} instanciando-se o *er-literal*

Figura 10 – Conjunto de argumentos $Args_{\mathcal{S}\alpha}$ para o Exemplo 1.1.



Fonte: Autoria própria.

Figura 11 – Conjunto de argumentos $Args_{\mathcal{S}\beta}$ para o Exemplo 1.2.



Fonte: Autoria própria.

$(@,\neg ed(m1))$ com a conclusão do argumento A_1^β, que por sua vez é baseado na regra r_{a1}. Mais uma vez, isso só foi possível pelo fato da regra $r_{\beta 1}^{\mathcal{F}}$ ter sido compartilhada junto ao foco de consulta β, o que faz com que o agente a fique ciente de que m_2 se trata de um *death cap*.

É interessante notar que esse conjunto de argumentos é diferente dos argumentos gerados para a consulta α e que ambas as consultas não interferem uma na outra, visto que os argumentos gerados em ambas as consultas são independentes entre si. Isso exemplifica como diferentes focos levam a diferentes conjuntos de argumentos.

□

Continuação do Exemplo 1.3. (Veja o Exemplo 1 na página 68 e o Exemplo 1.3 na página 73) O conjunto de argumentos para o foco de consulta γ é simplesmente um subconjunto de $Args_{\mathcal{S}\alpha}$, contendo apenas o argumento equivalente a A_2^α e seus subargumentos. O argumento equivalente a A_1^α não é incluído porque depende da existência de C_1^α, que por sua vez depende da existência de E_1^α. Da mesma forma, o argumento equivalente a A_3^α não é incluído porque depende da existência de D_1^α, que por sua vez depende da existência de E_1^α.

□

Exemplo 3. Este é um exemplo abrangente com o objetivo de mostrar as possibilidades de derivação de argumentos, incluindo argumentos falaciosos e múltiplos argumentos baseados na mesma regra. A seguir são apresentadas as bases de conhecimento de cada agente.

$$\mathbf{KB_a} = \{r_{a1} : (a,x_1) \Leftarrow (@,x_2), (@,x_3);\ r_{a2} : (a,\neg x_1) \Leftarrow (b,x_4)\}$$

$$\mathbf{KB_b} = \{r_{b1} : (b,x_2) \leftarrow (b,y_1);\ r_{b2} : (b,x_4) \Leftarrow\}$$

$$\mathbf{KB_c} = \{r_{c1} : \langle c,x_3\rangle \Leftarrow \langle @,y_2\rangle;\ r_{c2} : \langle c,x_3\rangle \Leftarrow \langle @,y_3\rangle;\ r_{c3} : \langle c,x_3\rangle \Leftarrow \langle @,z_1\rangle;\ r_{c4} : \langle c,\neg x_3\rangle \Leftarrow \langle b,x_4\rangle\}$$

$$\mathbf{KB_d} = \{r_{d1} : \langle d,y_2'\rangle \Leftarrow \langle @,x_1\rangle;\ r_{d2} : \langle d,y_2''\rangle \Leftarrow;\ r_{d3} : \langle d,y_2'''\rangle \Leftarrow;\ r_{d4} : \langle d,y_3\rangle \Leftarrow \langle @,\neg x_1\rangle;\ r_{d5} : \langle d,x_3\rangle \Leftarrow\}$$

Considere também um foco de consulta $\delta = \{\langle a, x_1\rangle, a, KB_\delta^{\mathcal{F}}\}$ tal que $KB_\delta^{\mathcal{F}} = \{r_{\delta 1}^{\mathcal{F}} : \langle \mathcal{F}, z_1\rangle \Leftarrow\}$. A Figura 12 mostra os argumentos gerados.

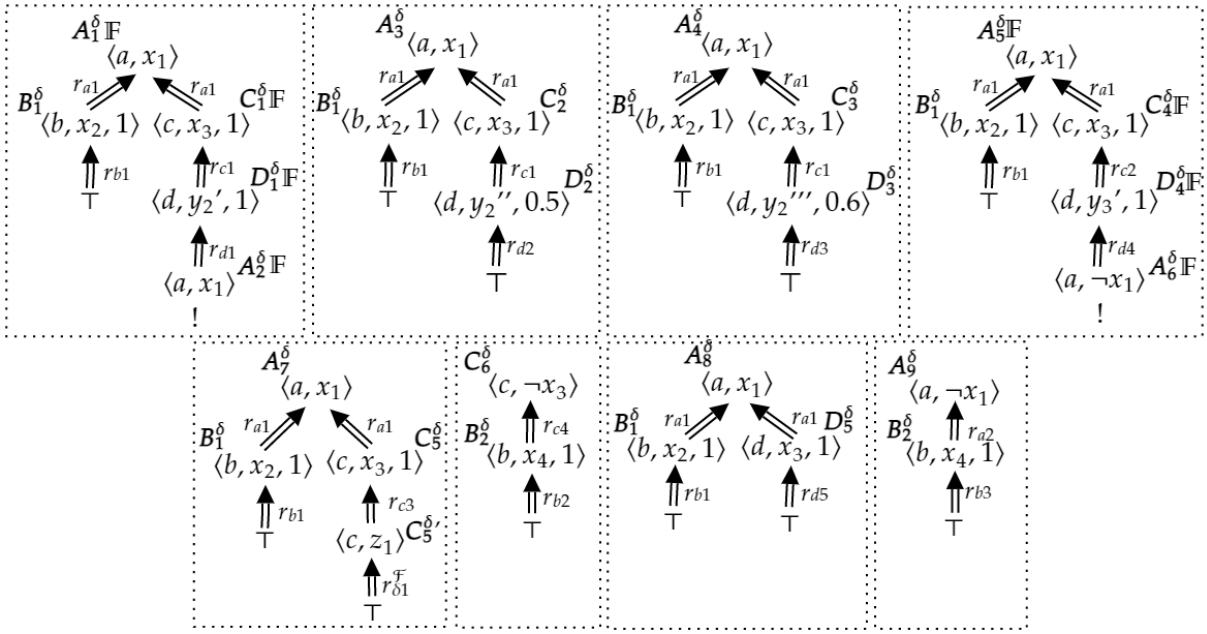
Os argumentos A_1^δ a A_8^δ (exceto A_6^δ) são todos derivados para apoiar a conclusão $\langle a, x_1\rangle$

e apenas A_6^δ e A_9^δ concluem $\langle a, \neg x_1 \rangle$.$A_1^\delta$ é baseado na regra r_{a1}, com subargumentos baseados em r_{b1} (B_1^α), r_{c1} (C_1^α) e r_{d1} (D_1^α). O argumento D_1^α, neste caso, acaba levando à geração de um nó-folha falacioso A_2^δ, visto depender do *r-literal* $\langle a, x_1 \rangle$, que já é conclusão de A_1^δ, caracterizando uma falácia de argumentação circular. Algo semelhante acontece na derivação do argumento A_4^δ: o argumento D_4^δ é criado, requerendo o *r-literal* $\langle a, \neg x_1 \rangle$, sendo que a conclusão contrária $\langle a, x_1 \rangle$ já aparece em A_4^δ. Isso é detectado como um argumento autodestrutivo, e, portanto, um nó-folha falacioso A_6^δ é criado. Portanto, todos os argumentos cujas conclusões são nós ancestrais nas árvores dos argumentos A_2^δ e A_6^δ são marcados como falaciosos, representado na figura com o símbolo $\mathbb{F}$.

O argumento A_2^δ difere de A_3^δ no sentido de que, no primeiro, $\langle @, y_2 \rangle$ está vinculado a $\langle d, y_2'' \rangle$ da regra r_{d2}, com grau de similaridade $0{,}5$, e no segundo $\langle @, y_2 \rangle$ está vinculado a $\langle d, y_2''' \rangle$ da regra r_{d3}, com grau de similaridade $0{,}6$. Observe que ambos os argumentos têm estruturas muito semelhantes, mudando-se apenas os *ir-literais* usados para instanciar o antecedente da regra r_{c1}, que possuem diferentes graus de similaridade.

No argumento A_7^δ, que é baseado na regra r_{a1}, o antecedente $\langle @, x_3 \rangle$ é vinculado a $\langle c, x_3 \rangle$ com base na regra r_{c3}, cujo antecedente $\langle @, z_1 \rangle$ é vinculado à regra de foco $r_{\delta 1}^{\mathcal{F}}$. Isso exemplifica o impacto do conhecimento de foco no raciocínio, uma vez que o argumento A_7^δ não existiria se $r_{\delta 1}^{\mathcal{F}}$ não fosse conhecido pelo agente c.

Figura 12 – Estruturas de argumentação criadas para o Exemplo 3.



Fonte: Autoria própria.

O argumento A_8^δ difere de A_7^δ no sentido de que $\langle @, x_3 \rangle$ está vinculado a $\langle d, x_3 \rangle$ da regra r_{d4}. Observe que ambos os argumentos têm estruturas muito semelhantes, mas A_7^δ é baseado no conhecimento vindo de a, b e c, enquanto A_8^δ é baseado no conhecimento de a, b e d.

O argumento C_6^δ é criado a partir das regras r_{c4} e r_{b2}. Como será apresentado mais adiante, esta é uma regra que seria capaz de realizar um *undercut* em A_1^δ, A_3^δ, A_4^δ, A_5^δ e A_7^δ, caso seja mais forte que estas.

Finalmente, o único argumento válido (i.e., não falacioso) para $\langle a, \neg x_1 \rangle$ é A_9^δ, baseado nas regras r_{a2} e r_{b2}.

□

3.3 SEMÂNTICA DE ACEITABILIDADE DE ARGUMENTOS

Uma vez que o conjunto de argumentos $Args$ tenha sido definido, é necessária uma maneira de determinar quais argumentos (e consequentemente quais *r-literais*) são aceitos como consequências lógicas de uma base de conhecimento KB. Uma semântica de argumentação visa definir quais argumentos são justificados e quais argumentos são rejeitados.

A semântica apresentada aqui é baseada na semântica com bloqueio de ambiguidade sem relação de superioridade entre regras da DL, conforme apresentado na Seção 2.2.2.1. Portanto, parte das definições são similares às de Governatori *et al.* (2004) e Bikakis e Antoniou (2010), com exceção da definição de *derrota*, da função de cálculo de força de argumentos e da função de comparação de forças de argumentos. A relação de superioridade entre regras não será utilizada, pois a resolução de conflitos proposta será baseada no cálculo de força de argumentos. No entanto, como demonstrado por Antoniou *et al.* (2001), a relação de superioridade entre regras pode ser simulada pelos outros ingredientes da DL, não sendo um requisito para uma lógica baseada em DL. Também são apresentadas variantes do cálculo de força representando diferentes atitudes dos agentes no que diz respeito à consideração dos argumentos gerados por outros agentes, além de uma versão do cálculo de força que considera a soma das forças de todos os argumentos que concluem o mesmo *r-literal*.

Esta seção é subdivida em 4 (quatro) subseções para permitir uma apresentação mais organizada das definições. Na Seção 3.3.1 são apresentadas as definições de ataque e derrota entre argumentos, juntamente com o cálculo e comparação de forças proposta. Em seguida, na Seção 3.3.2 são apresentados os conceitos de apoio e *undercut*. Na Seção 3.3.3 são apresentadas as definições de argumentos aceitáveis, assim como a construção dos conjuntos de argumentos

justificados e rejeitados. Por fim, na Seção 3.3.4 são apresentadas formas alternativas de cálculo e comparação de forças de argumentos.

3.3.1 Ataque, Derrota e Força dos Argumentos

A derivação das consequências lógicas locais de uma base de conhecimento é baseada, primeiramente, em seus argumentos estritos. De fato, as conclusões de todos os argumentos estritos em $Args$ são consequências lógicas de KB. A derivação de consequências lógicas não estritas é baseada em argumentos derrotáveis (que podem ser locais ou distribuídos). Nesse caso, deve-se também considerar os conflitos entre argumentos com conclusões contraditórias. As definições de *ataque* e *derrota* que se seguem representam a ideia de que, entre dois argumentos derrotáveis conflitantes, um argumento derrota o outro sempre que ele não é mais fraco que o outro, seguindo a interpretação de que "derrotar" significa "atacar e não ser mais fraco" (PRAKKEN; VREESWIJK, 2001), o que possibilita derrota mútua entre argumentos.

Definição 9 (Ataque entre Argumentos)**.** *Um argumento A_i ataca um argumento derrotável (local ou distribuído) A_j, denotado $A_i \mathcal{A} A_j$, sse suas conclusões são complementares. Formalmente, seja uma relação $\mathcal{A} \subset Args \times Args$: $A_i \mathcal{A} A_j$ sse $Conc(A_i) = p$ e $Conc(A_j) = \sim p$.*

Definição 10 (Derrota entre Argumentos)**.** *Um argumento A_i derrota um argumento derrotável (local ou distribuído) A_j, denotado $A_i \mathcal{D} A_j$ sse A_i ataca A_j, e A_j não é mais forte que A_i. Formalmente, seja uma relação $\mathcal{D} \subset Args \times Args$ e uma função de comparação de forças $Stronger : Args \times Args \rightarrow Args$: $A_i \mathcal{D} A_j$ sse $A_i \mathcal{A} A_j$ e $Stronger(A_i, A_j) \neq A_j$.*

Vale notar que argumentos que se atacam/derrotam sempre são definidos pelo mesmo agente, por isso a utilização dos símbolos A_i e A_j nas definições. Argumentos definidos por diferentes agentes nunca se atacam, pois o agente definidor da conclusão de um será diferente do definidor da conclusão do outro. Desse modo, a resolução de conflitos sempre é realizada localmente, porém considerando a força dos subargumentos (que podem ser definidos por outros agentes) dos argumentos conflitantes – o que vai impactar no cálculo de força desses argumentos, como apresentado mais adiante, e, consequentemente, na função $Stronger$ e na relação de derrota.

A função de comparação de forças $Stronger$ pode ser definida como um parâmetro da aplicação. Neste trabalho, é apresentada uma função $Stronger$ padrão baseada na comparação de um valor numérico de força de dois argumentos. Portanto, antes de se definir a função $Stronger$,

é necessário definir a função de cálculo de força de argumentos ($StArg$), que, por sua vez, depende da definição da função de força de *ir-literais* ($StIRL$), apresentada na Definição 11.

Definição 11 (Força de um *ir-literal*)**.** *A força de um ir-literal $q_{inst} = (b, x, \theta)$ do ponto de vista do agente a é $StIRL(q_{inst}, a) = P_a(b) \times \theta$.*

A força de um *ir-literal* é fundamental, pois é o que define o elemento básico do cálculo de força de um argumento. Intuitivamente, pode-se dizer que as forças dos *ir-literais* representam as forças dos elos (ou vinculações) entre um argumento e seus subargumentos instanciados – os subargumentos definidos por diferentes agentes e/ou que possuem grau de similaridade agregado.

É importante mencionar também que tal função recebe por parâmetro, além do *ir-literal*, o agente por cujo ponto de vista essa força será calculada, considerando sua função de confiança individual. Isto é necessário pois, no cálculo de força padrão, a força de cada *ir-literal* é calculada do ponto de vista do agente que define o argumento que tem o *ir-literal* como nó-filho, o que intuitivamente representa o agente que define a regra em cujo corpo o *r-literal* correspondente ao *ir-literal* é encontrado. Isto é interessante pois a força de um argumento distribuído é, desse modo, calculada considerando os pontos de vista dos diferentes agentes que colaboram para a derivação do argumento, de modo a obter uma abordagem mais social de cálculo de força dos argumentos, similarmente ao apresentado em (PANISSON *et al.*, 2016).

Por exemplo, no Exemplo 2, cujos argumentos são ilustrados na Figura 9, para o *ir-literal* $\langle b, y'_2, 0.8\rangle$, o agente que será utilizado para o cálculo de sua força é o a, que define o nó ancestral imediato $\langle a,x_3\rangle$, que é também a conclusão do argumento $A_1^{\alpha\prime\prime}$, que é baseado na regra $r_{a2} \in KB_a$. Por outro lado, para o *ir-literal* $\langle d, y'_4, 0.7\rangle$, o agente c será utilizado, pois este define o nó ancestral imediato $\langle c,y_3,1\rangle$, que é também a conclusão do argumento C_1^{α}, que é baseado na regra $r_{c1} \in KB_c$.

A Definição 12 apresenta a função de cálculo de força padrão.

Definição 12 (Força de um Argumento)**.** $StArg : Args \rightarrow [0,1]$ *é uma função que recebe um argumento A e retorna um valor entre 0 e 1, onde 0 representa o menor valor possível de força e 1 o maior valor possível de força, da seguinte maneira:*

$$StArg(A) = \begin{cases} 1 & \text{se } DISA(A) = \emptyset \\ \dfrac{\sum\limits_{A' \in DISA(A)} StIRL(Conc(A'),ArgD(A)) \times StArg(A')}{|DISA(A)|} & \text{se } DISA(A) \neq \emptyset \end{cases}$$

Um argumento local (quando $DISA(A) = \emptyset$), do ponto de vista do próprio agente que define seu argumento, sempre terá $StArg(A) = 1$, dado que o argumento que ele contém é autossuficiente e pressupõe-se que o agente confia totalmente no seu conhecimento local. Caso contrário, a força de um argumento A é o somatório da força de cada *ir-literal* direto do argumento (isto é, de cada conclusão $Conc(A')$ de cada subargumento instanciado direto A') do ponto de vista do agente que define o argumento ($ArgD(A)$), multiplicado pela força de cada subargumento instanciado direto A' (capturado pela função $DISA(A)$), dividido pelo número de subargumentos instanciados diretos $|DISA(A)|$. A intuição é que a força de um argumento é dependente das forças dos subargumentos instanciados – isto é, dos subargumentos definidos por outros agentes e/ou que possuem conclusão com grau de similaridade agregado – dos quais ele depende diretamente, o que, mais uma vez, remete ao fato de que são considerados os pontos de vista dos diferentes agentes envolvidos, seguindo uma abordagem social de cálculo de força.

Uma característica interessante dessa fórmula é "enfraquecer" um argumento apoiado com base na confiança indireta entre os agentes, similarmente à abordagem cética de cálculo de confiança indireta apresentada na Seção 2.4. Isto ocorre porque o cálculo depende recursivamente da força dos subargumentos instanciados diretos, induzindo à multiplicação das forças dos *ir-literais* que são conclusões desses subargumentos. Como o contradomínio das funções de cálculo de força de *ir-literais* e de argumentos é o intervalo entre 0 e 1, segue então que os valores de força correspondentes a essas dependências indiretas, ao serem multiplicados, resultam em valores reduzidos quando forem menores que 1. Isto será demonstrado especialmente por meio do Exemplo 4.

A função de comparação de forças de argumentos é apresentada a seguir.

Definição 13 (Função de Comparação de Forças de Argumentos)**.** *Seja um argumento $A_i \in Args$ e um argumento $A_j \in Args$. A função $Stronger : Args \times Args \rightarrow Args$ é tal que:*

- $Stronger(A_i, A_j) = A_i$ *sse* $StArg(A_i) > StArg(A_j)$*, ou*
- $Stronger(A_i, A_j) = A_j$ *sse* $StArg(A_i) < StArg(A_j)$*, ou*
- $Stronger(A_i, A_j) = \perp$ *sse* $StArg(A_i) = StArg(A_j)$*;*

Continuação do Exemplo 1.1. (Ver Exemplo 1 na página 68, primeira parte do Exemplo 1.1 na página 72 e segunda parte do Exemplo 1.1 na página 80). Conforme ilustrado na Figura 10, existem alguns argumentos derrotáveis com conclusões contraditórias: A_1^{α} vs. A_2^{α} e A_3^{α}. Isso

significa que eles atacam uns aos outros: A_1^α ataca A_2^α e A_3^α e vice-versa. Pode-se, portanto, calcular as forças desses argumentos e concluir que A_1^α é mais forte do que A_2^α e A_3^α, derrotando assim a ambos, enquanto A_2^α e A_3^α não derrotam A_1^α.

$$StArg(A_1^\alpha) = \frac{StIRL(\langle c, ed(m_1),1\rangle,a) \times StArg(C_1^\alpha)}{1}$$
$$= P_a(c) \times 1 \times \frac{StIRL(\langle e, spa(m_1),0.8\rangle,c) \times StArg(E_1^\alpha)}{1}$$
$$= 0{,}6 \times 1 \times P_c(e) \times 0{,}8 \times 1 = 0{,}6 \times 1 \times 1 \times 0{,}8 = 0{,}48$$

$$StArg(A_2^\alpha) = \frac{StIRL(\langle b, \neg ed(m_1), 1\rangle,a) \times StArg(B_1^\alpha)}{1} = P_a(b) \times 1 = 0{,}4$$

$$StArg(A_3^\alpha) = \frac{StIRL(\langle d, \neg ed(m_1), 1\rangle,a) \times StArg(D_1^\alpha)}{1}$$
$$= P_a(d) \times 1 \times \frac{StIRL(\langle e, spa(m_1), 0.4\rangle,d) \times StArg(E_1^\alpha)}{1}$$
$$= 0{,}2 \times 1 \times P_d(e) \times 0{,}4 \times 1 = 0{,}2 \times 1 \times 1 \times 0{,}4 = 0{,}08$$

É interessante notar, neste caso, que embora A_1^α tenha sido calculada como mais forte, ainda assim tenha ficado com um valor de força relativamente baixo devido à dependência indireta de argumentos definidos por outros agentes, pois foi necessário multiplicar a confiança de Alice em Charles, de $0{,}6$, e a similaridade entre amanita da primavera ($spa(m_1)$) e *amanita velosa* ($avl(m_1)$), de $0{,}8$. A confiança de valor 1 de Charles em Eric evitou que a força fosse ainda menor. Em contraste, no caso de A_2^α, embora a confiança de Alice em Barb seja menor, de $0{,}4$, o argumento tem força não muito inferior a A_1^α, visto depender apenas de uma dependência direta de um subargumentos definido por outro agente.

Logo, $Stronger(A_1^\alpha,A_2^\alpha) = A_1^\alpha$ e $Stronger(A_1^\alpha,A_3^\alpha) = A_1^\alpha$.

□

Continuação do Exemplo 3. (Ver primeira parte do Exemplo 3 na página 82). Neste exemplo não são realizados os cálculos, uma vez que o exemplo em questão é muito grande e demandaria um esforço e espaço maior para sua realização. Portanto, são feitas apenas algumas comparações entre alguns dos argumentos de modo a se perceber qual seria mais forte e por qual motivo.

Os argumentos A_3^δ e A_4^δ possuem estruturas muito semelhantes, mudando-se apenas o *ir-literal* que instancia o *er-literal* $\langle @,y_2\rangle$. Neste caso, A_4^δ será mais forte que A_3^δ, visto que

ambos dependem exatamente dos mesmos agentes, de modo que a função de confiança não causa diferença entre suas forças, porém A_4^δ possui um *ir-literal* com maior grau de similaridade do que o *ir-literal* de A_3^δ. Este é um exemplo de como a vinculação baseada na similaridade entre *r-literais* de diferentes agentes afeta a força dos argumentos.

Os argumentos A_7^δ e A_8^δ têm estruturas muito semelhantes, mas A_7^δ é baseado no conhecimento vindo de a, b e c, enquanto A_8^δ é baseado no conhecimento de a, b e d. Portanto, a diferença de força entre eles será baseada apenas na função de confiança de a, visto não haver diferenças relacionadas à similaridade de *ir-literais*. Se $P_a(c) > P_a(d)$, então A_7^δ será mais forte que A_8^δ, e vice-versa.

Quanto às relações de ataque e derrota, é fácil observar que todos os argumentos de A_1^δ a A_8^δ (exceto o A_6^δ) atacam A_9^δ, e vice-versa. Se a força de A_9^δ for maior ou igual que de todos os demais, então A_9^δ derrota a todos. Como um operador de comparação aritmético ($>$) é utilizado para definir a função $Stronger$, segue que A_9^δ derrota a todos os demais se for tão ou mais forte que o argumento mais forte dentre os demais. Logo, é interessante notar que, dados vários argumentos para a mesma conclusão, o argumento mais forte é definitivo quanto à conclusão ser ou não derrotada.

Por fim, C_6^δ ataca a todos os argumentos de C_1^δ a C_5^δ, sendo as relações de derrota dependentes da força de cada argumento.

□

Exemplo 4. Este exemplo, também genérico, similarmente ao Exemplo 3, primeiramente visa demonstrar como o cálculo de força é afetado pela confiança indireta em relação a argumentos definidos por outros agentes. Mais adiante, o mesmo exemplo será utilizado para ilustrar com maior clareza as relações de apoio e *undercut*, e também a construção do conjunto de argumentos justificados.

Considere um DDRMAS $\mathcal{S}$ com 6 (seis) agentes com as seguintes bases de conhecimento:

$$\mathbf{KB_a} = \{r_{a1} : \langle a, x_1\rangle \Leftarrow \langle a, y_1\rangle;\ r_{a2} : \langle a, y_1\rangle \Leftarrow \langle b, y_2\rangle;\ r_{a3} : \langle a, \neg y_1\rangle \Leftarrow \langle @, y_3\rangle, \langle @, y_4\rangle\}$$

$$\mathbf{KB_b} = \{r_{b1} : \langle b, y_2\rangle \Leftarrow \langle e, y_5\rangle;\ r_{b2} : \langle b, \neg y_2\rangle \Leftarrow \langle f, y_6\rangle\}$$

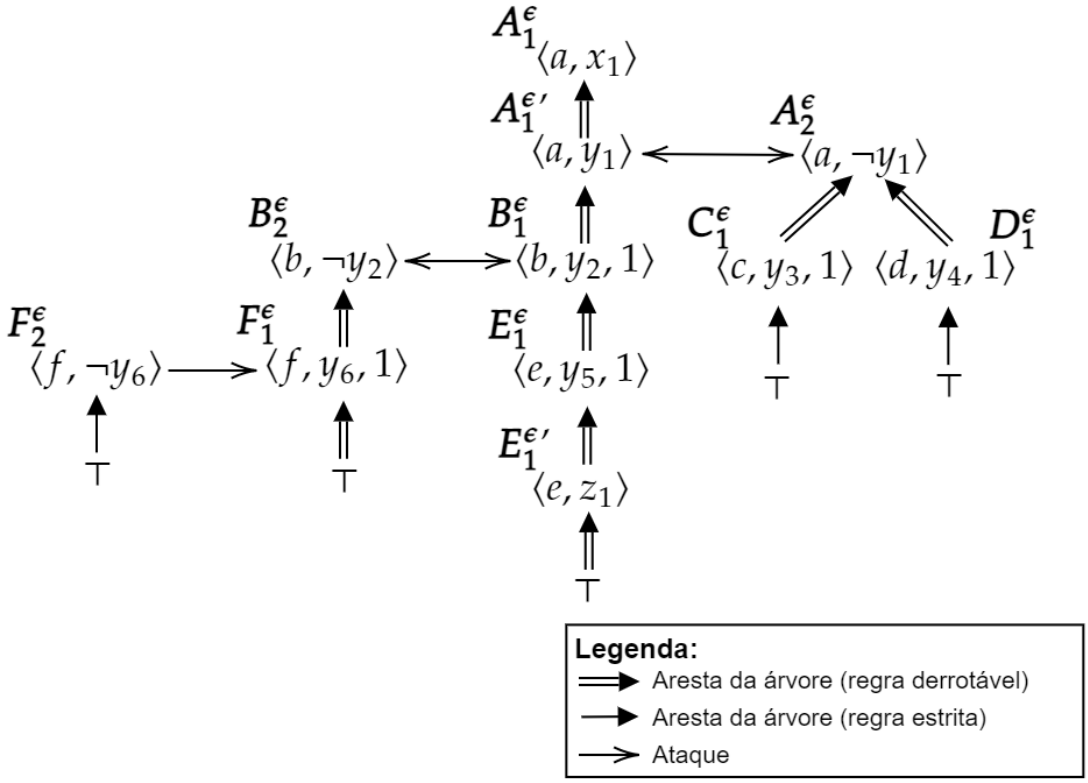
$\mathbf{KB_c} = \{r_{c1} : \langle c, y_3\rangle \leftarrow\}$ $\mathbf{KB_f} = \{r_{f1} : \langle f, y_6\rangle \Leftarrow;$

$\mathbf{KB_d} = \{r_{d1} : \langle d, y_4\rangle \leftarrow\}$ $r_{f2} : \langle f, \neg y_6\rangle \leftarrow$

$\mathbf{KB_e} = \{r_{e1} : \langle e, y_5\rangle \Leftarrow \langle e, z_1\rangle\}$ $\}$

Considerando que $\epsilon = (\langle a, x_1\rangle, a, KB_\epsilon^{\mathcal{F}})$ e $KB_\epsilon^{\mathcal{F}} = \{r_{\epsilon 1}^{\mathcal{F}} : \langle \mathcal{F}, z_1\rangle \Leftarrow\}$, os argumentos e subargumentos que podem ser formados a partir de $\mathcal{S}$ e do foco de consulta ϵ, que constituem o conjunto de argumentos $Args_{\mathcal{S}\epsilon}$, são mostrados na Figura 13.

Figura 13 – Argumentos do Exemplo 4.



Fonte: Autoria própria.

Seja $P_a(b) = P_a(c) = P_a(d) = P_b(e) = P_b(f) = 0{,}8$. A seguir é apresentado o cálculo de força dos argumentos:

$$StArg(B_1^\epsilon) = \frac{0{,}8 \times 1 \times StArg(E_1^\epsilon)}{1} = 0{,}8 \times 1 \times 1 = 0{,}8$$

$$StArg(B_2^\epsilon) = \frac{0{,}8 \times 1 \times StArg(F_1^\epsilon)}{1} = 0{,}8 \times 1 \times 1 = 0{,}8$$

$$StArg(A_1^\epsilon) = \frac{0{,}8 \times 1 \times StArg(B_1^\epsilon)}{1} = 0{,}8 \times 0{,}8 = 0{,}64$$

$$StArg(A_2^\epsilon) = \frac{0{,}8 \times 1 \times StArg(C_1^\epsilon) + 0{,}8 \times 1 \times StArg(D_1^\epsilon)}{2} = \frac{0{,}8 \times 1 + 0{,}8 \times 1}{2} = 0{,}8$$

A diferença entre $StArg(A_1^\epsilon)$ e $StArg(A_2^\epsilon)$ demonstra o impacto das dependências indiretas entre diferentes agentes no valor da força. Argumentos que possuem apenas dependência direta de argumentos definidos por outros agentes tendem a ser mais fortes que argumentos que possuem dependências indiretas.

Note também que $StArg(B_1^\epsilon) = StArg(B_2^\epsilon)$, logo $Stronger(B_1^\epsilon, B_2^\epsilon) = \bot$, o que implica que ambos os argumentos B_1^ϵ e B_2^ϵ se derrotam mutuamente. □

3.3.2 Argumentos Apoiados e *Undercut*

As noções de argumentos *apoiados* e *undercut* por um conjunto de argumentos é fundamental para as definições posteriores de argumentos aceitáveis, rejeitados e justificados, sendo ingredientes para estas definições. A noção de argumento *apoiado* (Definição 14) é simples: um argumento é apoiado por um conjunto de argumentos se todos os seus subargumentos pertencem a esse conjunto.

Definição 14 (Argumento Apoiado)**.** *Um argumento $A \in Args$ é apoiado por um conjunto de argumentos $S \subseteq Args$, denotado $A\ Supp\ S$, sse todo subargumento próprio de A pertence a S. Formalmente, seja a relação $Supp : Args \times 2^{Args}$, $A\ Supp\ S$ sse $SA(A) \subseteq S$.*

Um argumento A ser *undercut* por um conjunto de argumentos S (Definição 15) indica que alguma premissa de A não pode ser aceita se os argumentos em S forem aceitos, uma vez que existe um argumento apoiado por S que derrota um subargumento próprio de A.

Definição 15 (Argumento *Undercut*)**.** *Um argumento $A \in Args$ é undercut por um conjunto de argumentos $S \subset Args$, denotado $A\ Undc\ S$ sse existe um argumento B, tal que B é apoiado por S, e B derrota um subargumento próprio de A. Formalmente, seja a relação $Undc : Args \times 2^{Args}$, $A\ Undc\ S$ sse $\exists B \in Args$ t.q. $B\ Supp\ S$ e $\exists C \in SA(A)$ t.q. $B \mathcal{D} C$.*

Continuação do Exemplo 3. (Veja a primeira parte do Exemplo 3 na página 82 e a segunda parte na página 88). Suponha $S = \{B_1^\delta, C_2^\delta, D_2^\delta\}$. A_3^δ é apoiado por S, pois todos os seus subargumentos estão em S. Suponha que C_6^δ derrote C_5^δ. Então um conjunto que contenha C_6^δ *undercuts* A_7^δ, visto que C_6^δ derrota um subargumento de A_7^δ.

□

Continuação do Exemplo 4. (Veja a primeira parte do Exemplo 4 na página 89). Seja $P_a(b) = 0{,}8$, $P_a(c) = 0{,}5$ e $P_a(d) = 0{,}5$, $P_b(e) = 0{,}8$, $P_b(f) = 0{,}8$. A força dos argumentos B_1^ϵ e B_2^ϵ continuam as mesmas de anteriormente, e as de A_1^ϵ e A_2^ϵ são dadas a seguir:

$$StArg(A_1^\epsilon) = \frac{0{,}8 \times 1 \times StArg(B_1^\epsilon)}{1} = 0{,}8 \times 0{,}8 = 0{,}64$$

$$StArg(A_2^\epsilon) = \frac{0{,}5 \times 1 \times StArg(C_1^\epsilon) + 0{,}5 \times 1 \times StArg(D_1^\epsilon)}{2} = \frac{0{,}5 \times 1 + 0{,}5 \times 1}{2} = 0{,}5$$

Logo, $Stronger(A_1^\epsilon,A_2^\epsilon) = A_1^\epsilon$ e $Stronger(B_1^\epsilon,B_2^\epsilon) = \bot$. Além disso, é importante mencionar que F_2^ϵ derrota F_1^ϵ, mas não o contrário, visto que F_2^ϵ é um argumento estrito. Seja $S = \{E_1^{\epsilon\prime}, E_1^\epsilon, F_2^\epsilon\}$, pode-se afirmar que:

- B_1^ϵ é apoiado por S, pois todos os seus subargumentos próprios (E_1^ϵ e $E_1^{\epsilon\prime}$) pertencem a S.
- B_2^ϵ é *undercut* por S, pois F_2^ϵ, que pertence a S e é um argumento estrito, derrota F_1^ϵ, que é o único subargumento de B_2^ϵ.

Assumindo que $S = \{E_1^{\epsilon\prime}, E_1^\epsilon, F_2^\epsilon, C_1^\epsilon, D_1^\epsilon, B_1^\epsilon\}$, o seguinte pode ser afirmado:

- A_2^ϵ e $A_1^{\epsilon\prime}$ são ambos apoiados por S.
- $A_1^{\epsilon\prime}$ não é *undercut* por S, pois nenhum de seus subargumentos é derrotado por um argumento apoiado por S (B_2^ϵ derrota B_1^ϵ, pois $Stronger(B_1^\epsilon,B_2^\epsilon) = \bot$, mas B_2^ϵ não é apoiado por S).

□

3.3.3 Argumentos Aceitáveis, Justificados e Rejeitados

Um argumento $A_i \in Args$ ser *aceitável* no que diz respeito a um conjunto de argumentos $Acc \subseteq Args$ – um conjunto de argumentos já tidos como aceitáveis – implica que, se aceitarmos os argumentos em Acc, então somos obrigados a aceitar A_i. Um argumento estrito sempre é aceitável, pois não pode ser derrotado. Um argumento A_i derrotável é aceitável se for apoiado e, de certa forma, defendido por Acc, uma vez que se requer que todos os argumentos que derrotam A_i, se existirem, sejam *undercut* por Acc. Um argumento falacioso nunca pode ser

aceitável, pois não pode ser usado para derivar positivamente sua conclusão. No entanto, um argumento falacioso A_j pode ainda ser usado para evitar que um argumento contrário A_i seja aceitável no que diz respeito a Acc, mesmo que A_i seja válido e apoiado por Acc, o que ocorre quando A_j derrota A_i e A_j não é *undercut* por Acc.

Definição 16 (Argumento Aceitável)**.** *Um argumento $A_i \in Args$ é aceitável n.q.d.r. a um conjunto de argumentos Acc sse:*

1. *A_i é um argumento estrito, i.e., $Type(A_i) = strict$; ou*
2. *a) A_i é um argumento válido (não falacioso), i.e., $Fall(A_i) = false$; e*

   b) A_i é apoiado por Acc, i.e., A_i $Supp$ Acc; e

   c) todo argumento em $Args$ que derrota A_i é undercut por Acc, i.e., $\forall A_j \in Args$, $A_j \mathcal{D} A_i \rightarrow A_j$ $Undc$ Acc.

Com base no conceito de argumentos aceitáveis, definem-se argumentos justificados, que são argumentos que resistem a qualquer refutação tomando-se o conjunto de argumentos como um todo, juntamente com as relações entre eles. Um argumento é justificado se ele for aceitável no que diz respeito a um conjunto de argumentos para os quais já se averiguou serem justificados. A definição é feita, portanto, em estágios, começando com um conjunto vazio e criando-se, a cada estágio, um novo conjunto contendo, além dos argumentos já contidos no conjunto anterior, os argumentos aceitáveis no que diz respeito a este, de modo que a sequência de conjuntos de cada estágio aumenta monotonicamente. Dessa forma, no primeiro estágio, acrescentam-se todos os argumentos estritos e os argumentos base derrotáveis que não são derrotados. Em seguida, são acrescentados todos os argumentos aceitáveis no que diz respeito aos argumentos justificados no primeiro estágio, e assim por diante, até que nenhum argumento que não foi justificado possa ser incluído em um novo estágio. Essa construção fica clara por meio dos exemplos apresentados em sequência.

Definição 17 (Argumentos Justificados)**.** *Seja um conjunto de argumentos $Args$. J_i é o i-ésimo estágio de justificação em $Args$, sendo definido como segue:*

- $J_0 = \emptyset$*;*
- $J_{i+1} = \{A \in Args | A$ *é aceitável n.q.d.r. a* $J_i\}$

Definição 18 (Conjunto Completo de Argumentos Justificados)**.** *O conjunto completo de argumentos justificados é denotado* $JArgs = \bigcup_{i=1}^{\infty} J_i$*, i.e., é a união dos conjuntos referentes a todos os estágios de justificação.*

Vale notar que a definição de conjunto completo de argumentos justificados como a união de uma quantidade infinita de conjuntos referentes aos estágios de justificação que se pode gerar provém do fato de que, a partir do estágio em que não há mais argumentos a serem justificados, todos os estágios seguintes serão iguais.

Um *r-literal* p é justificado quando existe um argumento justificado cuja conclusão é p, e, portanto, p é uma consequência lógica de KB.

Definição 19 (*R-literal* Justificado)**.** *Um r-literal* p *é justificado em* KB *sse existe um argumento para* p *em* $JArgs$*. Formalmente,* $KB \models p$ *sse* $\exists A \in JArgs$ *t.q.* $Conc(A) = p$*.*

Finalmente, introduz-se a noção de *argumentos rejeitados* e *r-literais rejeitados* a fim de caracterizar as conclusões que se pode demonstrar não serem derivadas de KB. Dados um conjunto de argumentos Rej – um conjunto dos argumentos já rejeitados – e um conjunto de argumentos Jus – um conjunto de argumentos justificados – um argumento ser rejeitado n.q.d.r. a Rej e Jus implica que, ou ele possui subargumentos que estão em Rej, ou não pode superar um ataque de um argumento apoiado por Jus.

Definição 20 (Argumento Rejeitado)**.** *Um argumento* A *é rejeitado n.q.d.r. aos conjuntos de argumentos* Rej *e* Jus *se* A *não é estrito e:*

1. *um subargumento próprio de* A *pertence a* Rej*; ou*
2. A *é derrotado por um argumento apoiado por* Jus*.*

Com base na definição de argumentos individuais rejeitados, o conjunto parcial de argumentos rejeitados dado um conjunto de argumentos $Args$ é definido como segue:

Definição 21 (Conjunto Parcial de Argumentos Rejeitados)**.** *Seja um conjunto de argumentos* $Args$*, e o conjunto completo de argumentos justificados* $JArgs$ *.* R_i *é o i-ésimo estágio de rejeição em* $Args$*, sendo definido como segue:*

- $R_0 = \emptyset$*;*
- $R_{i+1} = \{A \in Args | A$ *é rejeitado n.q.d.r. a* R_i *e* $JArgs\}$

Definição 22 (Conjunto Completo de Argumentos Rejeitados)**.** *O conjunto completo de argumentos rejeitados é denotado* $RArgs = \bigcup_{i=1}^{\infty} R_i$*, i.e., a união dos conjuntos referentes a todos os estágios de rejeição.*

Um *r-literal* p é tido como rejeitado se não houver nenhum argumento em $Args \setminus RArgs$ com a conclusão p (ver Definição 23). A intuição é que um *r-literal* é rejeitado se: (i) não existem argumentos para p em $Args$; ou (ii) existem argumentos para p em $Args$, mas estes são todos argumentos rejeitados, e, portanto, pertencem a $RArgs$. Assim, o caso (i) diz respeito à inexistência de motivos para aceitar p, enquanto o caso (ii) diz respeito à existência de motivos para aceitar p, porém tais motivos (os argumentos) são todos rejeitados dada a existência de argumentos contrários que os superam. O fato de p ser rejeitado significa que se provar que ele não é uma consequência lógica de KB.

Definição 23 (*R-literal* Rejeitado)**.** *Um r-literal* p *é rejeitado sse todos os argumentos para* p *pertencem a* $RArgs$*. Formalmente,* $KB \not\models p$ *sse* $\exists A \in Args \setminus RArgs$ *t.q.* $Conc(A) = p$*.*

Um argumento $A \in Args$ pode não ser nem justificado nem rejeitado, o que é apresentado na Propriedade 1, que segue diretamente das definições 16, 17 e 20. O caso (i) é trivial. O caso (ii), todavia, diz respeito a um argumento válido, porém impedido de ser justificado, seja por ter um subargumento que não é justificado, seja por ser derrotado por um argumento que não é *undercut*. Além disso, não pode ser rejeitado, seja por não ser derrotado por um argumento apoiado pelos justificados, mas apenas por argumentos rejeitados ou que não são justificados nem rejeitados, seja por ter um subargumento que não é nem rejeitado nem justificado.

Propriedade 1 (Argumento Não Justificado Nem Rejeitado)**.** *Um argumento* $A \in Args$ *pode não ser nem justificado nem rejeitado, o que ocorre nos seguintes casos:*

1. A *é falacioso (logo, pela condição (a) da Definição 16, não é aceitável por nenhum conjunto de argumentos, e, portanto, nunca pode ser justificado) e não possui um subargumento rejeitado nem é derrotado por um argumento apoiado por* $JArgs$ *(logo não é rejeitado, pela Definição 20); ou*

2. A *é válido, e não possui um subargumento rejeitado nem é derrotado por um argumento apoiado por* $JArgs$ *(logo não é rejeitado, pela Definição 20). Além disso: ou* A *não é apoiado por argumentos justificados (isto é, nem todos os seus subargumentos são justificados), e ao mesmo tempo não possui um subargumento rejeitado, i.e., pelo menos*

um de seus subargumentos não é nem justificado nem rejeitado; ou A é apoiado pelos justificados, mas é derrotado por um argumento que não é apoiado pelos justificados e que não é undercut, i.e., A é derrotado por um outro argumento que é rejeitado ou que não é nem justificado nem rejeitado (logo, pela Definição 16, não é aceitável, e portanto não justificado).

Um *r-literal* p pode não ser nem justificado nem rejeitado, conforme a Propriedade 2, que decorre das definições 19 e 23.

Propriedade 2 (*R-literal* Não Justificado Nem Rejeitado)**.** *Um r-literal p não é justificado nem rejeitado sse não existem argumentos justificados para p mas existem argumentos para p que não são nem justificados nem rejeitados, i.e., $\exists A \in JArgs$ t.q. $Conc(A) = p$ e $\exists A \in Args \setminus RArgs$, t.q. $Conc(A) = p$.*

A seguir são apresentados dois teoremas, que foram usados nos trabalhos anteriores sobre DL para afirmar a coerência e consistência do modelo (ANTONIOU *et al.*, 2000; GOVERNATORI *et al.*, 2004; BIKAKIS; ANTONIOU, 2010). As suas provas, apresentadas no Apêndice A, são relativamente simples e valem também para o modelo proposto neste trabalho, visto que as definições da semântica do DDRMAS são equivalentes às definições de DL. De fato, a inovação apresentada neste trabalho, no escopo da semântica de argumentação, concentra-se na relação de derrota, que é uma relação abstraída no *framework* de argumentação para DL proposto por Governatori *et al.* (2004), e difere também da definição de derrota proposta na CDL de Bikakis e Antoniou (2010), que não utiliza o conceito de cálculo de força de argumentos da mesma forma como é feito neste trabalho (ver Capítulo 5 para uma comparação aprofundada). A Seção 3.4 apresenta um procedimento que realiza uma transformação de DDRMAS para DL, o que reforça o fato de que o DDRMAS herda essas e outras propriedades da DL.

O Teorema 1 representa o fato de que dois argumentos (e consequentemente dois *r-literais*) não podem ser ambos justificados e rejeitados.

Teorema 1 (Coerência de um DDRMAS)**.** *Dado um DDRMAS $\mathcal{S}$ e um foco de consulta α:*

- *Um argumento não pode ser tanto justificado quanto rejeitado*
- *Um r-literal não pode ser tanto justificado quanto rejeitado*

Por fim, uma vez que se tem por pressuposto que as regras estritas de um agente são consistentes (i.e., não é possível derivar duas conclusões complementares como consequências

estritas da base de conhecimento do agente), então pode-se demonstrar a consistência do modelo por meio do Teorema 2. De acordo com Antoniou *et al.* (2000), a consistência diz que um literal (no caso deste trabalho, um *r-literal*) e sua negação podem ser provados derrotavelmente apenas quando o literal (*r-literal*) e sua negação podem ser provadas definitivamente; desse modo, não é possível que a inferência derrotável introduza inconsistência. Portanto, o Teorema 2 baseia-se em um absurdo para demonstrar que o modelo não leva à dedução de conclusões inconsistentes.

Teorema 2 (Consistência de um DDRMAS)**.** *Se o conjunto de argumentos justificados em um DDRMAS $\mathcal{S}$, $JArgs_{\mathcal{S}\alpha}$, contém dois argumentos com conclusões conflitantes, então ambos são argumentos estritos.*

Continuação do Exemplo 1.1. (Veja o Exemplo 1 na página 68, o Exemplo 1.1 na página 72, e suas continuações nas páginas 80 e 87). A Figura 14 apresenta uma ilustração dos argumentos justificados do Exemplo 1.1, também explicado textualmente a seguir.

- $J_0^{\mathcal{S}\alpha} = \{\}$, conforme a Definição 17;
- $J_1^{\mathcal{S}\alpha} = \{B_1^{\alpha\prime}, E_1^{\alpha\prime}, E_1^{\alpha\prime\prime}\}$, pois todos são argumentos base derrotáveis que não são derrotados por outros argumentos.
- $J_2^{\mathcal{S}\alpha} = \{B_1^{\alpha\prime}, E_1^{\alpha\prime}, E_1^{\alpha\prime\prime}, B_1^{\alpha}, E_1^{\alpha}\}$, pois B_1^{α} e E_1^{α} são ambos apoiados por $J_1^{\mathcal{S}\alpha}$ e não são derrotados por outros argumentos.
- $J_3^{\mathcal{S}\alpha} = \{B_1^{\alpha\prime}, E_1^{\alpha\prime}, E_1^{\alpha\prime\prime}, B_1^{\alpha}, E_1^{\alpha}, C_1^{\alpha}, D_1^{\alpha}\}$, pois: C_1^{α} e D_1^{α} são ambos apoiados por $J_2^{\mathcal{S}\alpha}$ e não são derrotados por outros argumentos. Embora A_2^{α} seja apoiado por $J_2^{\mathcal{S}\alpha}$, ele é derrotado por A_1^{α}, que não é *undercut* por nenhum argumento em $J_2^{\mathcal{S}\alpha}$. Logo, A_2^{α} não é justificado.
- $J_4^{\mathcal{S}\alpha} = \{B_1^{\alpha\prime}, E_1^{\alpha\prime}, E_1^{\alpha\prime\prime}, B_1^{\alpha}, E_1^{\alpha}, C_1^{\alpha}, D_1^{\alpha}, A_1^{\alpha}\}$, pois: A_1^{α} é apoiado por $J_3^{\mathcal{S}\alpha}$ e nenhum argumento o derrota, pois embora A_2^{α} e A_3^{α} ataquem A_1^{α}, eles não o derrotam. Embora A_3^{α} seja apoiado por $J_3^{\mathcal{S}\alpha}$, ele é derrotado por A_1^{α}, que não é *undercut* por nenhum argumento em $J_3^{\mathcal{S}\alpha}$. Logo, A_3^{α} também não é justificado.
- Não havendo mais argumentos a serem justificados, $JArgs_{\mathcal{S}\alpha} = J_4^{\mathcal{S}\alpha}$.

Quanto aos argumentos rejeitados, basta constatar que A_2^{α} e A_3^{α} são ambos derrotados por um argumento apoiado por $JArgs_{\mathcal{S}\alpha}$ (A_1^{α}). Portanto $RArgs_{\mathcal{S}\alpha} = R_1^{\mathcal{S}\alpha} = \{A_2^{\alpha}, A_3^{\alpha}\}$.

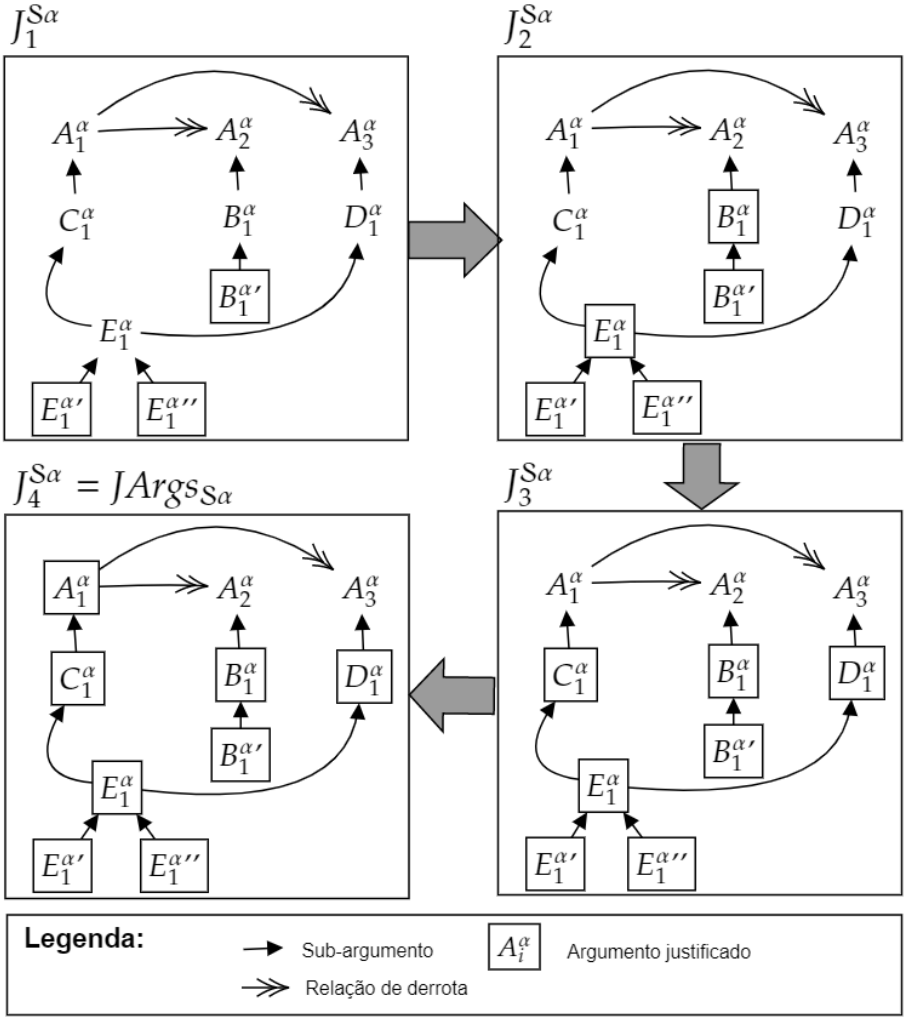
Portanto, os *r-literais* $\langle a,col(m_1)\rangle$, $\langle a,ed(m_1)\rangle$, $\langle b,\neg ed(m_1)\rangle$, $\langle c,ed(m_1)\rangle$, $\langle d,\neg ed(m_1)\rangle$, $\langle e,spa(m_1)\rangle$, $\langle b, hv(m_1)\rangle$, $\langle e, hv(m_1)\rangle$ e $\langle e, pbc(m_1)\rangle$ são *r-literais* justificados, enquanto $\langle a, \neg col(m_1)\rangle$ é o único *r-literal* rejeitado.

□

Continuação do Exemplo 1.2. (Veja o Exemplo 1 na página 68, o Exemplo 1.2 na página 72, e sua continuação na página 81). A partir do conjunto de argumentos $Args_{\mathcal{S}\beta}$, as seguintes etapas são usadas para criar um conjunto de argumentos justificados:

- $J_0^{\mathcal{S}\beta} = \{\}$, conforme a Definição 17;
- $J_1^{\mathcal{S}\beta} = \{A_1^{\beta\prime}\}$, pois $A_1^{\beta\prime}$ é um argumento base derrotável que não é derrotado por outros argumentos.

Figura 14 – Ilustração da construção do conjunto de argumentos justificados para o Exemplo 1.1.



Fonte: Autoria própria.

- $J_2^{\mathcal{S}\beta} = \{A_1^{\beta\prime}, A_1^{\beta}\}$, pois A_1^{β} é apoiado por $J_1^{\mathcal{S}\beta}$ e nenhum outro argumento o derrota.
- $J_3^{\mathcal{S}\beta} = \{A_1^{\beta\prime}, A_1^{\beta}, B_1^{\beta}\}$, pois B_1^{β} é apoiado por $J_2^{\mathcal{S}\beta}$ e nenhum outro argumento o derrota.
- Não havendo mais argumentos a serem justificados, $JArgs_{\mathcal{S}\beta} = J_3^{\mathcal{S}\beta}$.

Não há argumentos rejeitados. Portanto, os *r-literais* $\langle b, \neg col(m_2)\rangle$, $\langle a, \neg ed(m_2)\rangle$ e $\langle a, dc(m_2)\rangle$ são justificados.

□

Continuação do Exemplo 4. (Veja o Exemplo 4 na página 89 e sua continuação na página 92). Pode-se derivar os seguintes argumentos justificados, dados os argumentos da Figura 13 e as relações de força $Stronger(A_1^{\epsilon\prime}, A_2^{\epsilon}) = A_1^{\epsilon\prime}$ e $Stronger(B_1^{\epsilon}, B_2^{\epsilon}) = \bot$.

- $J_0^{\mathcal{S}\epsilon} = \{\}$, conforme a Definição 17;
- $J_1^{\mathcal{S}\epsilon} = \{C_1^{\epsilon}, D_1^{\epsilon}, F_2^{\epsilon}, E_1^{\epsilon\prime}\}$, pois: os três primeiros são argumentos estritos, logo são sempre aceitáveis; e $E_1^{\epsilon\prime}$, que é o subargumento de E_1^{ϵ} com conclusão $\langle \mathcal{F}, z_1\rangle$, é um argumento base derrotável mas não tem argumento que o derrote;
- $J_2^{\mathcal{S}\epsilon} = \{C_1^{\epsilon}, D_1^{\epsilon}, F_2^{\epsilon}, E_1^{\epsilon\prime}, E_1^{\epsilon}, \}$, pois E_1^{ϵ} é apoiado por $J_1^{\mathcal{S}\epsilon}$ e nenhum argumento o derrota. Embora A_2^{ϵ} seja apoiado por $J_1^{\mathcal{S}\epsilon}$, ele é derrotado por A_1^{ϵ}, que não é *undercut* por nenhum argumento em $J_1^{\mathcal{S}\epsilon}$. Logo, A_2^{ϵ} não é justificado.
- $J_3^{\mathcal{S}\epsilon} = \{C_1^{\epsilon}, D_1^{\epsilon}, F_2^{\epsilon}, E_1^{\epsilon\prime}, E_1^{\epsilon}, B_1^{\epsilon}\}$, pois: B_1^{ϵ} é apoiado por $J_2^{\mathcal{S}\epsilon}$; e B_1^{ϵ} é derrotado por B_2^{ϵ}, porém B_2^{ϵ} é *undercut* por $J_1^{\mathcal{S}\epsilon}$ em F_1^{ϵ}, visto que F_2^{ϵ} derrota F_1^{ϵ}.
- $J_4^{\mathcal{S}\epsilon} = \{C_1^{\epsilon}, D_1^{\epsilon}, F_2^{\epsilon}, E_1^{\epsilon\prime}, E_1^{\epsilon}, B_1^{\epsilon}, A_1^{\epsilon\prime}\}$, pois: $A_1^{\epsilon\prime}$ é apoiado por $J_3^{\mathcal{S}\epsilon}$ e nenhum argumento o derrota, pois embora A_2^{ϵ} ataque $A_1^{\epsilon\prime}$, ele não o derrota.
- $J_5^{\mathcal{S}\epsilon} = \{C_1^{\epsilon}, D_1^{\epsilon}, F_2^{\epsilon}, E_1^{\epsilon\prime}, E_1^{\epsilon}, B_1^{\epsilon}, A_1^{\epsilon\prime}, A_1^{\epsilon}\}$, pois: A_1^{ϵ} é apoiado por $J_4^{\mathcal{S}\epsilon}$ e nenhum argumento o derrota.
- Não havendo mais argumentos a serem justificados, $JArgs_{\mathcal{S}\epsilon} = J_5^{\mathcal{S}\epsilon}$.

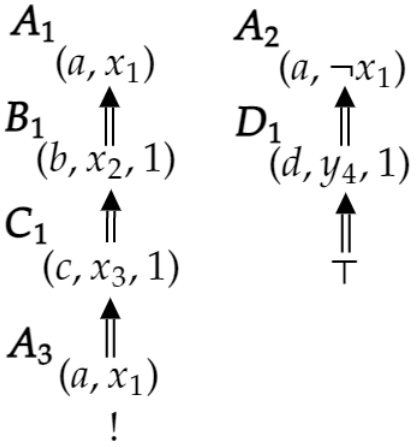
Quanto aos argumentos rejeitados, basta constatar que A_2^{ϵ}, F_1^{ϵ} e B_2^{ϵ} são todos derrotados por argumentos apoiados por $JArgs_{\mathcal{S}\epsilon}$ (A_1^{ϵ}', F_2^{ϵ}, e B_1^{ϵ}, respectivamente). Portanto $RArgs_{\mathcal{S}\epsilon} = R_1^{\mathcal{S}\epsilon} = \{A_2^{\epsilon}, F_1^{\epsilon}, B_2^{\epsilon}\}$.

Portanto, os *r-literais* $\langle a,x_1\rangle$, $\langle a,y_1\rangle$, $\langle b,y_2\rangle$, $\langle c,y_3\rangle$, $\langle d,y_4\rangle$, $\langle e,y_5\rangle$ e $\langle f, \neg y_6\rangle$ são *r-literais* justificados, enquanto $\langle a, \neg y_1\rangle$, $\langle b, \neg y_2\rangle$ e $\langle f, y_6\rangle$ são *r-literais* rejeitados.

□

Exemplo 5. Este exemplo visa mostrar a semântica havendo argumentos falaciosos. Apresenta-se diretamente os argumentos (Figura 15), sem as regras, visto estas não serem necessárias para a demonstração. Também não está sendo considerado um foco de consulta específico para o exemplo.

Figura 15 – Conjunto de argumentos para o Exemplo 5.



Fonte: Autoria própria.

Neste sistema, há um argumento falacioso A_1 e um válido A_2, tal que A_1 não é apoiado por nenhum conjunto de argumentos, pois pela definição um argumento falacioso nunca é apoiado. Logo, A_1 também não pode ser justificado em nenhuma hipótese.

Se $Stronger(A_1,A_2) = A_1$, então A_1 derrota A_2 e nenhum argumento *undercuts* A_1. Logo A_2 também não pode ser justificado. Como não há argumento apoiado pelos argumentos justificados que derrote nem A_1 e nem A_2, ambos também não podem ser rejeitados.

Se $Stronger(A_1,A_2) = A_2$, então A_2 derrota A_1 e nenhum argumento *undercuts* A_2. Logo A_2 pode ser justificado, visto que D_1 também é justificado. Neste caso, A_1 é rejeitado, pois existe um argumento justificado que o derrota.

Se $Stronger(A_1,A_2) = \perp$, então A_2 derrota A_1 e nenhum argumento *undercuts* A_2, mas A_1 também derrota A_2 e nenhum argumento *undercuts* A_1. Logo nem A_1 nem A_2 são justificados. Neste caso, porém, A_1 é rejeitado, pois existe um argumento apoiado por argumentos justificados (A_1) que o derrota, enquanto A_2 não pode ser rejeitado, pois, embora seja apoiado por argumentos justificados (D_1), não existe um argumento apoiado por argumentos justificados

que o derrote.

□

3.3.4 Cálculos Alternativos de Força e Comparação de Argumentos

Esta seção apresenta algumas ideias alternativas para o cálculo de força e comparação de argumentos.

Como apresentado na Seção 3.3.1, o cálculo de força padrão de um argumento $StArg$ considera a ideia de confiança social, no sentido de que se considera a força de cada argumento derivado a partir de regras de um agente colaborador como sendo calculada do ponto de vista deste, e não do agente que emitiu a consulta. Isto, além de se aproximar mais de uma forma de raciocínio contextual, que deve atender à propriedade de perspectiva, também permite que resultados parciais do cálculo sejam calculados pelos diferentes agentes colaboradores, o que também torna o cálculo mais eficiente, visto que o agente que recebe uma resposta a uma consulta poderia receber também as forças dos argumentos resultantes já calculadas. Esse aspecto do cálculo da força será chamado de *receptivo*, uma vez que considera o ponto de vista dos demais agentes, e não somente o próprio.

Outro aspecto tocado pelo cálculo de força padrão é a questão das dependências indiretas entre os argumentos. A intuição é que, se um argumento para um determinado literal depende da existência de outro argumento definido por outro agente, que por sua vez depende da existência de outro argumento definido por outro agente, e assim por diante, tal argumento terá força menor do que se dependesse diretamente de argumentos de um único agente. Isso pode ser comparado às relações de confiança direta e indireta entre agentes, como apresentado na Seção 2.4. Uma analogia interessante é a da brincadeira infantil chamada de "telefone sem fio", que começa com uma mensagem inicial que deve ser passada por cada jogador ao próximo cochichando-a ao ouvido, o que acaba, muitas vezes, por deturpar a mensagem original. É claro que, em meio a isso, entram os valores da função de confiança e similaridade dos *r-literais*, mas o Exemplo 4 demonstra claramente que, em condições iguais no que diz respeito a esses parâmetros e tendo os agentes colaboradores confiança menor que 1 entre si, as dependências indiretas impactam negativamente na força do argumento. Esse aspecto do cálculo de força será chamado de *suspeitador*, visto que suspeita de informações vindas de terceiros.

Também é possível considerar a possibilidade de agentes mais céticos ou mais crédulos no que diz respeito ao cálculo de força de argumentos. O cálculo de força padrão realiza uma

média entre as forças dos subargumentos instanciados diretos, com a intuição de que todos os argumentos externos do qual um agente depende diretamente colaboram igualmente para o alcance de uma conclusão. De fato, se um dos subargumentos instanciados de um argumento não existir, o argumento nem mesmo existiria. No entanto, em uma abordagem mais cética, é possível que um agente considere, no cálculo de força, apenas a força do subargumento mais fraco. Essa abordagem pode ser mais interessante quando se pretende uma maior confiabilidade na tomada de decisões, e faz sentido do ponto de vista lógico, visto que os argumentos são baseados em regras com corpo conjuntivo. Outra abordagem, mais crédula, seria considerar apenas a força do subargumento mais forte.

Portanto, são apresentadas quatro abordagens distintas que se contrapõem às intuições endereçadas pelo cálculo de força padrão. A primeira, chamada *cálculo de força arrogante* ($StArg_{arr}$), apresentada na Seção 3.3.4.1, desconsidera a perspectiva dos agentes colaboradores, calculando a força de todos os argumentos do ponto de vista do agente que emitiu a consulta. A segunda, chamada *cálculo de força ingênua* ($StArg_{naive}$), apresentada na Seção 3.3.4.2, considera iguais todos os *ir-literais* na estrutura do argumento, mesmo que digam respeito a subargumentos instanciados não diretos, de modo que a força de um argumento será proporcional à média das forças de todos os seus *ir-literais*. Por fim, também são apresentados os cálculos de força *cético* e *crédulo*, que consistem simplesmente na substituição da média utilizada nas fórmulas pelos operadores de mínimo e máximo.

Outra ideia interessante que pode ser considerada é uma definição alternativa para a função de comparação de forças $Stronger$. Na definição de $Stronger$ padrão, compara-se as forças de argumentos individuais, de modo que sempre os argumentos mais fortes, que não são *undercut*, para ambas as conclusões conflitantes, são os que definem qual das conclusões será justificada. A intuição para isso é que um agente racional vai se posicionar a favor de uma conclusão com base no melhor motivo para considerá-la verdadeira, comparado ao melhor motivo para considerá-la falsa. No entanto, uma outra ideia poderia ser a consideração da força conjunta de todos os argumentos a favor de uma conclusão e a força conjunta de todos os argumentos contrários, com a intuição de que, quanto mais motivos existirem para se posicionar a favor de uma determinada conclusão, e quanto mais fortes eles forem, melhor. Essa definição será chamada de *comparação de forças conjuntas* de argumentos, denotada pela função $Stronger_{joint}$, a qual será apresentada na Seção 3.3.5.1.

3.3.4.1 Cálculo de Força de Argumento Arrogante

A ideia do cálculo de força arrogante, dado o $StArg$ padrão, é simples: em vez de se calcular a força de um *r-literal* considerando a função de confiança do agente que define o argumento, calcula-se a força de um *r-literal* considerando a função de confiança do agente que emitiu a consulta. Para isso, é necessário definir uma função $StArg_{Aux}$, que, além do argumento, recebe também o agente cuja função de confiança deve-se utilizar.

Definição 24 (Cálculo Arrogante de Força de um Argumento)**.** *Seja* $StArg_{Aux} : Args \times Ags \rightarrow [0,1]$ *uma função que recebe um argumento* A *e um identificador de um agente* a_0 *e retorna um valor entre 0 e 1 da seguinte maneira:*

$$StArg_{Aux}(A,a_0) = \begin{cases} 1 & \textit{se } DISA(A) = \emptyset \\ \dfrac{\sum\limits_{A' \in DISA(A)} StIRL(Conc(A'),a_0) \times StArg_{Aux}(A', a_0)}{|DISA(A)|} & \textit{se } DISA(A) \neq \emptyset \end{cases}$$

$StArg_{arr} : Args \rightarrow [0,1]$ *é uma função que recebe um argumento* A *e retorna um valor entre 0 e 1 da seguinte maneira:*

$$StArg_{arr}(A) = StArg_{Aux}(A,ArgD(A))$$

Dessa forma, todos os argumentos terão a força baseada na função de confiança de a, que submeteu a consulta.

3.3.4.2 Cálculo de Força de Argumento Ingênuo

A fórmula desse cálculo de força segue uma abordagem diferente do cálculo de força padrão. Em vez da fórmula ser recursiva sobre os subargumentos instanciados diretos, simplesmente calcula-se a média da força de todos os *ir-literais* de um argumento, capturados pela função $IRLs$.

Definição 25 (Cálculo Ingênuo de Força de um Argumento)**.** $StArg_{naive} : Args \rightarrow [0,1]$ *é uma função que recebe um argumento* A *e retorna um valor entre 0 e 1 da seguinte maneira:*

$$StArg_{naive}(A) = \frac{\sum_{q}^{IRLs(A)} StIRL(q,D(q))}{|IRLs(A)|}$$

É possível também definir uma versão arrogante do cálculo de força ingênuo, juntando os dois aspectos alternativos identificados. A fórmula é simplesmente uma variante da fórmula para força ingênua ($StArg_{naive}$), porém não utilizando o definidor de cada *ir-literal*, mas sim o definidor do próprio argumento A. A função $StArg_{naive\&arr}$ é definida então da seguinte forma:

Definição 26 (Cálculo Arrogante e Ingênuo de Força de um Argumento)**.** $StArg_{naive\&arr} : Args \rightarrow [0,1]$ *é uma função que recebe um argumento* at *e retorna um valor entre 0 e 1 da seguinte maneira:*

$$StArg_{naive\&arr}(A) = \frac{\sum_{q}^{IRLs(A)} StIRL(q,ArgD(A))}{|IRLs(A)|}$$

3.3.5 Cálculos de Força de Argumento Cético e Crédulo

A ideia dos cálculos de força cético e crédulo é simples: em vez de se utilizar uma média da força dos *ir-literais* e dos subargumentos instanciados, utiliza-se min, no caso de força cética, e max, no caso de força crédula.

Definição 27 (Cálculo de Força Cética de um Argumento)**.** $StArg_{sk} : Args \rightarrow [0,1]$ *é uma função que recebe um argumento* A *e retorna um valor entre* 0 *e* 1*, onde* 0 *representa o menor valor possível de força e* 1 *o maior valor possível de força, da seguinte maneira:*

$$StArg_{sk}(A) = \begin{cases} 1 & \text{se } DISA(A) = \emptyset \\ \min\limits_{A' \in DISA(A)} StIRL(Conc(A'),ArgD(A)) \times StArg_{sk}(A') & \text{se } DISA(A) \neq \emptyset \end{cases}$$

Definição 28 (Cálculo de Força Crédula de um Argumento)**.** $StArg_{cr} : Args \rightarrow [0,1]$ *é uma função que recebe um argumento* A *e retorna um valor entre* 0 *e* 1*, onde* 0 *representa o menor valor possível de força e* 1 *o maior valor possível de força, da seguinte maneira:*

$$StArg_{cr}(A) = \begin{cases} 1 & \text{se } DISA(A) = \emptyset \\ \max\limits_{A' \in DISA(A)} StIRL(Conc(A'),ArgD(A)) \times StArg_{cr}(A') & \text{se } DISA(A) \neq \emptyset \end{cases}$$

Esses métodos podem também ser combinados ao cálculo de força arrogante e ingênua, bastando substituir a operação de média (somatório dividido pela quantidade de subargumentos instanciados diretos, no caso da não ingênua, ou pela quantidade de *ir-literais*, no caso da ingênua) pelos operadores min e max.

3.3.5.1 Comparação de Forças de Argumentos Conjuntas

A função de comparação de forças conjunta considera a soma das forças de todos os argumentos para um *r-literal* p e seu complemento $\sim p$. A definição da força conjunta é dada então como segue.

Definição 29 (Função de Comparação de Forças Conjuntas)**.** *Seja um conjunto de argumentos* $Args_p \subseteq Args$ *que concluem* p *e um conjunto de argumentos* $Args_{\sim p} \subseteq Args$ *que concluem* $\sim p$*. Seja um argumento* $A_p \in Args_p$ *e um argumento* $A_{\sim p} \in Args_{\sim p}$*, a função* $Stronger_{joint}$ *é tal que:*

- $Stronger_{joint}(A_p, A_{\sim p}) = A_p$ *sse* $\sum_{A'_p \in Args_p}(StArg(A'_p)) > \sum_{A'_{\sim p} \in Args_{\sim p}}(StArg(A'_{\sim p}))$*, ou*
- $Stronger_{joint}(A_p, A_{\sim p}) = A_{\sim p}$ *sse* $\sum_{A'_p \in Args_p}(StArg(A'_p)) < \sum_{A'_{\sim p} \in Args_{\sim p}}(StArg(A'_{\sim p}))$*, ou*
- $Stronger_{joint}(A_p, A_{\sim p}) = \bot$ *sse* $\sum_{A'_p \in Args_p}(StArg(A'_p)) = \sum_{A'_{\sim p} \in Args_{\sim p}}(StArg(A'_{\sim p}))$.

Continuação do Exemplo 1.1. (Veja o Exemplo 1 na página 68, o Exemplo 1.1 na página 72, e suas continuações nas páginas 80, 87 e 97). Os cálculos de força nas diferentes alternativas ficam como segue.

- Arrogante:

$$StArg_{arr}(A_1^{\alpha}) = StArg_{Aux}(A_1^{\alpha}, a) = \frac{StIRL(\langle c, ed(m_1), 1\rangle, a) \times StArg_{Aux}(C_1^{\alpha}, a)}{1}$$

$$= P_a(c) \times 1 \times \frac{StIRL(\langle e, spa(m_1), 0.8\rangle, a) \times StArg_{Aux}(E_1^{\alpha}, a)}{1}$$

$$= 0{,}6 \times 1 \times P_a(e) \times 0{,}8 \times 1 = 0{,}6 \times 1 \times 0{,}8 \times 0{,}8 = 0{,}384$$

$$StArg_{arr}(A_2^{\alpha}, a) = StArg_{Aux}(A_2^{\alpha}) = \frac{StIRL(\langle b, \neg ed(m_1), 1\rangle, a) \times StArg_{Aux}(B_1^{\alpha}, a)}{1}$$

$$= P_a(b) \times 1 = 0{,}4$$

$$StArg_{arr}(A_3^{\alpha}, a) = StArg(A_3^{\alpha}) = \frac{StIRL(\langle d, \neg ed(m_1), 1\rangle, a) \times StArg_{Aux}(D_1^{\alpha}, a)}{1}$$

$$= P_a(d) \times 1 \times \frac{StIRL(\langle e, spa(m_1), 0.4\rangle, a) \times StArg_{Aux}(E_1^\alpha, a)}{1}$$

$$= 0{,}2 \times 1 \times P_a(e) \times 0{,}4 \times 1 = 0{,}2 \times 1 \times 0{,}8 \times 0{,}4 = 0{,}064$$

É interessante notar que, neste caso, A_2^α derrota A_1^α, fazendo com que $\langle a, \neg col(m_1)\rangle$ seja justificado.

- Ingênuo:

$$StArg_{naive}(A_1^\alpha) = \frac{StIRL(\langle c, ed(m_1), 1\rangle, a) + StIRL(\langle e, spa(m_1), 0.8\rangle, c)}{2}$$

$$= \frac{P_a(c) \times 1 + P_c(e) \times 0{,}8}{2} = \frac{0{,}6 + 1 \times 0{,}8}{2} = \frac{0{,}6 + 0{,}8}{2} = 0{,}7$$

$$StArg_{naive}(A_2^\alpha) = \frac{StIRL(\langle b, \neg ed(m_1), 1\rangle, a)}{1} = P_a(b) \times 1 = 0{,}6$$

$$StArg_{naive}(A_3^\alpha) = \frac{StIRL(\langle d, \neg ed(m_1), 1\rangle, a) + StIRL(\langle e, spa(m_1), 0.4\rangle, d)}{2}$$

$$= \frac{P_a(d) \times 1 + P_d(e) \times 0{,}4}{2} = \frac{0{,}2 + 1 \times 0{,}4}{2} = \frac{0{,}2 + 0{,}4}{2} = 0{,}3$$

Note como a força de A_1^α, neste caso, é bem superior à força padrão, justamente por desconsiderar a dependência indireta que há entre a e e.

- Ingênuo e Arrogante:

$$StArg_{arr\&naive}(A_1^\alpha) = \frac{StIRL(\langle c, ed(m_1), 1\rangle, a) + StIRL(\langle e, spa(m_1), 0.8\rangle, a)}{2}$$

$$= \frac{P_a(c) \times 1 + P_a(e) \times 0{,}8}{2} = \frac{0{,}6 + 0{,}8 \times 0{,}8}{2} = \frac{0{,}6 + 0{,}64}{2} = 0{,}62$$

$$StArg_{arr\&naive}(A_2^\alpha) = \frac{StIRL(\langle b, \neg ed(m_1), 1\rangle, a)}{1} = P_a(b) \times 1 = 0{,}6$$

$$StArg_{arr\&naive}(A_3^\alpha) = \frac{StIRL(\langle d, \neg ed(m_1), 1\rangle, a) + StIRL(\langle e, spa(m_1), 0.4\rangle, d)}{2}$$

$$= \frac{P_a(d) \times 1 + P_a(e) \times 0{,}4}{2} = \frac{0{,}2 + 0{,}8 \times 0{,}4}{2} = \frac{0{,}2 + 0{,}32}{2} = 0{,}26$$

Quanto aos cálculos de força cético e crédulo, no caso deste exemplo, só farão diferença quando combinados ao cálculo de força ingênuo. No caso, $StArg_{naive\&sk}(A_1^\alpha) = StIRL(\langle c, ed(m_1), 1\rangle, a) = 0{,}6$ (força ingênua e cética), e $StArg_{naive\&cr}(A_1^\alpha) = StIRL(\langle e, spa(m_1), 0.8\rangle, c) = 0{,}8$ (força ingênua e crédula).

Quanto à comparação de forças conjuntas dos argumentos, considerando o cálculo de força padrão, a soma das forças dos argumentos para $\langle a, col(m1)\rangle$ é $0{,}48$, uma vez que existe apenas um argumento (A_1^α), enquanto que a soma das forças dos argumentos para $\langle a, \neg col(m1)\rangle$ (surpreendentemente) resulta em $0{,}4 + 0{,}08 = 0{,}48$. Portanto, $Stronger_{joint}(A_1^\alpha, A_2^\alpha) = Stronger_{joint}(A_1^\alpha, A_3^\alpha) = \bot$. Logo, A_1^α, A_2^α e A_3^α são todos rejeitados, e assim também $\langle a, col(m1)\rangle$ e $\langle a, \neg col(m1)\rangle$ são rejeitados.

□

3.4 TRANSFORMAÇÃO EM LÓGICA DERROTÁVEL (DL) CENTRALIZADA

O objetivo do procedimento descrito nesta seção é a construção de uma teoria centralizada de DL $\mathcal{DL}(\mathcal{S}, \alpha)$, que produz os mesmos resultados que um DDRMAS $\mathcal{S}$ dado um foco de consulta α. A existência desse procedimento permite transformar o raciocínio distribuído apresentado em um raciocínio centralizado, coletando as BCs distribuídas em uma única base de conhecimento central e criando uma teoria DL equivalente. Isso permite demonstrar que o formalismo e a semântica proposta neste trabalho é correta e completa no que diz respeito à DL (ver Seção 2.2.2.1), permitindo afirmar que o DDRMAS herda todas suas propriedades, tais como consistência e coerência (ANTONIOU *et al.*, 2000). Também permite afirmar que um sistema DDRMAS pode ser executado de forma centralizada com complexidade polinomial – visto que a Lógica Derrotável tem complexidade polinomial, conforme (MAHER, 2001).

Vale notar também que a transformação inversa não é possível, pois ocorre uma perda de expressividade ao se reduzir todos os conceitos contidos em um DDRMAS para um simples conjunto de regras estritas e derrotáveis e de uma relação de superioridade entre essas regras. Isto ocorre pois, transformando-se um DDRMAS em uma teoria DL, perdem-se informações referentes a regras esquemáticas, funções de confiança entre agentes e à função de similaridade entre *r-literais*. Além disso, a geração da relação de superioridade entre regras é feita a partir dos argumentos baseados nessas regras. Logo, o significado por trás da relação de superioridade é perdido, pois depende de informações que vão além do definido pelas regras individualmente. Isso somente demonstra que o DDRMAS pode ser visto como uma especialização da DL, tendo maior poder de expressividade a fim de atender aos requisitos apresentados, mas, mesmo assim, herdando as propriedades da DL já demonstradas na literatura.

O procedimento de transformação segue cinco etapas:

1. As regras estritas locais de cada BC são adicionadas como regras estritas em $\mathcal{DL}(\mathcal{S}, \alpha)$.

2. As regras locais derrotáveis e as regras de mapeamento, exceto as esquemáticas, de cada BC, são adicionadas como regras derrotáveis em $\mathcal{DL}(\mathcal{S}, \alpha)$.

3. As regras de foco são localizadas, isto é, são adicionadas a $\mathcal{DL}(\mathcal{S}, \alpha)$ como regras locais definidas por cada um dos agentes.

4. As regras esquemáticas são instanciadas conforme todas as possibilidades de instanciações baseadas em similaridade de suas premissas que são *er-literais*. Tais regras instanciadas são então adicionadas como regras derrotáveis em $\mathcal{DL}(\mathcal{S}, \alpha)$.

5. Para cada par de regras com conclusões contraditórias tal que existem argumentos para uma conclusão que não são derrotados por argumentos para a outra, uma relação de superioridade é adicionada entre essas regras.

É importante mencionar que, na teoria derrotável resultante $\mathcal{DL}(\mathcal{S}, \alpha)$, os rótulos de definidor dos *r-literais* devem ser mantidos, visto que existe a possibilidade de regras de mapeamento no formato $r_a : \langle a,x\rangle \Leftarrow \langle b,x\rangle$, isto é, regras cujo literal é afirmado em um agente a se for afirmado por outro agente b, como ocorre também em SMCs (ver Seção 2.3.3). Além disso, os *fr-literais* não são adicionados diretamente. Conforme a etapa 3, para cada regra de foco que contém *fr-literais*, novas regras, para cada agente distinto, são formadas substituindo-se os *fr-literais* por *cr-literais*, o que é chamado de *localização das regras de foco*. Portanto, os literais na teoria derrotável resultante são todos os *cr-literais* da lógica derrotável distribuída $\mathcal{S}$, acrescidos dos *cr-literais* gerados a partir das regras de foco do foco de consulta α, denotado $V^R_{\mathcal{DL}(\mathcal{S},\alpha)} = V^{CR}_{\mathcal{S}\alpha} \cup V^*_\alpha$, onde V^*_α é o conjunto de *r-literais* resultante da localização dos *fr-literais*. Os *er-literais* também não são adicionados, visto que são substituídos por literais concretos no processo de instanciação (etapa 4).

As etapas 1 e 2 do procedimento são triviais. A etapa 3 é apresentada a seguir:

- Para cada $a \in Ags$:
  - Para cada $r^{\mathcal{F}} : Head(r^{\mathcal{F}}) \Leftarrow Body(r^{\mathcal{F}}) \in KB^{\mathcal{F}}_\alpha$, cria-se uma regra $r^{\mathcal{F}}_a : Localized\_RL(Head(r^{\mathcal{F}}), a) \Leftarrow \{Localized\_RL(q, a) \mid q \in Body(r^{\mathcal{F}})\}$, onde $Localized\_RL : V^R_{\mathcal{S}\alpha} \times Ags \to V^R_{\mathcal{S}\alpha} \cup V^*_\alpha$ é uma função definida como segue:

$$Localized\_RL(q,a) = \begin{cases} q & \text{se } D(q) \neq \mathcal{F} \\ \langle a, L(q) \rangle & \text{se } D(q) = \mathcal{F} \end{cases}$$

Isto é, se q não for um *fr-literal*, usa-se o próprio q. Caso contrário, usa-se um *cr-literal* local com rótulo a. Em seguida, adiciona-se $r_a^{\mathcal{F}}$ a $\mathcal{DL}(\mathcal{S}, \alpha)$.

A etapa 4 é apresentada a seguir:

- Para cada regra esquemática r_a em $KB_{\mathcal{S}\alpha}^{@}$, calcula-se o conjunto de regras esquemáticas instanciadas $R_{inst}(r_a)$, e adiciona-se todas as regras de $R_{inst}(r_a)$ em $\mathcal{DL}(\mathcal{S}, \alpha)$. A função $R_{inst}(r_a)$ é definida como:

$$R_{inst}(r_a) = \{Head(r_a) \Leftarrow IBody \cup CBody(r_a) \mid IBody \in \eta_{SBody(r_a)}\}$$

onde $SBody(r_a) \subseteq Body(r_a)$ é o subconjunto dos antecedentes no corpo da regra r_a que são *er-literais*, $CBody(r_a) = Body(r_a) \setminus SBody(r_a)$ são os antecedentes que são *cr-literais*, e $\eta_{SBody(r_a)}$ é o conjunto de combinações de todas as substituições possíveis dos *er-literais* de r_a por *cr-literais* que são conclusões de outras regras em $KB_{\mathcal{S}\alpha}$, definido como segue:

$$\eta_{SBody(r_a)} = \prod_{p \in SBody(r_a)} \sigma(p, r_a)$$

onde $\sigma(p,r_a)$ denota o conjunto de todas as possíveis substituições de p por *cr-literais* similares o suficiente que são conclusões de outras regras:

$$\sigma(p, r_a) = \{p' \mid \exists r_b \in KB_{\mathcal{S}\alpha} \text{ t.q. } Head(r_b) = p', \Theta(p,p') \geq st\}$$

A etapa 5 consiste na relação de superioridade, formada da seguinte maneira:

- Para cada par de regras derrotáveis $r_{ai} : p \Leftarrow Body(r_{ai})$ e $r_{aj} :\sim p \Leftarrow Body(r_{aj})$ em $\mathcal{DL}(\mathcal{S}, \alpha)$, e seja $Args_p \subseteq Args_{\mathcal{S}\alpha}$ o conjunto de argumentos para p. Adiciona-se a superioridade $r_{ai} > r_{aj}$ a $\mathcal{DL}(\mathcal{S}, \alpha)$ sse $Args_p \cap JArgs_{\mathcal{S}\alpha} \neq \emptyset$.

A intuição da etapa 5 é que uma regra r_{ai} é considerada superior a uma regra r_{aj}, dado que ambas têm como consequentes *r-literais* complementares entre si, se algum argumento

para o *r-literal* que é o consequente da regra r_{ai} é justificado. Conforme o Teorema 2, se um argumento para p é justificado, então um argumento para $\sim p$ não pode ser justificado. Como a justificação de um *r-literal* depende dos conjuntos de argumentos para p e para $\sim p$, como um todo, e não de regras individuais, segue que é necessário que todas as regras com cabeça p sejam tidas como superiores a todas as regras com cabeça $\sim p$, se for o caso de p ser justificado.

Uma vez construída a lógica derrotável centralizada equivalente, é possível interpretá-la usando-se o método de prova (derivação) apresentado na Seção 2.2.2.1. O Teorema 3 afirma a equivalência entre um sistema $\mathcal{S}$ com foco de consulta α e sua lógica derrotável centralizada correspondente $\mathcal{DL}(\mathcal{S},\alpha)$ em termos de derivação lógica.

Teorema 3 (Equivalência entre DDRMAS e DL)**.** *Dado um literal* $p \in V^R_{\mathcal{DL}(\mathcal{S},\alpha)}$*:*

1. $\exists A \in JArgs_{\mathcal{S}\alpha}$ *t.q.* A *é um argumento estrito e* $Conc(A) = p$ *se e somente se* $\mathcal{DL}(S,\alpha) \vdash +\Delta p$*.*
2. $\exists A \in JArgs_{\mathcal{S}\alpha}$ *t.q.* A *é um argumento estrito e* $Conc(A) = p$ *se e somente se* $\mathcal{DL}(S,\alpha) \vdash -\Delta p$*.*
3. $\exists A \in JArgs_{\mathcal{S}\alpha}$ *t.q.* A *é um argumento derrotável e* $Conc(A) = p$ *se e somente se* $\mathcal{DL}(S,\alpha) \vdash +\partial p$*.*
4. $\exists A \in JArgs_{\mathcal{S}\alpha}$ *t.q.* A *é um argumento derrotável e* $Conc(A) = p$ *se e somente se* $\mathcal{DL}(S,\alpha) \vdash -\partial p$*.*

A prova é apresentada no Apêndice A.

Continuação do Exemplo 1.1. (Veja o Exemplo 1 na página 68, o Exemplo 1.1 na página 72, e suas continuações nas páginas 80, 87 e 97). Seguindo as etapas 1 e 2, as seguintes regras são adicionadas a $\mathcal{DL}(\mathcal{S}, \alpha)$. Considera-se que a variável livre M já tenha sido substituída pela constante m_1 por meio de algum mecanismo de unificação.

$$r_{a1} : \langle a, \neg ed(m_1)\rangle \Leftarrow \langle a, dc(m_1)\rangle$$

$$r_{b1} : \langle b, \neg ed(m_1)\rangle \Leftarrow \langle b, hv(m_1)\rangle$$

$$r_{e1} : \langle e, spa(m_1)\rangle \Leftarrow \langle e, hv(m1)\rangle, \langle e, pbc(m_1)\rangle$$

A etapa 3 adiciona as seguintes regras com base nas regras de foco por meio da função de localização:

$$r^{\mathcal{F}}_{a1} : \langle a, hv(m_1)\rangle \Leftarrow \qquad r^{\mathcal{F}}_{a2} : \langle a, pbc(m_1)\rangle \Leftarrow$$
$$r^{\mathcal{F}}_{b1} : \langle b, hv(m_1)\rangle \Leftarrow \qquad r^{\mathcal{F}}_{b2} : \langle b, pbc(m_1)\rangle \Leftarrow$$
$$r^{\mathcal{F}}_{c1} : \langle c, hv(m_1)\rangle \Leftarrow \qquad r^{\mathcal{F}}_{c2} : \langle c, pbc(m_1)\rangle \Leftarrow$$
$$r^{\mathcal{F}}_{d1} : \langle d, hv(m_1)\rangle \Leftarrow \qquad r^{\mathcal{F}}_{d2} : \langle d, pbc(m_1)\rangle \Leftarrow$$
$$r^{\mathcal{F}}_{e1} : \langle e, hv(m_1)\rangle \Leftarrow \qquad r^{\mathcal{F}}_{e2} : \langle e, pbc(m_1)\rangle \Leftarrow$$

A etapa 4 adiciona as instanciações possíveis das regras esquemáticas:

$$r'_{a2} : \langle a, col(m_1)\rangle \Leftarrow \langle c, ed(m_1)\rangle \qquad r'_{b2} : \langle b, col(m_1)\rangle \Leftarrow \langle c, ed(m_1)\rangle$$
$$r'_{a3} : \langle a, \neg col(m_1)\rangle \Leftarrow \langle b, \neg ed(m_1)\rangle \qquad r'_{b3} : \langle b, \neg col(m_1)\rangle \Leftarrow \langle b, \neg ed(m_1)\rangle$$
$$r''_{a3} : \langle a, \neg col(m_1)\rangle \Leftarrow \langle d, \neg ed(m_1)\rangle \qquad r''_{b3} : \langle b, \neg col(m_1)\rangle \Leftarrow \langle d, \neg ed(m_1)\rangle$$
$$r'_{c1} : \langle c, ed(m_1)\rangle \Leftarrow \langle e, spa(m_1)\rangle \qquad r'_{d1}\langle d, \neg ed(m_1)\rangle \Leftarrow \langle e, spa(m_1)\rangle$$

Finalmente, considerando os argumentos e seus *status* – A^{α}_1 é justificado, enquanto A^{α}_2 e A^{α}_3 são rejeitados – apenas as seguintes relações de superioridade (etapa 5) podem ser geradas:

$$r'_{a2} > r'_{a3} \qquad r'_{a2} > r''_{a3}$$

Quanto à derivação, seguindo as condições apresentadas na Seção 2.2.2.1: inicialmente, é possível derivar os fatos de que nenhuma das conclusões dentre todas as regras podem ser provadas definitivamente, visto não haver regras estritas. Portanto, considerando a existência de 20 *r-literais* distintos, tem-se $P(0..19)$, sendo que para todo *r-literal* $p \in \mathcal{DL}(\mathcal{S},\alpha)$, $-\Delta p$ é calculado.

As regras $r^{\mathcal{F}}_{11}$ a $r^{\mathcal{F}}_{52}$, totalizando 10 regras com 10 diferentes conclusões, não possuem corpo e não há nenhuma regra com conclusão contraditória. Portanto, seus consequentes podem ser derrotavelmente provados, isto é, $\forall r \in \{r^{\mathcal{F}}_{11}...r^{\mathcal{F}}_{52}\}$, $+\partial Head(r) \in P(20..29)$.

Uma vez tendo $P(0..29)$, é possível derivar $P(30)$ e $P(31)$ a partir das regras r_{b1} e r_{e1} como segue:

- $P(30) = +\partial\langle b,\neg ed(m_1)\rangle$ pois: a condição (2.1) é atendida, uma vez que existe a regra r_{b1} cujo único literal em seu corpo $\langle b, hv(m_1)\rangle$ está em $P(0..29)$; as condições (2.2) e (2.3) são atendidas pois $-\Delta\langle b,ed(m_1)\rangle$ está em $P(0..29)$ e não existem regras que concluem $\langle b,ed(m_1)\rangle$.

- $P(31) = +\partial\langle e,spa(m_1)\rangle$ pois: a condição (2.1) é atendida, uma vez que existe a regra r_{e1} cujos literais em seu corpo $\langle e, hv(m_1)\rangle$ e $\langle e, pbc(m_1)\rangle$ estão em $P(0..30)$; as condições (2.2) e (2.3) são atendidas pois $-\Delta\langle e,\neg spa(m_1)\rangle$ está em $P(0..30)$ e não existem regras que concluem $\langle e,\neg spa(m_1)\rangle$.

Em seguida, é possível derivar $P(32)$ e $P(33)$ por meio das regras r'_{c1} e r'_{d1}:

- $P(32) = +\partial\langle c, ed(m_1)\rangle$ pois: a condição (2.1) é atendida, uma vez que existe a regra r'_{c1} cujo único literal em seu corpo $\langle e, spa(m_1)\rangle$ está em $P(0..31)$ (especificamente $P(31)$); as condições (2.2) e (2.3) são atendidas pois $-\Delta\langle c, \neg ed(m_1)\rangle$ está em $P(0..31)$ e não existem regras que concluem $\langle c, \neg ed(m_1)\rangle$.

- $P(33) = +\partial\langle d, \neg ed(m_1)\rangle$ pois: a condição (2.1) é atendida, uma vez que existe a regra r'_{d1} cujo único literal em seu corpo $\langle e, spa(m_1)\rangle$ está em $P(0..32)$ (especificamente $P(31)$); as condições (2.2) e (2.3) são atendidas pois $-\Delta\langle d, ed(m_1)\rangle$ está em $P(0..32)$ e não existem regras que concluem $\langle d, ed(m_1)\rangle$.

Neste momento, tem-se as conclusões das regras r'_{a2}, r'_{a3} e r''_{a3} como candidatas para ter suas conclusões provadas derrotavelmente. No entanto, apenas $\langle a, col(m_1)\rangle$ pode ser provada conforme segue:

- $P(34) = +\partial\langle a, col(m_1)\rangle$ pois: a condição (2.1) é atendida, uma vez que existe a regra r'_{a2} cujo único literal em seu corpo $\langle c, ed(m_1)\rangle$ está em $P(0..33)$ (especificamente $P(32)$); a condição (2.2) é atendida pois $-\Delta\langle a, \neg col(m_1)\rangle$ está em $P(0..33)$; e a condição (2.3) é atendida visto que, para toda regra que conclui $\langle a, \neg col(m_1)\rangle$, o que inclui r'_{a3} e r''_{a3}, tem-se que $r'_{a2} > r'_{a3}$ e $r'_{a2} > r''_{a3}$, o que atende à condição (2.3.2).

Em seguida, é possível derivar o fato de que $\langle a, \neg col(m_1)\rangle$ não é provada derrotavelmente, como segue:

- $P(35) = -\partial\langle a,\neg col(m_1)\rangle$ pois: a condição (2.1) é atendida, uma vez que $-\Delta\langle a,\neg col(m_1)\rangle$; e a condição (2.3) é atendida, visto que existe uma regra (r'_{a2}) que conclui $\langle a,col(m_1)\rangle$, tal que é apoiada por $P(0..34)$, atendendo à condição (2.3.1), e tal que, para toda regra que conclui o contrário $\langle a, \neg col(m_1)\rangle$, o que inclui r'_{a3} e r''_{a3}, nenhuma delas é superior a r'_{a2}, isto é, $r'_{a3} \not> r'_{a2}$ e $r''_{a3} \not> r'_{a2}$, o que atende à condição (2.3.2).

□

3.5 DISCUSSÃO

Este capítulo apresentou um *framework* de argumentação estruturada capaz de modelar o problema proposto. Uma representação de conhecimento baseada em lógica derrotável acrescida de regras de mapeamento inspiradas em SMCs foi apresentada, assim como a formalização do conceito de consulta e BC de foco.

Em seguida, foi definida a estrutura de um argumento e como ele pode ser derivado das regras de uma BC. Os argumentos podem possuir *ir-literais*, que fazem a vinculação entre argumentos definidos por diferentes agentes e cujos *r-literais* possuem diferentes graus de similaridade.

Em seguida, foi apresentada a semântica de argumentação, que é baseada na semântica de argumentação da DL (GOVERNATORI *et al.*, 2004). A principal contribuição deste trabalho nesse aspecto é a definição de cálculo de força de argumentos, que considera a confiança entre os agentes e o grau de similaridade dos *ir-literais* que referenciam conhecimentos de diferentes agentes nos argumentos.

Outra contribuição apresentada é a proposta de alternativas de cálculos de forças de argumentos que refletem diferentes atitudes dos agentes. Também é apresentada uma função de comparação de argumentos alternativa que considera a soma das forças dos argumentos a favor de uma determinada conclusão.

A fim de atestar a completude e corretude do *framework*, foi apresentada uma transformação para uma teoria em DL centralizada, demonstrando que o *framework* pode ser visto como uma especialização da DL, portanto herdando suas propriedades, inclusive consistência e coerência.

4 ALGORITMO DISTRIBUÍDO PARA RESPOSTA A CONSULTAS

O capítulo anterior apresentou as definições formais e caracterização semântica da abordagem proposta. Este capítulo apresenta um algoritmo distribuído para avaliação de consultas entre agentes baseado nessa formalização. A Seção 4.1 descreve o algoritmo em detalhes, assim como exemplos de execução. A Seção 4.2 apresenta propriedades do algoritmo, o que inclui: corretude e completude no que diz respeito ao modelo de argumentação apresentado no Capítulo 3, terminação do algoritmo e complexidade computacional, assim como otimizações necessárias a fim de tornar o algoritmo viável.

4.1 DESCRIÇÃO DO ALGORITMO

Esta seção apresenta o algoritmo distribuído para avaliação de consultas que realiza o modelo proposto. O problema específico de raciocínio é o seguinte: *Dado um SMA $\mathcal{S}$ e um foco de consulta α para um r-literal p enviado ao agente a, compute o valor-verdade de p com base na existência ou inexistência de argumentos para algum r-literal similar a p.* O algoritmo produz uma resposta com três componentes: (i) um valor-verdade $tv_p \in \{true, false, undec\}$ para o *r-literal* p consultado; (ii) um conjunto de argumentos que concluem *r-literais* similares o suficiente a p; e (iii) um conjunto de argumentos que concluem *r-literais* similares o suficiente a $\sim p$. Dessa forma, o agente que recebe uma resposta também recebe os argumentos que apoiam e refutam p, de modo que ele possa realizar a resolução de conflitos do seu próprio ponto de vista. Como apresentado mais adiante no Teorema 6, o valor-verdade $true$ implica que algum *r-literal* p' similar o suficiente a p é justificado em $KB_{\mathcal{S}\alpha}$, $false$ implica que todos os p' similares o suficiente a p são rejeitados em $KB_{\mathcal{S}\alpha}$, e $undec$ implica que nenhum p' similar o suficiente a p é justificado, mas algum p' similar o suficiente a p não é nem justificado nem rejeitado em $KB_{\mathcal{S}\alpha}$.

Os argumentos nesta representação algorítmica possuem a mesma estrutura dos argumentos apresentados na Seção 3.2, possuindo, adicionalmente, três propriedades *booleanas*: J, indicando que o argumento já foi identificado como justificado; R, indicando que o argumento já foi identificado como rejeitado; e $SuppJ$, indicando que o argumento é apoiado por um conjunto de argumentos justificados. Essas três propriedades têm papel fundamental na comparação entre os conjuntos de argumentos a favor e contrários a p, de forma a implementar corretamente a semântica apresentada na Seção 3.3. Considera-se também que cada argumento possui uma

propriedade numérica St, referente à força do argumento, cujo valor padrão é 0, mas que pode ser modificado ao se calcular a força do argumento por meio da função $StArg$.

A apresentação do algoritmo será dividida em duas seções. A Seção 4.1.1 apresenta o procedimento principal $Query$, juntamente com algumas das funções auxiliares utilizadas, $Find\_Similar\_RLs$ e $Local\_Ans$. A Seção 4.1.2 apresenta o procedimento auxiliar $Find\_Def\_Args$, que corresponde a boa parte do algoritmo como um todo, e, portanto, será apresentado separadamente a fim de facilitar a leitura e compreensão. Esta última seção também apresenta as funções auxiliares de $Find\_Def\_Args$, a saber: $Query\_Agents$ e $Build\_Def\_Args\_For\_Rule$.

4.1.1 Procedimento $Query$

Cada agente implementa o mesmo algoritmo, que também define um protocolo básico de mensagens que permite que eles colaborem efetivamente. O procedimento principal, $Query$, apresentado no Algoritmo 1, é chamado quando um agente a recebe uma mensagem da forma $Query(p, \alpha, hist_p)$ de um agente a_0. Isto pode ser considerado um ato de fala solicitante (do Inglês, *Ask*) emitido por ele mesmo ou por outro agente, onde: p é o *r-literal* consultado; $\alpha = (p,a',KB_\alpha^{\mathcal{F}})$ é o foco de consulta; e $hist_p$ é uma lista de *r-literais* já avaliados durante o processamento da consulta, o que permite evitar laços infinitos por meio da detecção da ocorrência de argumentos falaciosos. O resultado da consulta é retornado para a_0 como uma mensagem no formato $Ans(p, \alpha, tv_p, Args_p, Args_{\sim p})$, onde: p é o *r-literal* consultado; α é o foco de consulta; tv_p é o valor-verdade de p; $Args_p$ é um conjunto de argumentos que apoia a conclusão p; e $Args_{\sim p}$ é um conjunto de argumentos que apoia a conclusão $\sim p$.

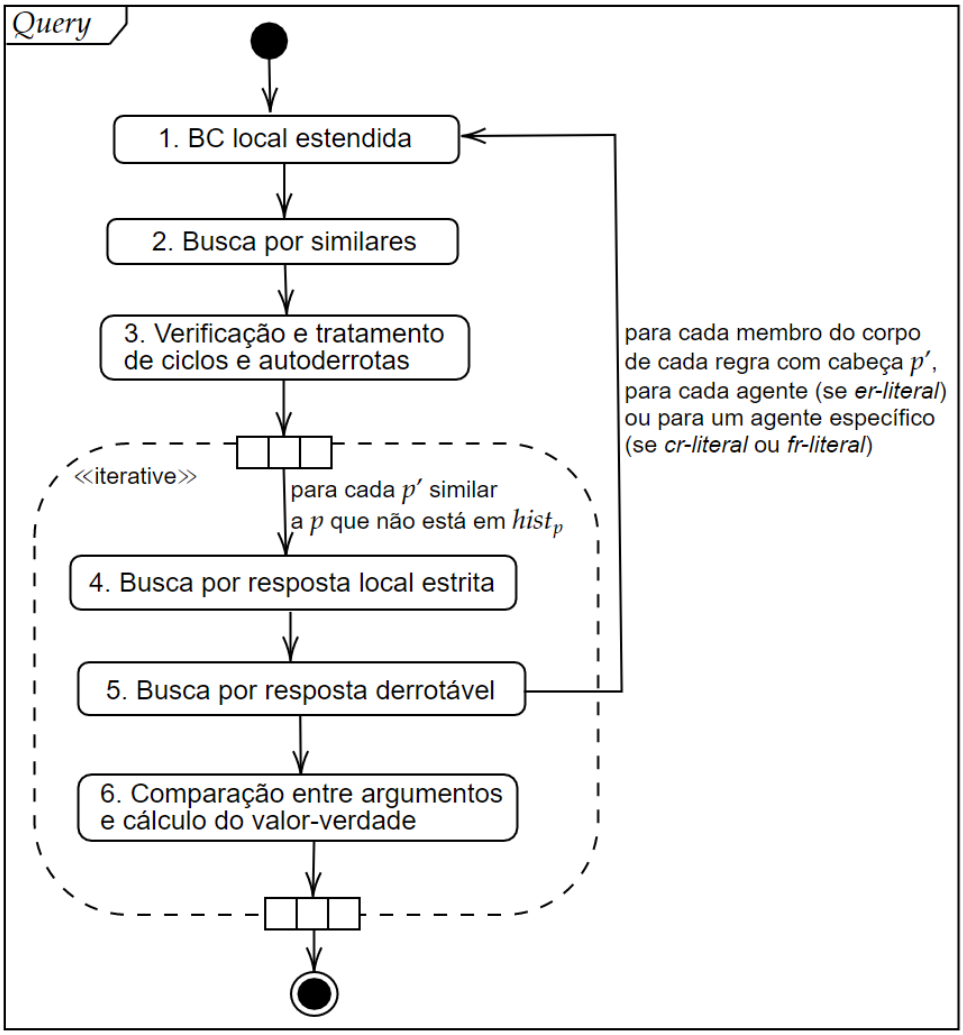
O algoritmo pode ser dividido em 6 (seis) etapas principais:

1. Criação da BC local estendida dado o foco de consulta α (linha 3)
2. Busca por *r-literais* similares a p na BC local estendida (linha 4 e 5)
3. Verificação e tratamento de ciclos e cadeias de regras autodestrutivas (linhas 6 e 7)
4. Busca de resposta estrita local para cada *r-literal* similar (linhas 8 a 15)
5. Busca de resposta derrotável local ou distribuída para cada *r-literal* similar (linhas 16 a 18)

6. Comparação entre argumentos e cálculo do valor-verdade baseado nas forças dos conjuntos de argumentos derrotáveis a favor e contra a conclusão de cada *r-literal* similar (linhas 19 a 26)

A Figura 16 apresenta um diagrama de atividades que ilustra a execução dessas etapas.

Figura 16 – Diagrama de atividades para o algoritmo $Query$.



Fonte: Autoria própria.

Antes das etapas, algumas variáveis são definidas e inicializadas (linha 2): tv_p como $false$, sendo o valor-verdade a ser retornado como resposta à consulta; $Args_p$ como um conjunto vazio, sendo o conjunto de argumentos para p (ou similar a p) retornado na resposta; e $Args_{\sim p}$ como um conjunto vazio, sendo o conjunto de argumentos para $\sim p$ (ou similar a $\sim p$) retornado na resposta. Tais variáveis serão atualizadas no decorrer do algoritmo, e por isso são marcadas como *thread-safe*, no sentido de que uma única *thread* por vez pode atualizá-las.

Algoritmo 1 – Pseudocódigo do procedimento principal $Query$.

```
1  when a receives message Query (p,α = (p,a',KB_α^F),hist_p) from a_0
2      thread-safe vars: tv_p = false; Args_p ← ∅; Args_~p ← ∅
3      KB_aα ← KB_a ∪ Localized_FRules(KB_α^F, a);
4      rlits ← Find_Similar_RLs(p,KB_aα)
5      if rlits = ∅ then send Ans(p,α,false,∅,∅) to a_0 and terminate
6      Args_p ← {p'! | p' ∈ rlits, {p', ~ p'} ∩ hist_p ≠ ∅}
7      if Args_p ≠ ∅ then tv_p = undec
8      executing in parallel for each p' ∈ rlits s.t. {p', ~ p'} ∩ hist_p = ∅
9          has_strict_answer_p' = false
10         if Local_Ans(p') = (true, A_p') then
11             Args_p ← Args_p ∪ {A_p'}
12             tv_p ← true; has_strict_answer_p' = true
13         if Local_Ans(~ p') = (true, A_~p') then
14             Args_~p ← Args_~p ∪ {A_~p'}
15             tv_p ← false; has_strict_answer_p' = true
16         executing commands in parallel and waiting for all to finish
17             Args_p' ← Find_Def_Args(p',KB_aα,α,hist_p)
18             Args_~p' ← Find_Def_Args(~ p',KB_aα,α,hist_p)
19         if has_strict_answer_p' = true and tv_p = false then
20             for A_p' ∈ Args_p' do A_p'.R ← true
21         else if has_strict_answer_p' = false then
22             tv_p' ← Compare_Def_Args(Args_p',Args_~p')
23             if tv_p' = true then tv_p = true
24             else if tv_p' = undec and tv_p ≠ true then tv_p ← undec
25         Args_p ← Args_p ∪ Args_p'
26         Args_~p ← Args_~p ∪ Args_~p'
27     send Ans(p,α,tv_p,Args_p,Args_~p) to a_0 and terminate
```

Fonte: Autoria própria.

Etapa 1: BC local estendida

Na linha 3, uma base de conhecimento local estendida $KB_{a\alpha}$ é definida por meio da união entre a base de conhecimento do agente, KB_a, e a base de conhecimento de foco $KB_\alpha^{\mathcal{F}}$ com as regras localizadas para o agente a, similarmente à localização de regras de foco apresentada na etapa 3 da Seção 3.4.

Etapa 2: Busca por similares

A linha 4 chama o procedimento $Find\_Similar\_RLs$, cujo objetivo é verificar se existe algum *r-literal* p' similar o suficiente a p em $KB_{a\alpha}$. Esta função é simples, consistindo em iterar por cada regra em $KB_{a\alpha}$ e comparar sua cabeça com o *r-literal* p por meio da função de

similaridade Θ. Portanto, a função retorna um conjunto de *r-literais* que são similares o suficiente a p. Se nenhum *r-literal* for encontrado, o procedimento $Query$ retorna imediatamente uma resposta $false$ para a_0 com conjuntos de argumentos vazios.

Etapa 3: Verificação e tratamento de ciclos e cadeias de regras autodestrutivas

A linha 6 cria as respostas para os *r-literais* identificados como estando em $hist_p$, de modo que nós-folha falaciosos são criados para p' para cada $p' \in rlits$ tal que $p' \in hist_p$ (significando que um ciclo foi detectado) ou $\sim p' \in hist_p$ (significando que um argumento autodestrutivo foi detectado). Os resultados são adicionados ao conjunto de argumentos $Args_p$. A linha 7 atribui o valor-verdade $undec$ a tv_p se nós-folha falaciosos foram adicionados a $Args_p$ na linha 6, uma vez que basta existir um argumento não rejeitado para que um *r-literal* não seja rejeitado.

Etapa 4: Busca por resposta local estrita

A iteração entre as linhas 8 e 24 toma todos os *r-literais* $p' \in rlits$ tal que nem p' nem $\sim p'$ estão em $hist_p$ tentando encontrar uma resposta e argumentos para p' e $\sim p'$, tanto localmente quanto de maneira distribuída. É importante que as iterações sejam executadas em paralelo, pois potencialmente exigirão que novas consultas sejam emitidas para outros agentes, o que requer que o agente espere pelas respostas. Sem essa paralelização, uma única *thread* de execução teria que esperar por cada resposta para cada consulta, que podem ser múltiplas, não permitindo usufruir dos benefícios da computação distribuída relacionados à simultaneidade.

A linha 10 chama a função auxiliar $Local\_Ans$ para tentar encontrar uma resposta localmente por meio das regras estritas locais. Esta função é apresentada no Algoritmo 2. $Local\_Ans$ recebe o *r-literal* p' e gera uma tupla $(tv_{p'}, A_{p'})$, onde $tv_{p'} \in \{true, false\}$, e $A_{p'}$ é um argumento criado por meio da função New_Arg (linhas 3 e 5), que recebe: um *r-literal* p' que é a conclusão do argumento; um vetor de premissas do argumento, que pode ser um conjunto de subargumentos ou um único símbolo $\top$, que marca o argumento como um argumento base; um sinalizador que pode ser o valor $strict$ (argumento estrito) ou $defeasible$ (argumento derrotável); e um valor de força padrão do argumento, que no caso é 1, conforme apresentado no cálculo de força padrão na Seção 3.3.1 no Capítulo 3. Além disso, no escopo de $Local\_Ans$, um argumento sempre é marcado como $strict$.

A busca é feita seguindo uma estratégia de busca em profundidade, iterando sobre cada regra estrita $r \in KB_a^s$ tal que $Head(r) = p'$, até que um argumento aplicável seja encontrado, ou até que se esgotem essas regras. Se $Body(r) = \emptyset$, então um argumento base A com conclusão p' é criado (linha 3). Caso contrário, $Local\_Ans$ é recursivamente chamado para cada $q \in Body(r)$ (linha 6). Se um valor-verdade $false$ for encontrado para qualquer q, a regra não é aplicável para criar um argumento e, portanto, é descartada (linha 8). Caso contrário, o argumento para q (A_q) é adicionado como um subargumento próprio de $A_{p'}$ por meio da função Add_Sub_Arg (linha 9). Se um argumento para p' for encontrado, então um valor-verdade $true$ é retornado juntamente com o argumento (linha 10). Caso contrário, um valor-verdade $false$ é retornado juntamente com um valor nulo $\perp$ (linha 11).

Algoritmo 2 – Pseudocódigo da função $Local\_Ans$.

```
1  function Local_Ans (p') :
2      for r ∈ KB_a^s and Head(r) = p' do
3          if Body(r) = ∅ then A ← New_Arg(p',[⊤],strict)
4          else
5              A_p' ← New_Arg(p',[],strict)
6              for q ∈ Body(r) do
7                  (tv_q, A_q) ← Local_Ans(q)
8                  if tv_q = false then stop and go to next rule
9                  Add_Sub_Arg(A_p', A_q)
10         return (true, A_p')
11     return (false, ⊥)
```

Fonte: Autoria própria.

De volta ao procedimento $Query$, quando $Local\_Ans$ para p' retorna $true$ (linha 10), o argumento $A_{p'}$ é adicionado ao conjunto de argumentos $Args_p$ (linha 11), o valor-verdade $true$ é atribuído tv_p, e $true$ é atribuído a uma variável *booleana* $has\_strict\_answer_{p'}$, que indica que uma resposta foi encontrada de forma estrita, de modo que, mesmo que outros argumentos derrotáveis sejam encontrados na próxima etapa, o valor-verdade não poderá mais ser alterado (linha 12). O mesmo é feito para $\sim p'$: $Local\_Ans$ é chamado para $\sim p'$ (linha 13). Se houver um argumento estrito que prove $\sim p'$, o argumento é adicionado a $Args_{\sim p}$, o valor-verdade $false$ é atribuído a tv_p, e $has\_strict\_answer_{p'}$ é marcado com $true$ (linhas 14 e 15).

Etapa 5: Busca por resposta derrotável

Nesta etapa o agente tenta construir argumentos derrotáveis para p' e $\sim p'$ por meio da função $Find\_Def\_Args$, apresentada no Algoritmo 4 e descrita em detalhes na Seção 4.1.2.

É importante mencionar que ambas as chamadas para $Find\_Def\_Args$ (linhas 17 e 18) são executadas em paralelo, visto que são processamentos independentes que podem resultar em diferentes emissões de mensagens $Query$.

$Find\_Def\_Args$ é um dos principais componentes do algoritmo distribuído, pois é por meio dessa função que outros agentes são consultados e os argumentos derrotáveis para um *r-literal* p' são construídos. A função recebe como entrada: um *r-literal* p', a base de conhecimento $KB_{a\alpha}$, um foco de consulta α e um histórico $hist_p$. A base de conhecimento passada por $Query$ é $KB_{a\alpha}$, pois é necessário considerar regras vindas da base de conhecimento do foco. O foco de consulta e o histórico também devem ser passados, pois serão encaminhados em consultas subsequentes para outros agentes, que também deverão ficar cientes do conhecimento de foco, e o histórico é necessário para detectar argumentos falaciosos nas consultas subsequentes. A função retorna um conjunto de argumentos $Args_{p'}$ que concluem p'.

Etapa 6: Comparação entre argumentos e cálculo do valor-verdade

Caso argumentos estritos para $\sim p'$ tenham sido encontrados anteriormente na etapa 4, é necessário marcar cada argumento em $Args_{p'}$ como rejeitado, por meio da propriedade R. Isto é feito nas linhas 19 e 20, e é necessário pois, se existe um argumento estrito para $\sim p$, então ele derrota todos os argumentos derrotáveis para p e não é derrotado por nenhum deles, de modo que estes então só podem ser rejeitados.

Caso não tenha sido encontrada uma resposta estrita na etapa 4, o que é sinalizado por $has\_strict\_answers_{p'}$ (linha 21), são feitas comparações entre os conjuntos $Args_{p'}$ e $Args_{\sim p'}$ de modo a derivar um valor-verdade a partir dos argumentos derrotáveis gerados, o que é realizado por meio da função $Compare\_Def\_Args$, apresentada no Algoritmo 3 e chamada na linha 22 de $Query$. Esse valor-verdade está relacionado ao fato de o *r-literal* p' ser justificado ou não, como apresentado no Teorema 6. Além disso, a função também marca os argumentos que podem ser tidos como justificados e rejeitados, por meio das propriedades *booleanas* J e R atreladas a cada argumento. Portanto, a comparação feita nessa etapa do algoritmo tem correspondência direta com a semântica de argumentação apresentada na Seção 3.3.

Para cada argumento $A_{p'}$ em $Args_{p'}$, verifica-se se ele é apoiado por argumentos justificados, por meio da propriedade *booleana* $SuppJ$ (linha 4 de $Compare\_Def\_Args$). Tal propriedade é atribuída na função $Find\_Def\_Args$ caso todos os subargumentos próprios de um argumento recém-criado sejam justificados. Se for o caso, e todos os argumentos para $\sim p'$

Algoritmo 3 – Pseudocódigo da função $Compare\_Def\_Args$.

```
1  function Compare_Def_Args (Args_{p'}, Args_{~p'}) :
2      tv_{p'} ← undec
3      for each A_{p'} ∈ Args_{p'} do
4          if A_{p'}.SuppJ = true and ∀A_{~p'} ∈ Args_{~p'} s.t. A_{~p'}.R = false:
               A_{~p'} D A_{p'} then
5              A_{p'}.J ← true
6              tv_{p'} ← true
7          else if ∃A_{~p'} ∈ Args_{~p'} s.t. A_{~p'}.SuppJ = true and A_{~p'} D A_{p'} then
8              A_{p'}.R ← true
9      if tv_{p'} ≠ true and (Args_{p'} = ∅ or ∀A_{p'} ∈ Args_{p'}, A_{p'}.R = true) then
10         tv_{p'} ← false
11     return tv_{p'}
```

Fonte: Autoria própria.

em $Args_{\sim p'}$ que não estejam marcados como rejeitados não derrotam $A_{p'}$, então $A_{p'}$ pode ser marcado como justificado (ver definições 16 e 17). Além disso, uma resposta $true$ pode ser atribuída a $tv_{p'}$ (linha 6), pois, visto que existe um argumento para p' que é justificado, segue que p' é justificado (ver Definição 19). Note que todo argumento que já está marcado como rejeitado indica que algum de seus subargumentos próprios é rejeitado, o que implica que o argumento é *undercut* pelo conjunto de argumentos justificados. Isto porque, se um de seus subargumentos é rejeitado, então o próprio subargumento ou um subargumento dele foi derrotado durante a execução do procedimento $Compare\_Def\_Args$ no escopo da consulta em que foi construído (ver linhas 7 e 8). Portanto, não é necessário verificar se argumentos já rejeitados derrotam $A_{p'}$, visto que tal derrota não pode impedir que $A_{p'}$ seja aceitável, uma vez que esses argumentos são *undercut* (ver definições 15, 16 e 17).

Caso exista um argumento para $\sim p$ que seja apoiado pelos argumentos justificados e que derrote $A_{p'}$, então $A_{p'}$ é marcado como rejeitado, o que decorre diretamente da definição de argumentos rejeitados (Definição 20). Após a iteração sobre todos os argumentos em $Args_{p'}$, verifica-se se $Args_{p'}$ é vazio ou se todos os argumentos em $Args_{p'}$ foram marcados como rejeitados. Neste caso, o valor-verdade $false$ será retornado, uma vez que p' é um *r-literal* rejeitado (ver Definição 23).

Voltando ao procedimento $Query$, sempre que um valor-verdade $true$ é encontrado para um *r-literal* p', significa que, para algum p' similar o suficiente a p, existe um argumento justificado. Logo, $true$ é atribuído a tv_p (linha 23). Caso um valor-verdade $undec$ seja encontrado para p', então $undec$ também será atribuído a tv_p, contanto que tv_p já não seja $true$ – o que indica que um argumento justificado para um outro p' já foi encontrado (linha 24). Vale mencionar que,

por padrão, $tv_p = false$, então, se $tv_{p'} = false$, não será preciso alterar tv_p. Além disso, se $tv_{p'} = false$ e existem argumentos rejeitados em $Args_{p'}$, tais argumentos não podem alterar a resposta tv_p, visto que basta que exista um argumento justificado ou um que não seja justificado nem rejeitado para que a resposta seja diferente de $false$.

Antes de encerrar a iteração, adicionam-se todos os argumentos de $Args_{p'}$ a $Args_p$, e todos os argumentos de $Args_{\sim p'}$ a $Args_{\sim p}$, de modo que a resposta final da consulta conterá todos os argumentos possíveis para todos os *r-literais* p' similares o suficiente a p.

Por fim, após todos os *r-literais* similares o suficiente terem sido processados, a resposta contendo tv_p, $Args_p$ e $Args_{\sim p}$ é retornada a a_0 (linha 25).

Continuação do Exemplo 1.1. (Veja o Exemplo 1 na página 68, o Exemplo 1.1 na página 72, e suas continuações nas páginas 80, 87 e 97). A seguir é apresentada resumidamente uma simulação do processamento de $Query$ para a consulta de a para $\langle a,col(m_1)\rangle$.

- a emite a consulta $Query(p = \langle a,col(m_1)\rangle,\alpha = (p,a,\{r^{\mathcal{F}}_{\alpha 1},r^{\mathcal{F}}_{\alpha 2}\}),hist_p = [])$ para a
  - Retorno esperado: $Ans(\langle a, col(m_1)\rangle, \alpha, true, \{A^{\alpha}_1\}, \{A^{\alpha}_2, A^{\alpha}_3\})$
  - **Explicação**: $Query(p, ...) \Rightarrow$
    - $\Rightarrow Find\_Similar\_RLs(p,KB_{a\alpha})$ retorna $rlits = \{\langle a,col(m_1)\rangle\}$, que não está em $hist_p$ (linhas 3 a 5)
    - Seja $p' = \langle a,col(m_1)\rangle$ (linha 8):
      - $\Rightarrow$ $Local\_Ans(\langle a,col(m_1)\rangle)$ e $Local\_Ans(\langle a,\neg col(m_1)\rangle)$, ambos, retornam $false$ (linhas 10 a 15)
      - $\Rightarrow$ $Find\_Def\_Args(\langle a,col(m_1)\rangle,...)$ retorna $tv_{p'} = true$ e $Args_{p'} = \{A^{\alpha}_1\}$ (linha 17)
      - $\Rightarrow Find\_Def\_Args(\langle a,\neg col(m_1)\rangle,...)$ retorna $tv_{\sim p'} = true$ e $Args_{\sim p'} = \{A^{\alpha}_2, A^{\alpha}_3\}$ (linha 18). O processo será explicado na continuação do Exemplo 1.1 na Seção 4.1.1
      - Como não há argumentos que derrotam A^{α}_1, $tv_{p'}$ recebe $true$, assim como tv_p (linhas 22 a 24)
    - Não havendo outros p' similares o suficiente a p, retorna a resposta com $tv_p = true$ e os argumentos gerados (linha 27).

A seguir é apresentada resumidamente uma simulação do processamento de $Query$ para a consulta de a para o *er-literal* $\langle @,avl(m_1)\rangle$.

- c emite a consulta $Query(p = \langle @,avl(m_1)\rangle,\alpha = (p,a,\{r^{\mathcal{F}}_{\alpha 1},r^{\mathcal{F}}_{\alpha 2}\}),hist_p = [\langle a,col(m_1)\rangle,\langle c,ed(m_1)\rangle])$ para e
  - Retorno esperado: $Ans(\langle @, avl(m_1)\rangle, \alpha, true, \{[E^{\alpha}_1]\}, \emptyset)$
  - **Explicação**: $Query(p, ...) \Rightarrow$
    - $\Rightarrow Find\_Similar\_RLs(p,KB_{e\alpha})$ retorna $rlits = \{\langle e,spa(m_1)\rangle\}$, que não está em $hist_p$ (linhas 3 a 5)
    - Seja $p' = \langle e,spa(m_1)\rangle$ (linha 8):
      - $\Rightarrow Local\_Ans(\langle e,spa(m_1)\rangle)$ e $Local\_Ans(\langle e,\neg spa(m_1)\rangle)$, ambos, retornam $false$ (linhas 10 a 15)
      - $\Rightarrow Find\_Def\_Args(\langle e,spa(m_1)\rangle,...)$ retorna $tv_{p'} = true$ e $Args_{p'} = \{E^{\alpha}_1\}$ (linha 17)
      - $\Rightarrow Find\_Def\_Args(\langle e,\neg spa(m_1)\rangle,...)$ retorna $tv_{\sim p'} = false$ e $Args_{\sim p'} = \emptyset$ pois não existem regras com cabeça $\langle e,\neg spa(m_1)\rangle$ em $KB_{e\alpha}$ (linha 18)
      - Como não há argumentos que derrotam E^{α}_1, $tv_{p'}$ recebe $true$, assim como tv_p (linhas 22 a 24)
    - Não havendo outros p' similares o suficiente a p, retorna a resposta com $tv_p = true$ e os argumentos gerados (linha 27).

□

4.1.2 Procedimento $Find\_Def\_Args$

O procedimento $Find\_Def\_Args$ é apresentado no Algoritmo 4. Suas entradas e saídas já foram descritas na Seção 4.1.1.

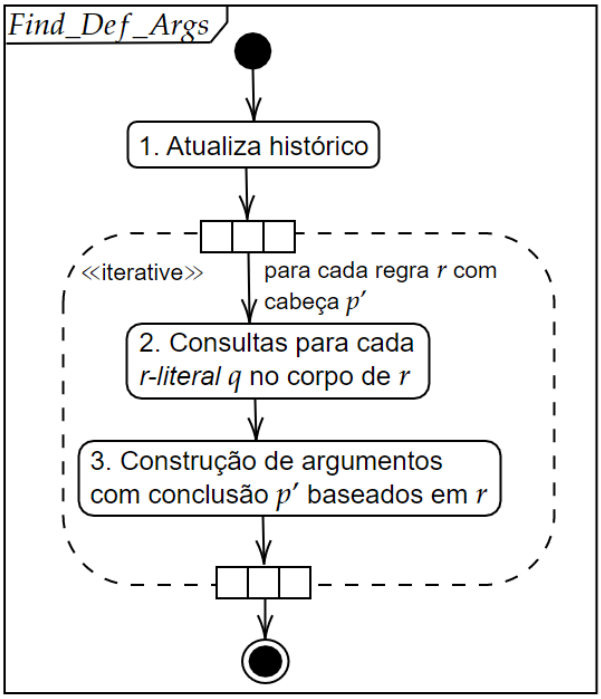
O procedimento é dividido em 3 (três) etapas:

1. Atualização do histórico com o p' a ser processado (linha 3)
2. Envio de consultas para cada *r-literal* q no corpo de r, para cada regra r com cabeça p' (linhas 7 a 12)

3. Construção de argumentos com conclusão p' dados os resultados das consultas para o corpo de r, para cada regra r com cabeça p' (linha 13)

A Figura 17 apresenta um diagrama de atividades que ilustra a execução dessas etapas.

Figura 17 – Diagrama de atividades para o algoritmo $Find\_Def\_Args$.



Fonte: Autoria própria.

Algoritmo 4 – Pseudocódigo do procedimento $Find\_Def\_Args$.

```
1  function Find_Def_Args (p′,KB_aα,α,hist_p) :
2      thread-safe var: Args_p′ ← ∅
3      hist_p′ ← [hist_p|p′]
4      executing in parallel for each r ∈ KB_aα s.t. Head(r) = p′
5          thread-safe var: possible_subargs_r ← ∅
6          executing in parallel for each q ∈ Body(r)
7              if D(q) = @ then
8                  Args_q ← Query_Agents(Ags,q,α,hist_p′)
9              else if D(q) ∈ Ags then
10                 Args_q ← Query_Agents({D(q)},q,α,hist_p′)
11             if Args_q = ∅ then stop and discard all processing for rule r
12             possible_subargs_r ← possible_subargs_r ∪ {Args_q}
13         Args_p′ ← Args_p′ ∪ Build_Def_Args_For_Rule(p′,possible_subargs_r)
14     return Args_p′
```

Fonte: Autoria própria.

Etapa 1: Atualização do histórico

Antes do processamento das regras ser iniciado, um histórico $hist_{p'}$ atualizado é criado anexando-se p' a $hist_p$ (linha 3). Isso permitirá que os agentes consultados em sequência detectem um argumento falacioso caso o mesmo *r-literal* p' apareça.

Etapa 2(a): Consultas para cada antecedente de uma regra

A intuição de $Find\_Def\_Args$ é iterar sobre cada regra r com cabeça p' tentando construir argumentos para ela. A iteração é paralelizada, tanto para cada regra quanto para cada membro do corpo da regra. Isso é útil, pois cada *thread* de execução emitirá novas consultas para outros agentes, o que envolve a espera por suas respostas.

Para cada $q \in Body(r)$, uma decisão deve ser tomada quanto a quais agentes consultar. Se q é um *er-literal* (ou seja, seu definidor é igual a @), então todos os agentes, incluindo o próprio a, devem ser consultados (linhas 7 e 8). Caso contrário, se q for definido por um agente específico, apenas este agente deve ser consultado (linhas 9 e 10). As consultas são emitidas por meio de uma função auxiliar $Query\_Agents$, cuja entrada é um conjunto de identificadores para agentes, o *r-literal* q, e também α e $hist_p$, que são necessários para as consultas subsequentes, como já foi explicado.

Este trabalho não se aprofunda em detalhes sobre tecnologias e infraestruturas utilizadas, mas pode-se supor que exista algum meio de comunicação entre os agentes que, além de permitir comunicação direta entre agentes, também permite o envio de mensagens *broadcast* indiscriminadamente (AZUMA *et al.*, 2013), de modo que um agente não precisaria ser capaz de referenciar individualmente todos os agentes no ambiente, o que requereria que o agente obtivesse uma lista de todos os agentes no sistema por meio de algum serviço de páginas brancas fornecido pela infraestrutura que dá suporte à aplicação. No entanto, por simplicidade de apresentação do pseudocódigo, optou-se por representar esse envio de consultas em *broadcast* como se o agente tivesse acesso a uma relação de agentes no ambiente.

O pseudocódigo para $Query\_Agents$ é apresentado no Algoritmo 6. Ele retorna um conjunto de argumentos $Args_q$, que é a união de todos os conjuntos $Args'_q$ resultantes das consultas a cada agente. Quando uma consulta é enviada (linha 4) a um agente a, a *thread* de execução entra em estado de espera pela resposta. Em seguida, os argumentos recebidos do agente são armazenados no conjunto $Args_q$ (linha 8). Após todos os agentes retornarem suas

respostas (ou um limite de tempo for atingido, o que pode ser configurado pelo desenvolvedor da aplicação), o conjunto $Args_q$ é retornado.

Algoritmo 5 – Pseudocódigo do procedimento $Query\_Agents$.

```
1 function Query_Agents (agents,q,α,hist_q) :
2     thread-safe var: Args_q ← ∅
3     executing in parallel for each a ∈ agents do
4         send to a: Query(q,α,hist_q) and wait for response:
            Ans(q,α,tv'_q,Args'_q,Args'_~q), else if timeout reached then discard
5         Args_q ← Args_q ∪ Args'_q
6     return Args_q
```

Fonte: Autoria própria.

De volta à função $Find\_Def\_Args$, ao receber de $Query\_Agents$ um conjunto vazio de argumentos para um *r-literal* q, então todo o processamento para a regra r é descartado (linha 11), uma vez que não é possível construir um argumento baseado em uma regra se não houver subargumentos para todos os membros de seu corpo. Dessa forma, se o processamento para os membros restantes do corpo dessa regra ainda estiver em andamento, as respostas recebidas para eles podem ser simplesmente ignoradas. Para evitar mais processamento inútil por parte dos outros agentes que foram consultados para os outros antecedentes da regra, o agente poderia enviar uma mensagem sinalizando a todos os agentes para que interrompam sua execução naquela linha específica de raciocínio, mas isso é uma otimização técnica que não é apresentada no algoritmo para evitar complexidade extra em sua apresentação. Caso o conjunto não seja vazio, adiciona-se esse conjunto a $possible\_subargs_r$. Tal variável, criada individualmente para cada regra processada, armazena os conjuntos de argumentos possíveis para cada membro do corpo – que são possíveis subargumentos para os argumentos a serem criados para p' – servindo, portanto, de insumo para a próxima etapa.

Etapa 2(b): Construção dos argumentos para uma regra

Após todos os membros do corpo da regra r terem sido avaliados, é necessário criar argumentos com conclusão p' (a cabeça da regra) com base nos subargumentos que foram construídos para os membros do corpo da regra. Isto é feito por meio de uma função auxiliar $Build\_Def\_Args\_For\_Rule$ (Algoritmo 23), chamada na linha 13, que recebe o *r-literal* p' e o conjunto de subargumentos possíveis $possible\_subargs_r$, e retorna um conjunto de argumentos baseados na regra r, $Args_r$.

Algoritmo 6 – Pseudocódigo da função $Build\_Def\_Args\_For\_Rule$.

```
1  function Build_Def_Args_For_Rule (p',possible_subargs_r) :
2      Args_r ← ∅
3      if possible_subargs_r = ∅ then
4          A_p' ← New_Arg(p', [⊤], defeasible)
5          A_p'.SuppJ ← true
6          return {A_p'}
7      for subargs_combination ∈ ∏_{Args_q ∈ possible_subargs_r} Args_q do
8          A_p' ← New_Arg(p',[], defeasible)
9          for A_q' ∈ subargs_combination do
10             q' ← Conc(A_q')
11             if D(q') ≠ a or Θ(q,q') < 1 then
12                 q_inst ← ⟨D(q'),L(q'),Θ(q,q')⟩
13                 A_inst ← New_Arg(q_inst, Sub_Args(A_q'), defeasible)
14                 Add_Sub_Arg(A_p', A_inst)
15             else
16                 Add_Sub_Arg(A_p', A_q')
17             if ∃A_q' ∈ Sub_Args(A_p') s.t. A_q'.R = true then
18                 A_p'.R ← true
19             else if ∀A_q' ∈ Sub_Args(A_p'), A_q'.J = true then
20                 A_p'.SuppJ ← true
21         A_p'.St ← StArg(A_p')
22         Args_r ← Args_r ∪ {A_p'}
23     return Args_r
```

Fonte: Autoria própria.

Se $possible\_subargs_r$ for vazio, então significa que o corpo da regra r é vazio (linha 3 de $Build\_Def\_Args\_For\_Rule$). Logo, um argumento base derrotável para p' pode ser criado (linha 4) e retornado pela função. Além disso, como um argumento base sempre é apoiado por qualquer conjunto, marca-se a propriedade $SuppJ$ do argumento como $true$. Se o corpo da regra não for vazio, realiza-se a construção de argumento com base nas combinações possíveis de subargumentos referentes a cada membro do corpo da regra. A intuição é que, para cada combinação de argumentos tomados de cada conjunto de argumentos em $possible\_subargs_r$, um novo argumento com a conclusão p' deve ser criado contendo, como subargumentos, os argumentos contidos nessa combinação de argumentos. Portanto, é calculado o produto cartesiano entre todas os conjuntos de argumentos ($Args_q$) em $possible\_subargs_r$, formando um conjunto de combinações de argumentos (linha 6). A continuação do Exemplo 3, apresentada em sequência, ilustra a geração dessas combinações de subargumentos.

Para cada uma dessas combinações, denotadas $subargs\_combination$, um novo argumento derrotável $A_{p'}$ com conclusão p' é criado (linha 7). Para cada $A_{q'} \in$

$subargs\_combination$, verifica-se se o definidor da conclusão q' do argumento $A_{q'}$ é diferente do próprio agente a, ou se a similaridade entre q' e q é menor que 1 (linha 10). Se for o caso, então é necessário criar um *ir-literal* q_{inst} e um subargumento A_{inst}, que é uma cópia do argumento $A_{q'}$, porém com cabeça q_{inst} (linhas 11 e 12). Por fim, adiciona-se A_{inst} como subargumento de $A_{p'}$ (linha 13). Caso contrário, isto é, se q' for um *cr-literal* local e idêntico a q, então apenas adiciona-se $A_{q'}$ como subargumento de A (linhas 14 e 15).

Em seguida, verifica-se se algum subargumento de $A_{p'}$ está marcado como rejeitado (linha 18). Se for o caso, $A_{p'}$ também é marcado como rejeitado, por meio da propriedade R, visto que basta existir um subargumento rejeitado para que um argumento seja também rejeitado, como apresentado na Definição 20. Caso contrário, verifica-se se todos os subargumentos estão marcados como justificados. Se for o caso, $A_{p'}$ é marcado como apoiado por argumentos justificados, por meio da propriedade $SuppJ$. Essas verificações e marcações, realizadas na linha 5 e das linhas 17 a 20, visam definir previamente o *status* de alguns argumentos, de modo a facilitar a comparação entre conjuntos de argumentos realizada na Etapa 6 de $Query$, apresentada na Seção 4.1.1.

Finalmente, a função $StArg$ (ver Definição 12) é utilizada para calcular a força do argumento (linha 20), que é então armazenada na propriedade St do argumento, pois, como a função de cálculo padrão é recursiva e a força de cada *ir-literal* é calculado do ponto de vista do agente que constrói localmente um argumento, é possível reaproveitar o valor da força já calculado de cada subargumento instanciado direto (capturados por meio da função $DISA$, conforme a Definição 7) para calcular o valor da força para um argumento.

De volta a $Find\_Def\_Args$, uma vez gerados todos os argumentos possíveis para p', eles são retornados na linha 14.

Continuação do Exemplo 1.1. (Veja o Exemplo 1 na página 68, o Exemplo 1.1 na página 72, e suas continuações nas páginas 80, 87, 97 e 122). A seguir é apresentada uma simulação resumida do processamento derivado da chamada $Find\_Def\_Args(\langle a,\neg col(m_1)\rangle, KB_{a\alpha},\alpha,[])$

- Retorno esperado: $Args_{p'} = \{A_2^\alpha, A_3^\alpha\}$
- **Explicação**: $Find\_Def\_Args(\langle a,\neg col(m_1)\rangle,...) \Rightarrow$
  - Para $q = \langle @,\neg ed(m_1)\rangle$, onde $q \in Body(r_{a3})$ (linha 6, $Find\_Def\_Args$):
    - $\Rightarrow$ $Query\_Agents(\{a,b,c,d,e\},q,\alpha,[\langle a,\neg col(m_1)\rangle])$ (linha 8, $Find\_Def\_Args$):

· Chama $Query$ para *r-literal* q para todos os agentes. Todos respondem $false$ imediatamente por não existirem *r-literais* similares, exceto os agentes b e d (linhas 4 e 5, $Query\_Agents$)

· $\Rightarrow$ b recebe $Query(\langle @,\neg ed(m_1)\rangle,\alpha,[\langle a,\neg col(m_1)\rangle])$ e retorna $Ans(\langle @,\neg ed(m_1)\rangle, \alpha, true, Args'_q = \{B_1^\alpha\}, Args'_{\sim q} = \emptyset)$. Os argumentos em $Args'_q$ são adicionados a $Args_q$ (linhas 4 e 5, $Query\_Agents$)

· $\Rightarrow$ d recebe $Query(\langle @,\neg ed(m_1)\rangle,\alpha,[\langle a,\neg col(m_1)\rangle])$ e retorna $Ans(\langle @,\neg ed(m_1)\rangle, \alpha, true, Args''_q = \{D_1^\alpha\}, Args''_{\sim q} = \emptyset)$. Os argumentos em $Args''_q$ são adicionados a $answers_q$ (linhas 4 e 5, $Query\_Agents$)

· $\Rightarrow$ $Args_q = \{B_1^\alpha, D_1^\alpha\}$ é retornado por $Query\_Agents$ (linha 6, $Query\_Agents$)

* $\Rightarrow$ $Args_q$ para $q = \langle @, \neg ed(m_1)\rangle$ é adicionado a $possible\_subargs_r$ referente a r_{a3} (linha 12, $Find\_Def\_Args$)

* $\Rightarrow$ $Build\_Def\_Args\_For\_Rule(p',possible\_subargs_r)$ (linha 13, $Find\_Def\_Args$):

· Apenas duas combinações de argumentos baseadas em $possible\_subargs_r$: $\{B_1^\alpha\}$, $\{D_1^\alpha\}$ (linha 6, $Build\_Def\_Args\_For\_Rule$)

· Dois argumentos são gerados a partir de cada combinação: A_2^α e A_3^α (linhas 8 a 22, $Build\_Def\_Args\_For\_Rule$)

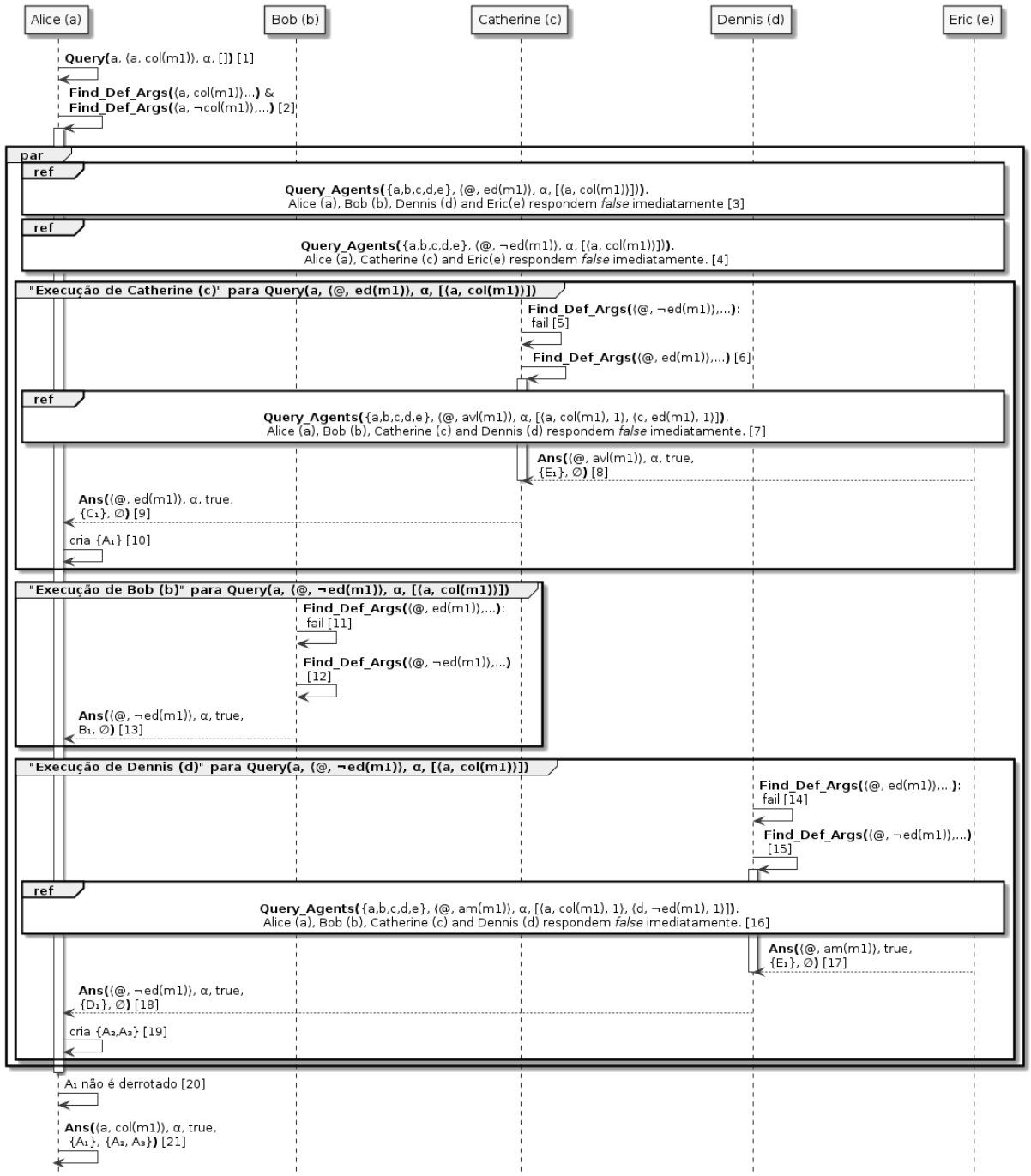
* A_2^α e A_3^α são adicionadas a $Args_{p'}$ (linha 13, $Find\_Def\_Args$)

– Não havendo mais regras com cabeça p', é retornado $Args_{p'} = \{A_2^\alpha, A_3^\alpha\}$ (linha 4 e 14, $Find\_Def\_Args$)

A Figura 18 apresenta um diagrama de sequência que ilustra a execução completa do foco de consulta α do Exemplo 1.1.

A execução começa com o agente de Alice enviando uma consulta para $col(m_1)$ a si mesmo (1) e chamando a função $Find\_Def\_Args$ para ambos $col(m_1)$ e $\neg col(m_1)$ (2). A regra r_{a2} com cabeça $\langle a, col(m_1)\rangle$ é encontrada com um único antecedente $\langle @, ed(m_1)\rangle$, e a regra r_{a3} com cabeça $\langle a, \neg col(m_1)\rangle$ é encontrado com um único antecedente $\langle @, \neg ed(m_1)\rangle$. Como são *er-literais*, todos os agentes conhecidos, incluindo a própria Alice, são consultados por $\langle @, ed(m_1)\rangle$ e $\langle @, \neg ed(m_1)\rangle$ (3, 4). Exceto no agente de Charles, todas as consultas para $\langle @, ed(m_1)\rangle$ falham (retornam um valor-verdade $false$) porque nenhum *r-literal* similar é encontrado nas cabeças

Figura 18 – Diagrama de sequência para a execução do Exemplo 1.1.



Fonte: Autoria própria.

das regras dos agentes; e exceto para Barb e Dennis, o mesmo ocorre para as consultas para $\langle @, \neg ed(m_1)\rangle$.

O agente de Charles (c) recebe a consulta para $\langle @, ed(m_1)\rangle$ e chama $Find\_Def\_Args$ para $\langle c, ed(m_1)\rangle$ e $\langle c, \neg ed(m_1)\rangle$. O último falha porque não há regra em c com cabeça $\langle c, \neg ed(m_1)\rangle$ [5], mas para o primeiro a regra r_{c1} é encontrada com um único antecedente $\langle @, avl(m_1)\rangle$ (6). Por ser um *er-literal*, todos os agentes conhecidos, incluindo o próprio c, são consultados por $\langle @, avl(m_1)\rangle$ (7). Exceto para o agente de Eric, todas as consultas falham porque nenhum *r-literal* similar é encontrado. O agente de Eric encontra um *r-literal* similar $\langle e, spa(m_1)\rangle$ com grau de similaridade $\theta = 0{,}8$ e chama $Find\_Def\_Args$ para $\langle e, spa(m_1)\rangle$ e $\langle e, \neg spa(m_1)\rangle$. O último falha, mas o primeiro é capaz de retornar uma resposta com valor-verdade $true$ e $Args_{\langle e,spa(m_1)\rangle} = \{E_1^\alpha\}$, dada a regra r_{e1} e as regras de foco $r_{\alpha1}^{\mathcal{F}}$ e $r_{\alpha2}^{\mathcal{F}}$ que são provadas localmente, retornando assim uma resposta verdadeira a Charles (8). Como não há conflito no processamento da consulta do agente de Charles, ele retorna a Alice uma resposta $true$ com o conjunto de argumentos $Args_{\langle c,ed(m_1)\rangle} = \{C_1^\alpha\}$ (9), indicando que descobriu que $ed(m_1)$ é verdadeiro através do argumento que inclui, como subargumento, aquele fornecido por Eric. O agente de Alice então cria o conjunto de argumentos $Args_{\langle a,col(m_1)\rangle} = \{A_1^\alpha\}$ e espera pelas respostas de $Find\_Def\_Args$ para $\langle a, \neg col(m_1)\rangle$ (10).

O agente de Barb recebe a consulta para $\langle @, \neg ed(m_1)\rangle$ e chama $Find\_Def\_Args$ para $\langle b, \neg ed(m_1)\rangle$ e $\langle b, ed(m_1)\rangle$. O último falha porque não há regra em Barb com cabeça $\langle b, ed(m_1)\rangle$ (11), mas para o primeiro a regra r_{b1} é encontrada com um único antecedente $\langle b, hv(m_1)\rangle$, que é provado localmente pela regra de foco $r_{\alpha1}^{\mathcal{F}}$ (12). Portanto, Barb retorna uma resposta $true$ para $\langle @, \neg ed(m_1)\rangle$ para Alice (13) com $Args_{\langle b,\neg ed(m_1)\rangle} = \{B_1^\alpha\}$, uma vez que a resposta de Barb é baseada apenas em seu conhecimento local juntamente com o conhecimento de foco do foco da consulta. Supondo que Barb responda antes de Dennis, Alice armazena a resposta e espera pela resposta restante de Dennis.

O agente de Dennis recebe a consulta para $\langle @, \neg ed(m_1)\rangle$ e chama $Find\_Def\_Args$ para $\langle d, \neg ed(m_1)\rangle$ e $\langle d, ed(m_1)\rangle$. O último falha porque não há regra em Dennis com cabeça $\langle d, ed(m_1)\rangle$ (14), mas para o primeiro a regra r_{d1} é encontrada com um único antecedente $\langle @, am(m_1)\rangle$ (15). Por ser um *er-literal*, todos os agentes conhecidos, incluindo o próprio Dennis, são consultados por $\langle @, am(m_1)\rangle$ (16). Exceto para Eric, todas as consultas falham porque nenhum *r-literal* similar foi encontrado. Eric encontra um *r-literal* similar $\langle e, spa(m_1)\rangle$ com grau de similaridade $\theta = 0{,}4$ e chama $Find\_Def\_Args$ para $\langle e, spa(m_1)\rangle$ e $\langle e, \neg spa(m_1)\rangle$,

com os mesmos resultados (incluindo a mesmo conjunto de argumentos $Args_{\langle e,spa(m_1)\rangle} = \{E_1^\alpha\}$) dados a Charles anteriormente, retornando assim uma resposta $true$ a Dennis (17). Como não há conflito no processamento da consulta de Dennis, ele retorna a Alice uma resposta $true$ com o conjunto de argumentos $Args_{\langle d,\neg ed(m_1)\rangle} = \{D_1^\alpha\}$, indicando que encontrou que $\neg ed(m_1)$ é verdadeiro através do argumento que inclui, como subargumento, aquele fornecido por Eric. Tendo as respostas para $\langle @, \neg ed(m_1)\rangle$ de todos os agentes, Alice então cria o conjunto de argumentos para $Args_{\langle a,\neg col(m_1)\rangle} = \{A_2^\alpha, A_3^\alpha\}$ (19).

Finalmente, Alice compara os conjuntos de argumentos para $\langle a, col(m_1)\rangle$ e $\langle a, \neg col(m_1)\rangle$ (20), que são os resultados retornados das chamadas a $Find\_Def\_Args$ em (2). Ambos os resultados têm um valor-verdade $true$, e o único argumento para $\langle a, col(m_1)\rangle$ é mais forte do que ambos os argumentos para $\langle a, \neg col(m_1)\rangle$. Portanto, a resposta $true$ é retornada à consulta original feita por Alice, incluindo os conjuntos de argumentos que concluem $\langle a, col(m_1)\rangle$ e $\langle a, \neg col(m_1)\rangle$ (21).

□

Continuação do Exemplo 3. (Veja o Exemplo 3 na página 82 e suas continuações nas páginas 88 e 91). A fim de ilustrar mais claramente como funciona a geração dos argumentos, principalmente quando vários podem ser derivados de uma mesma regra, apresenta-se uma simulação da chamada a $Find\_Def\_Args$ para o literal $\langle a,x_1\rangle$.

- Retorno esperado: $Args_{p'} = \{A_1^\delta, A_3^\delta, A_4^\delta, A_5^\delta, A_7^\delta, A_8^\delta\}$
- **Explicação**: $Find\_Def\_Args(\langle a,x_1\rangle,...) \Rightarrow$
  - Para $r = r_{a1}$:
    - Para $q = \langle @,x_2\rangle$, onde $q \in Body(r_{a1})$ (linha 6, $Find\_Def\_Args$):
      - $\Rightarrow Query\_Agents(\{a,b,c,d\},q,\delta,[\langle a,x_1\rangle])$ (linha 8, $Find\_Def\_Args$):
      - Chama $Query$ para *r-literal* q para todos os agentes. Todos respondem $false$ imediatamente por não existirem *r-literais* similares, exceto o agente b (linhas 4 e 5, $Query\_Agents$)
      - $\Rightarrow$ b recebe $Query(\langle @,x_2\rangle,\delta,[\langle a,x_1\rangle])$ e retorna $Ans(\langle @,x_2\rangle, \delta, true, Args'_q = \{B_1^\delta\}, Args'_{\sim q} = \emptyset)$. $(true,Args'_q)$ é adicionado a $answers_q$ (linhas 4 e 5, $Query\_Agents$)
      - $Args_q = \{B_1^\delta\}$ (linha 7, $Query\_Agents$)

  - $Query\_Agents$ retorna $Args_q$ (linha 6, $Query\_Agents$)

* $Args_q$ para $q = \langle @,x_2 \rangle$ é adicionado a $possible\_subargs_r$ referente a r_{a1} (linha 12, $Find\_Def\_Args$)

* Para $q = \langle @,x_3 \rangle$, onde $q \in Body(r_{a1})$ (linha 6, $Find\_Def\_Args$):
  - $\Rightarrow Query\_Agents(\{a,b,c,d\},q,\delta,[\langle a,x_1 \rangle])$ (linha 8, $Find\_Def\_Args$):
  - Chama $Query$ para ***r-literal*** q para todos os agentes. b responde $false$ imediatamente por não existirem ***r-literais*** similares, enquanto c e d respondem $true$ (linhas 4 e 5, $Query\_Agents$)
  - $\Rightarrow$ c recebe $Query(\langle @,x_3 \rangle,\delta,[\langle a,x_1 \rangle])$. Supondo que nessa consulta são geradas os conjuntos de argumentos $Args'_q = \{C_1^\delta, C_2^\delta, C_3^\delta, C_4^\delta, C_5^\delta\}$ e $Args'_{\sim q} = \{C_6^\delta\}$), e que pelo menos um dentre C_2^δ, C_3^δ e C_5^δ não é derrotado por C_6^δ (C_1^δ e C_4^δ não são considerados por serem falaciosos), a consulta retorna $Ans(\langle @,x_3 \rangle, \delta, true, Args'_q, Args'_{\sim q})$. Os argumentos em $Args'_q$, portanto, são adicionados a $Args_q$ (linhas 4 e 5, $Query\_Agents$)
  - $\Rightarrow$ d recebe $Query(\langle @,x_3 \rangle,\delta,[\langle a,x_1 \rangle])$. Supondo que nessa consulta são geradas os conjuntos de argumentos $Args''_q = \{D_5^\delta\}$ e $Args''_{\sim q} = \emptyset$, a consulta retorna $Ans(\langle @,x_3 \rangle, \delta, true, Args''_q, Args''_{\sim q})$. Os argumentos em $Args''_q$, portanto, são adicionados a $Args_q$ (linhas 4 e 5, $Query\_Agents$)
  - $Query\_Agents$ retorna $Args_q$ (linha 6, $Query\_Agents$)

* $Args_q$ para $q = \langle @,x_3 \rangle$ é adicionado a $possible\_subargs_r$ referente a r_{a1} (linha 12, $Find\_Def\_Args$)

* $\Rightarrow$ $Build\_Def\_Args\_For\_Rule(p',possible\_subargs_r)$ (linha 13, $Find\_Def\_Args$):
  - Conjunto de combinações de argumentos para ambos os antecedentes de r_{a1}, resultante do produtório cartesiano da linha 7 de $Build\_Def\_Args\_For\_Rule$: $\{(B_1^\delta,C_1^\delta), (B_1^\delta,C_2^\delta), (B_1^\delta,C_3^\delta), (B_1^\delta,C_4^\delta)(B_1^\delta,C_5^\delta), (B_1^\delta,D_5^\delta)\}$, a partir das quais são construídos os argumentos $Args_r = \{A_1^\delta, A_3^\delta, A_4^\delta, A_5^\delta, A_7^\delta, A_8^\delta\}$, todos concluindo $\langle a,x_1 \rangle$. Veja a Figura 19 para uma ilustração gráfica da geração desse conjunto de combinações.

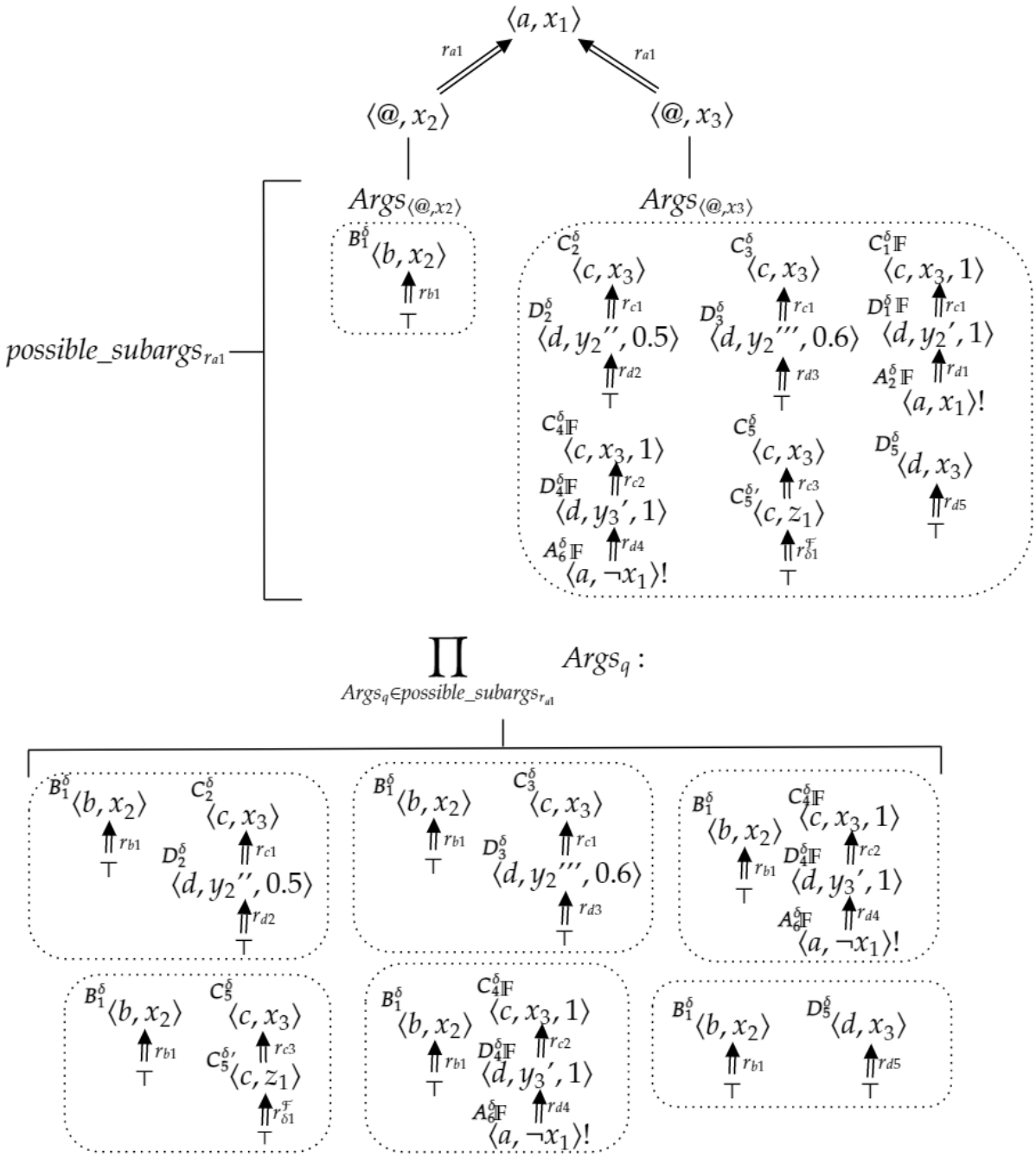
* Os argumentos em $Args_r$ para r_{a1} são adicionados a $Args_{p'}$ (linha 14,

$Find\_Def\_Args$)

– Não havendo mais regras com cabeça p', é retornado $Args_{p'} = \{A_1^\delta, A_3^\delta, A_4^\delta, A_5^\delta, A_7^\delta, A_8^\delta\}$ (linhas 4 e 15, $Find\_Def\_Args$)

□

Figura 19 – Geração das combinações de subargumentos para gerar os argumentos baseados na regra r_{a1} do Exemplo 3.



Fonte: Autoria própria.

4.2 PROPRIEDADES DO ALGORITMO E OTIMIZAÇÕES

Esta seção apresenta algumas propriedades do algoritmo, como terminação (Seção 4.2.1), corretude e completude (Seção 4.2.2) e complexidade computacional (Seção 4.2.3).

4.2.1 Terminação

É garantido que o algoritmo termina graças à detecção de ciclos baseada em histórico, conforme apresentado no Teorema 4. A prova é apresentada no Apêndice A.

Teorema 4 (Terminação do Algoritmo)**.** *O algoritmo termina em tempo finito retornando um dos valores* $true$*,* $false$ *ou* $undec$ *para o r-literal consultado* p *e conjuntos de argumentos para r-literais similares a* p *e* $\sim p$*.*

4.2.2 Corretude e Completude

Os teoremas 5 e 6 associam os resultados calculados por $Query$ e $Local\_Ans$ aos conceitos do *framework* de argumentação. Especificamente, o Teorema 5 fornece uma caracterização das respostas retornadas especificamente pela função $Local\_Ans$, com base na existência de argumentos estritos, enquanto o Teorema 6 associa os valores-verdade retornados por $Query$ aos conceitos de *r-literais* justificados e rejeitados. As provas para essas proposições são apresentadas no Apêndice A.

Teorema 5 (Corretude e Completude do Procedimento $Local\_Ans$)**.** *Para um SMA* $\mathcal{S} = (Ags, F_Q, \Theta, st)$*, um r-literal* p*, e um agente* $a \in Ags$*, tal que existe* $p' \in V^R_{a\alpha}$ *similar o suficiente a* p*,* $Local\_Ans(p')$ *retorna*

1. $true$ *se e somente se existe um argumento estrito* $A \in Args_a$ *para* p'
2. $false$ *se e somente não existe um argumento estrito* $A \in Args_a$ *para* p'

Teorema 6 (Corretude e Completude do Procedimento $Query$)**.** *Para um SMA* $\mathcal{S} = (Ags, F_Q, \Theta, st)$*, um r-literal* p*, um agente* $a \in Ags$ *e um foco de consulta* $\alpha = (p,a',KB^{\mathcal{F}}_{\alpha}) \in F_Q$*,* $Query(p,\alpha,hist_p)$ *retorna*

1. $true$ *se e somente se existe* $p' \in V^R_{a\alpha}$ *similar o suficiente a* p *que é justificado em* $KB_{\mathcal{S}\alpha}$
2. $false$ *se e somente, para todo* $p' \in V^R_{a\alpha}$ *similar o suficiente a* p*,* p' *é rejeitado em* $KB_{\mathcal{S}\alpha}$

3. $undec$ *se e somente se não existe* $p' \in V^R_{a\alpha}$ *similar o suficiente a* p *que é justificado em* $KB_{\mathcal{S}\alpha}$*, mas existe* $p' \in V^R_{a\alpha}$ *que não é nem justificado nem rejeitado em* $KB_{\mathcal{S}\alpha}$.

4.2.3 Complexidade Computacional e Número de Mensagens

A complexidade computacional e o número de mensagens serão analisados por meio de um cenário de pior caso (Seção 4.2.3.1). O desempenho será então melhorado por uma otimização baseada em *cache* (Seção 4.2.3.2) e a complexidade medida novamente na Seção 4.2.3.3.

4.2.3.1 Cenário de Pior Caso

Antes de avaliar a complexidade, é necessário definir um tamanho de entrada n. Seja $\mathcal{S}$ o SMA e α o foco de consulta em questão, $V^P_{\mathcal{S}\alpha} = \bigcup_A^{Args_{\mathcal{S}\alpha}} Prem(A)$ é o conjunto de *r-literais* que são premissas dos argumentos que podem ser derivados. O tamanho da entrada é definido como $n = |V^P_{\mathcal{S}\alpha}|$, ou seja, a quantidade total de premissas dos argumentos, o que coincide também com as conclusões dos subargumentos próprios de todos os argumentos gerados. Por exemplo, nos argumentos gerados para o Exemplo 1.1, $V^P_{\mathcal{S}\alpha} = \{\langle c, ed(m_1), 1\rangle, \langle b, \neg ed(m_1), 1\rangle, \langle d, \neg ed(m_1), 1\rangle, \langle b, hv(m_1)\rangle, \langle e, hv(m_1)\rangle, \langle e, pbc(m_1)\rangle, \langle e, spa(m_1), 0{,}8\rangle, \langle e, spa(m_1), 0{,}4\rangle\}$, de modo que $n = |V^P_{\mathcal{S}\alpha}| = 8$. A razão para essa escolha é que o número de chamadas recursivas para $Query$ está diretamente relacionado à quantidade de premissas nos argumentos, e essas premissas nem sempre são proporcionais à quantidade de *r-literais* e/ou regras na base de conhecimento global, visto que possivelmente existem muitos *ir-literais* que podem ser gerados a partir de um único *er-literal* por meio da função de similaridade – por exemplo, observa-se que $\langle e, spa(m_1), 0{,}8\rangle$ e $\langle e, spa(m_1), 0{,}4\rangle$ são gerados a partir da correspondência do *er-literal* $\langle @, spa(m1)\rangle$ com dois *r-literais* diferentes. Portanto, $n = |V^P_{\mathcal{S}\alpha}|$ é uma boa medida dos parâmetros do sistema, uma vez que incorpora, não apenas a quantidade de *r-literais* e/ou regras, mas também o impacto da função de similaridade e do limite de similaridade. Além disso, é possível que nem todos os *r-literais* do sistema sejam usados no processo de raciocínio para uma determinada consulta, como foi demonstrado no Exemplo 1.2.

O pior cenário ocorre quando o agente emissor tem uma regra cujo corpo contém *er-literais* correspondentes a todos os *r-literais* do sistema, exceto o *r-literal* na cabeça da regra,

e todos os outros agentes têm regras com um número decrescente de premissas no corpo de modo que nenhum ciclo seja introduzido, uma vez que os ciclos não induzem novas chamadas recursivas, não tendo impacto significativo na complexidade. Um exemplo com $n = 3$ é dado a seguir:

$$r_{11} : \langle a, x_1\rangle \Leftarrow \langle @, x_2\rangle, \langle @, x_3\rangle, \langle @, x_4\rangle; \qquad r_{21} : \langle b, x_2\rangle \Leftarrow \langle @, x_3\rangle, \langle @, x_4\rangle$$

$$r_{31} : \langle c, x_3\rangle \Leftarrow \langle @, x_4\rangle; \qquad r_{41} : \langle d, x_4\rangle \Leftarrow$$

É fácil notar que os *ir-literais* $(b, x_2, 1)$, $(c, x_3, 1)$ e $(d, x_4, 1)$ serão gerados, constituindo o conjunto de *ir-literais* dos argumentos, o que neste caso coincide com o conjunto total de premissas distintas nos argumentos. A Figura 20 apresenta uma árvore de chamadas, considerando apenas as consultas que atingem o estágio de $Find\_Def\_Args$ (etapa 5 do algoritmo $Query$), o qual leva a chamadas recursivas ao procedimento $Query$. Casos em que as consultas não atingem esse estágio não induzem recursividade sobre as regras das bases de conhecimento dos agentes, a saber: quando o agente consultado não possui um *r-literal* similar em sua base de conhecimento, ou quando o *r-literal* consultado ou seu complemento está no histórico $hist_p$. Por simplicidade, portanto, considera-se o critério da análise de complexidade como a quantidade de chamadas a $Query$ que alcançam o estágio de $Find\_Def\_Args$.

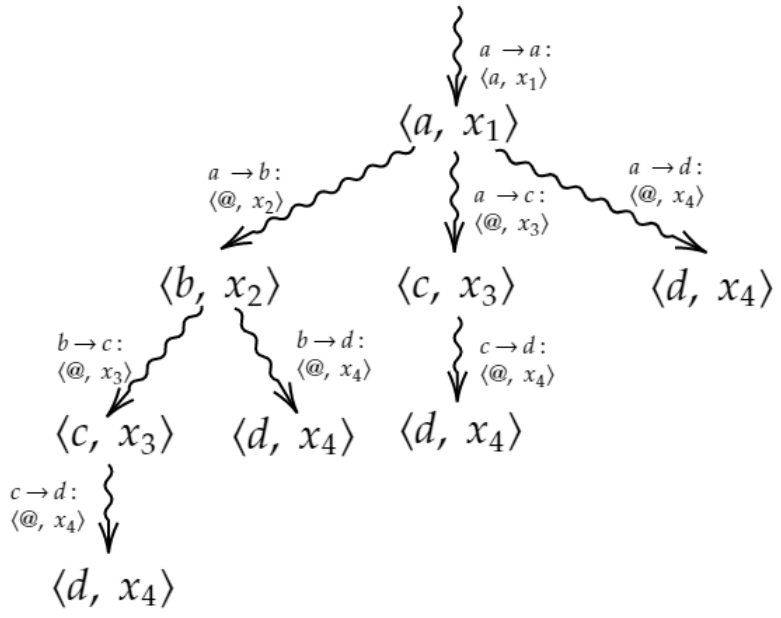
Cada aresta da árvore de chamadas apresentada na Figura 20 é rotulada com a forma $a \rightarrow b : p$, onde a é o agente que submeteu a consulta, b é o agente que recebeu a consulta, e p é o *r-literal* que está sendo consultado. Cada vértice representa o *r-literal* p' que foi achado similar o suficiente a p, e para o qual será buscada uma resposta.

A partir desse exemplo, observa-se que o número de consultas que alcançam o estágio de $Find\_Def\_Args$ para um número arbitrário de premissas nos argumentos é assintoticamente $O(2^n)$. Isso pode ser demonstrado por indução, considerando $n = 3$ um caso base válido. Suponha então $n = k - 1$. Sua árvore de chamadas terá uma raiz $\langle a_{k-1}, x_{k-1}\rangle$ com $k - 2$ subárvores, a primeira para $\langle a_{k-2}, x_{k-2}\rangle$, a segunda para $\langle a_{k-3}, x_{k-3}\rangle$, e assim por diante até $\langle a_1, x_1\rangle$. Somando-se o número de chamadas de cada subargumento mais a chamada à raiz, tem-se $1 + 2^{k-2} + 2^{k-3} + ... + 2^1 + 2^0 = 1 + 2^{k-1} - 1 = 2^{k-1}$. Dado isso, supondo $n = k$, então ele tem uma raiz $\langle a_k, x_k\rangle$ com $k - 1$ subargumentos, a primeira para $\langle a_{k-1}, x_{k-1}\rangle$, o segundo para $\langle a_{k-2}, x_{k-2}\rangle$, até $\langle a_1, x_1\rangle$. Somando-se o número de chamadas de cada subargumento mais a chamada à raiz, tem-se $1 + 2^{k-1} + (2^{k-2} + 2^{k-3}... + 2^1 + 2^0) = 2^k$. Do resultado anterior para

$n = k - 1$, sabe-se que $(2^{k-2} + 2^{k-3}... + 2^1 + 2^0) = 2^{k-1} - 1$. Assim, é possível calcular o número de chamadas para $n = k$ usando: $1 + 2^{k-1} + (2^{k-1} - 1) = 2 \times 2^{k-1} = 2^k$.

Com relação ao número de mensagens, é ainda maior considerando o número de mensagens que cada agente envia para cada *er-literal* e para cada agente conhecido. Isso porque, em cada uma das consultas, representadas pelas arestas na árvore de chamadas, como apresentado na Figura 20, deve-se considerar que o agente, na realidade, envia mensagens de consulta para cada agente conhecido sobre o *er-literal*. Por exemplo, o agente a, a fim de buscar apoio de outros agentes para o *er-literal* $\langle @,x_2 \rangle$, envia a consulta para b, c e d. Apenas b continua o processamento, visto que somente ele possui um *cr-literal* similar o suficiente a $\langle @,x_2 \rangle$ que é cabeça de uma regra em b. No entanto, as mensagens para c e d também são enviadas. O mesmo ocorre quando o agente b busca apoio de outros agentes para o *er-literal* $\langle @,x_3 \rangle$: ele enviará mensagens para a, c e d, embora somente d realizará um processamento que chegue ao estágio de $Find\_Def\_Args$. Portanto, o número de mensagens trocadas no cenário de pior caso apresentado, incluindo as mensagens de resposta de cada agente que recebe uma consulta, é $2^n \times (|Ags| - 1) \times 2$, isto é, para cada consulta que chega ao estágio de $Find\_Def\_Args$, o que já se constatou ser no total de 2^n, multiplica-se a quantidade total de agentes menos o próprio agente que está enviando a mensagem de consulta – pois as consultas emitidas por um agente a si mesmo não são contabilizadas, uma vez que não requer o uso de nenhum recurso de rede. Assim, segue que o número de mensagens trocadas possui complexidade assintótica de

Figura 20 – Árvore de chamadas para o exemplo de cenário de pior caso com $n = 3$.



Fonte: Autoria própria.

$O(|Ags| \times 2^n)$ no pior caso.

4.2.3.2 Otimização Baseada em *Cache*

Uma solução para o problema da complexidade computacional no pior caso é uma memória *cache* que permite armazenar e reutilizar as respostas das consultas recebidas. Essa memória é individual e local em cada agente e mapeada para cada foco de consulta específico, de forma que a memória de um foco de consulta não seja afetada por outra. Duas *caches* diferentes são propostas: $C_\alpha[(p, hist_p)]$, que é mapeada por *r-literal* e histórico; e $C_\alpha[p]$, que é mapeada apenas por *r-literal*. A primeira permite aproveitar uma resposta anterior sem a necessidade de qualquer ajuste, mas não permite reutilizar a resposta quando o mesmo *r-literal* é consultado com um histórico diferente. Portanto, esta *cache*, por si só, não resolveria o problema do pior cenário apresentado. Por exemplo, se $n = 3$, conforme apresentado na Figura 20, quando a consulta para $\langle @, x_3 \rangle$ é emitida por a para c, o histórico é $[\langle a, x_1 \rangle]$, mas quando a mesma consulta é emitida por b para c, o histórico é $[\langle a, x_1 \rangle, \langle b, x_2 \rangle]$. Se apenas $C_\alpha[(p, hist_p)]$ estiver disponível, a consulta deverá ser executada novamente. Isso é resolvido por meio de $C_\alpha[p]$, que não requer que a consulta tenha o mesmo histórico para que a resposta em *cache* seja aproveitada. Porém, apenas reutilizar uma resposta desconsiderando a diferença de históricos entre as chamadas pode levar a uma resposta inválida, visto que um item no histórico pode levar à criação de argumentos falaciosos que não ocorreriam na ausência desse item. Portanto, algumas verificações devem ser realizadas, bem como ajustes na resposta em *cache* para torná-la adequada ao histórico da consulta atual, conforme apresentado, passo a passo, a seguir.

É importante notar aqui que esta solução se baseia na premissa de que os argumentos serão construídos a partir do estado da BC de cada agente no momento em que ele recebe a consulta, de modo que mudanças nas BCs ocorridas durante o processamento de um foco de consulta são desconsideradas se tais mudanças afetarem as regras a partir das quais argumentos já foram construídos ou ainda estão sendo construídos. Em outras palavras, se, no escopo de um foco de consulta α, uma consulta para um *r-literal* p foi previamente processada pelo agente a, e as regras de a que concluem p são alteradas, as próximas consultas para p, se houverem, continuarão obtendo as respostas calculadas antes das alterações. Isso evita a repetição do processamento e não gera inconsistências, pois a resposta final retornada ao agente emissor dependerá sempre do conhecimento dos agentes no dado instante em que os agentes recebem a consulta pela primeira vez no escopo daquele foco de consulta. Mais especificamente, a partir

do momento em que um agente identificou uma regra e os membros de seu corpo a fim de realizar consultas acerca deles durante o processamento da função $Find\_Def\_Args$, alterações realizadas após esse instante não mais impactarão nos argumentos gerados. Por exemplo, se a regra for excluída, mas o processamento dos membros de seu corpo já tiver sido iniciado (lembrando que isso ocorre paralelamente para cada membro do corpo), os argumentos baseados na regra continuarão sendo construídos. Se for necessário que se considere o conhecimento que foi atualizado no decorrer ou após uma consulta ter terminado, basta que o agente emissor reenvie a consulta para tirar proveito do conhecimento atualizado.

Outro pressuposto aqui é que cada foco de consulta é independente um do outro, sendo possível que eles obtenham respostas diferentes dependendo do estado dos agentes no momento em que são consultados sobre algum *r-literal*. Isso é muito importante porque cada foco de consulta lida com uma BC global estendida diferente, que representa um estado de coisas específico, uma vez que podem ter BCs de foco diferentes. Portanto, mesmo que as BCs de cada agente não mudem durante o processamento de dois focos de consulta simultâneos, eles podem ter resultados diferentes dependendo do conhecimento de foco de cada foco de consulta, conforme apresentado nos Exemplos 1.1, 1.2 e 1.3.

Um registro $C_\alpha[(p, hist_p)]$ (ou $C_\alpha[p]$) em a pode ter um valor nulo ($\perp$), um *futuro* (como explicado a seguir), ou a resposta real retornada na consulta anterior. Cada vez que um agente a recebe uma consulta para um *r-literal* p, ele primeiro verifica se essa consulta não foi recebida anteriormente no mesmo foco de consulta verificando o valor de $C_\alpha[(p, hist_p)]$, e, se não existir, de $C_\alpha[p]$. Se for o caso e a resposta já tiver sido retornada, ele prossegue em usar essa resposta. Se a resposta ainda não foi encontrada – ou seja, os agentes ainda estão no meio do processo de raciocínio para encontrá-la – a *thread* para a consulta repetida fica esperando até que uma resposta seja recebida. Isso é alcançado por meio do conceito de *futuro* (ou *promessa*) (BAKER; HEWITT, 1977), um padrão de programação assíncrona implementado por muitas linguagens de programação modernas e que está intimamente relacionado ao padrão publicador/subscritor (do Inglês, *publisher-subscriber*). Em termos simples, um futuro é um objeto que atua como *proxy* de um resultado inicialmente desconhecido porque o cálculo de seu valor ainda não foi concluído. Muitas *threads* paralelas podem esperar (ou "se inscreverem") até que o valor deste objeto seja definido. Quando isso acontece, elas são notificadas, e assim são capazes de capturar o valor e continuar sua execução.

A fim de ajustar o algoritmo para acomodar este *cache* e todo o processamento envolvido,

o código apresentado no Algoritmo 25 deve ser inserido antes da linha 3 do Algoritmo 1.

O algoritmo é dividido em 3 (três) etapas principais, sendo que a segunda possui 2 (duas) subetapas, uma das quais possui mais 2 (duas) subetapas:

1. Tenta recuperar resposta da *cache* por *r-literal* e histórico $C_\alpha[(p, hist_p)]$ (linhas 1 a 3)

2. Caso não seja possível, tenta recuperar ou adaptar resposta da *cache* apenas por *r-literal* $C_\alpha[p]$ (linhas 4 a 23):

    a) Verifica o caso em que há argumentos no histórico da *cache* $C_\alpha[p]$ cujas conclusões estão no histórico da consulta atual (linhas 6 a 8). Se houver:

        i. Substitui os argumentos em *cache* cujas conclusões estão no histórico da consulta atual por nós-folha falaciosos (linhas 9 a 12)

        ii. Recalcula a força dos conjuntos de argumentos e o valor-verdade da conclusão (linhas 13 a 18)

    b) Grava a resposta reutilizada ou adaptada em $C_\alpha[(p, hist_p)]$ e retorna a resposta (linhas 19 e 23)

3. Caso não tenha sido possível reutilizar ou adaptar a resposta em *cache*, cria novo futuro em $C_\alpha[(p, hist_p)$ – e em $C_\alpha[p]$, se este estiver vazio – e prossegue com o processamento normal da consulta (linhas 24 e 25)

A linha 1 verifica se há um valor não nulo em $C_\alpha[(p, hist_p)]$, que pode ser uma resposta futura ou formada (linha 1). Se for o caso, então a resposta é devolvida a a_0 se não for um futuro; caso contrário, entra em um estado de espera pela resolução do futuro, após o que a resposta é enviada (linhas 2 e 3). Se $C_\alpha[(p, hist_p)]$ é nulo, mas $C_\alpha[p]$ não, então a resposta é recuperada dele ou a *thread* aguarda sua resolução (linhas 4 e 5).

As linhas 6 a 18 tratam do caso em que há uma resposta em $C_\alpha[p]$, mas há itens em $hist_p$ (o histórico da consulta atual) que aparecem em $Args_p^C$, os quais serão chamados de *r-literais* repetidos, de modo que tal resposta não poderia ser aceita da forma que está em *cache* porque os *r-literais* repetidos criariam novos argumentos falaciosos. Neste caso, em vez de voltar a processar toda a consulta, o que potencialmente exigiria a consulta a vários agentes novamente, é possível adaptar a resposta em *cache* para a consulta atual. Isso é feito substituindo todos os

Algoritmo 7 – Pseudocódigo para o mecanismo de *cache* a ser inserido antes da linha 3 do procedimento $Query$.

1 **if** $C_\alpha[(p, hist_p)] \neq \bot$ **then**
2 | $Ans(p,\alpha,tv_p^C,Args_p^C, Args_{\sim p}^C) \leftarrow$ wait for resolution of $C_\alpha[(p, hist_p)]$
3 | send $Ans(p,\alpha,tv_p^C,Args_p^C, Args_{\sim p}^C)$ to a_0 and terminate
4 **if** $C_\alpha[p] \neq \bot$ **then**
5 | $Ans(p,\alpha,tv_p^C,Args_p^C, Args_{\sim p}^C) \leftarrow$ wait for resolution of $C_\alpha[p]$
6 | $repeated\_prems \leftarrow hist_p \cap \bigcup_A^{Args_p^C} Prem(A)$
7 | $repeated\_prems_{\sim p} \leftarrow hist_p \cap \bigcup_A^{Args_{\sim p}^C} Prem(A)$
8 | **if** $repeated\_prems \neq \emptyset$ *or* $repeated\_prems_{\sim p} \neq \emptyset$ **then**
9 | | **for** *each* $q \in repeated\_prems$ **do**
10 | | | $Args_p^{C\prime} \leftarrow Replace\_Args\_For\_Prems\_By\_Fall\_Nodes(Args_p^C, q)$
11 | | **for** *each* $q \in repeated\_prems_{\sim p}$ **do**
12 | | | $Args_{\sim p}^C{}' \leftarrow$ $Replace\_Args\_For\_Prems\_By\_Fall\_Nodes(Args_{\sim p}^C, q)$
13 | | $tv_p^{C\prime} = tv_p^C$
14 | | **if** $\exists A \in Args_p^{C\prime} \cup Args_{\sim p}^C{}'$ *s.t.* $Type(A) = strict$ **then**
15 | | | **for** *each pair* $Args_{p'}^{C\prime}$, $Args_{\sim p'}^C{}'$, *for each distinct* $p' \in \{Conc(A) \mid A \in Args_p^{C\prime}\}$ **do**
16 | | | | $tv_{p'} \leftarrow Compare\_Def\_Args(Args_{p'}^{C\prime}, Args_{\sim p'}^C{}')$
17 | | | | **if** $tv_{p'} = true$ **then** $tv_p^{C\prime} = true$
18 | | | | **else if** $tv_{p'} = undec$ *and* $tv_p^{C\prime} \neq true$ **then** $tv_p^{C\prime} \leftarrow undec$
19 | | $C_\alpha[(p, hist_p)] \leftarrow Ans(p,\alpha,tv_p^{C\prime},Args_p^{C\prime}, Args_{\sim p}^C{}')$
20 | | send $Ans(p,\alpha,tv_p^{C\prime},Args_p^{C\prime}, Args_{\sim p}^C{}')$ to a_0 and terminate
21 | **else**
22 | | $C_\alpha[(p, hist_p)] \leftarrow Ans(p,\alpha,tv_p^C,Args_p^C, Args_{\sim p}^C)$
23 | | send $Ans(p,\alpha,tv_p^C,Args_p^C, Args_{\sim p}^C)$ to a_0 and terminate
24 $C_\alpha[(p, hist_p)] \leftarrow$ new future to be resolved when $Ans(p,\alpha,tv_p,Args_p,Args_{\sim p})$ is finished
25 **if** $C_\alpha[p] = \bot$ **then** $C_\alpha[p] \leftarrow C_\alpha[(p, hist_p)]$

Fonte: Autoria própria.

argumentos que concluem os *r-literais* repetidos por um nó-folha falacioso por meio de uma função $Replace\_Args\_For\_Prems\_By\_Fall\_Nodes$ (linhas 9 a 12), que também recalcula a força dos argumentos correspondentes aos nós ancestrais dos argumentos substituídos. Isso é feito nos dois conjuntos de argumentos $Args_p^C$ e $Args_{\sim p}^C$, resultando na criação dos conjuntos de argumentos $Args_p^{C\prime}$ e $Args_{\sim p}^C{}'$.

Em seguida, das linhas 13 a 18, as comparações entre os argumentos atualizados com vista a descobrir seus valores-verdade é refeita, de forma similar ao que é feito na etapa 6 do procedimento $Query$. Primeiramente, verifica-se se existe algum argumento em $Args_p^{C\prime}$ e $Args_{\sim p}^C{}'$ que é estrito, pois, se for o caso, o valor-verdade deve ser mantido conforme está em

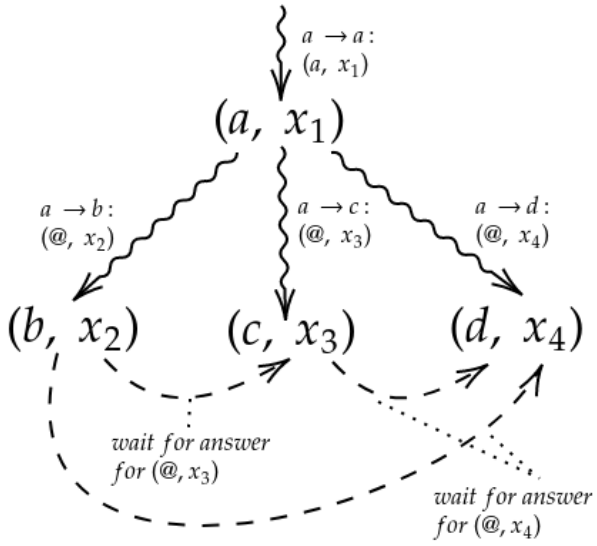
cache (tv_p^C). Isso se deve ao fato de que argumentos estritos, além de definirem definitivamente o valor-verdade, se existirem, também nunca seriam modificados por esse procedimento, visto se que pressupõe que o conjunto de regras estritas de um agente é sempre consistente, portanto, livre da possibilidade de ciclos ou de cadeias de raciocínio autodestrutivas. Finalmente, a resposta encontrada adaptada é atribuída a $C_\alpha[(p, hist_p)]$ para possíveis consultas futuras ao mesmo *r-literal* p e histórico $hist_p$, e finalmente a resposta é retornada a a_0 (linhas 19 e 20).

Caso nenhum *r-literal* repetido seja encontrado, simplesmente retorna-se a resposta em *cache*. Observe que, se a *thread* do agente entrar nas condicionais da linha 1 ou 4, ele sempre encerrará a execução enviando uma resposta para a_0, não alcançando a linha 24 e, portanto, não executando a consulta novamente. Por outro lado, se nenhuma resposta puder ser reutilizada, a linha 24 cria um novo futuro em $C_\alpha[(p, hist_p)]$ e, se não houver resultado ou futuro em $C_\alpha[p]$, o mesmo futuro será atribuído a $C_\alpha[p]$, de forma que terá a mesma resposta de $C_\alpha[(p, hist_p)]$ quando a consulta for finalizada (linha 25).

4.2.3.3 Complexidade após a Otimização

Dada a inclusão do mecanismo de *cache*, a árvore de chamadas para o exemplo de pior caso quando $n = 3$, mostrando apenas as chamadas que atingem o estágio de $Find\_Def\_Args$, é apresentada na Figura 21.

Figura 21 – Árvore de chamadas do exemplo do cenário de pior caso quando $n = 3$ com *cache*.



Fonte: Autoria própria.

Observe que apenas $n + 1 = 4$ consultas chegam ao estágio de $Find\_Def\_Args$ neste

caso, uma vez que se eliminou a necessidade de repetir o mesmo processamento de consulta para $\langle @, x_3 \rangle$ e $\langle @, x_4 \rangle$, como ocorre na versão sem *cache*. Logo, pode-se observar que o número de chamadas para $Query$ que chegam ao estágio de $Find\_Def\_Args$ é agora proporcional ao número de *ir-literais* (que neste caso são todas as premissas nos argumentos). Portanto, pode-se concluir que a complexidade assintótica da quantidade de chamadas a $Query$ que alcançam o estágio de $Find\_Def\_Args$ é $O(n)$.

Quanto ao número de mensagens, também é reduzido. No pior caso apresentado, para $n = 3$, apenas $3 \times 3 = 9$ mensagens são enviadas por a para cada um dos agentes b, c e d ($|Ags| - 1 = 3$) acerca de cada um dos *er-literais* no corpo da regra r_{a1} ($n = 3$). Em seguida, haverá $2 \times 3 = 6$ mensagens, correspondentes aos dois *er-literais* no corpo de r_{b1}, que também devem ser enviados aos 3 demais agentes. Em seguida haverá $1 \times 3 = 3$ mensagens enviadas por c, correspondentes ao único *er-literal* no corpo de r_{c1} enviadas aos 3 demais agentes. Portanto, $(3 \times (3 + 2 + 1)) \times 2 = 36$ mensagens são trocadas, incluindo as mensagens de consulta e resposta. Generalizando, o número de mensagens $m = [(|Ags| - 1) \times \sum_{i=1}^{n} i)] \times 2$ = $[(|Ags| - 1) \times (n \times (1 + n)/2)] \times 2 = (|Ags| - 1) \times (n^2 + n)$. Logo, o número de mensagens trocadas com a otimização baseada em *cache* é $O(|Ags| \times n^2)$.

Em um caso médio, no entanto, o número de mensagens tende a ser menor. Seja m a quantidade de *er-literais* e k a quantidade de *cr-literais* estrangeiros. Supondo que cada *er-literal* ocorra somente uma vez em toda a base de conhecimento estendida – diferente do pior caso, em que se repetem – então para cada *er-literal*, são enviadas $(|Ags| - 1) \times 2$ mensagens, contando-se as mensagens de consulta e resposta. Além disso, para cada *cr-literal* estrangeiro, são enviadas 2 mensagens, uma de consulta e uma de resposta. Logo, a quantidade total de mensagens enviadas é $M = m \times (|Ags| - 1) \times 2 + k \times 2$. Portanto, a complexidade assintótica nesse caso médio é $O(|Ags| \times m + k)$.

4.3 DISCUSSÃO

Este capítulo apresenta um algoritmo distribuído de resposta a consultas que realiza computacionalmente o *framework* apresentado no Capítulo 3. Em resumo, o algoritmo apresenta as seguintes características:

- Segue uma abordagem preguiçosa (do Inglês, *lazy evaluation*) de avaliação de consultas. Isto é, o processo de raciocínio em busca do valor-verdade, o que inclui a construção de

argumentos a favor e contra a conclusão do *r-literal*, é realizado no momento em que o agente recebe uma consulta. Isto permite que o algoritmo opere de forma eficiente, além de ser naturalmente tolerante a mudanças nas BCs, uma vez que basta realizar novamente a consulta para que a resposta possa ser baseada em conhecimentos atualizados.

- O algoritmo constrói estruturas baseadas em argumentação, o que permite: (1) a reutilização dos argumentos gerados pelos agentes em diferentes etapas do raciocínio, como é feito na otimização baseada em *cache*; e (2) a utilização das estruturas em outras possíveis futuras funcionalidades, tal como em aprendizagem, revisão de crenças e geração de explicações.

- O algoritmo permite a utilização de diferentes estratégias de cálculo de força de argumentos e comparação de argumentos.

- O algoritmo prevê os pontos em que pode ser executado com paralelização, o que permite sua execução eficiente em um ambiente distribuído.

- Foi demonstrado que o algoritmo termina, que é correto e completo em relação ao formalismo de argumentação proposto, e que possui complexidade polinomial com a otimização proposta.

Uma limitação do algoritmo diz respeito à possibilidade de haver instabilidades nas comunicações entre agentes, o que faria com que nem todos os argumentos possíveis pudessem ser gerados, consequentemente afetando a resposta da consulta. Por exemplo, supondo quatro agentes a, b e c e d, tal que a tem uma regra $r_{a1} : \langle a,x_1\rangle \Leftarrow \langle b,x_2\rangle, \langle c,x_3\rangle$, b tem uma regra $r_{b1} : \langle b,x_2\rangle \Leftarrow \langle d,x_4\rangle$, c tem uma regra $r_{c1} : \langle c,x_3\rangle \Leftarrow \langle d,x_4\rangle$, e por fim d tem uma regra $r_{d1} : \langle d,x_4\rangle \Leftarrow$. Tanto b quanto c devem enviar uma mensagem a d acerca de $\langle d,x_4\rangle$. Supondo que b receba uma resposta de d, mas c, por algum motivo, tal como instabilidade de comunicação, não recebe uma resposta, então o subargumento com conclusão $\langle c,x_3\rangle$ não será criado, pois o argumento com conclusão $\langle d,x_4\rangle$, do ponto de vista local de c, não pôde ser criado, mesmo que o agente b já o tenha recebido de d.

Isto é, mesmo que, do ponto de vista global, todas as informações necessárias para se concluir $\langle a, x_1\rangle$ possam ser derivadas, não se consegue atingir a conclusão. Isto se deve ao fato de o raciocínio ser feito localmente por cada agente que está tentando provar um ramo da árvore que representa a cadeia de raciocínio, e esses agentes – em diferentes ramos – não se comunicam

entre si, mas apenas com o agente que lhe submeteu a consulta e com os agentes para os quais eles porventura submetam consultas. No exemplo dado, b já possui o conhecimento necessário para que c possa construir seu argumento, mas b e c não se comunicam, sendo, portanto, uma limitação imposta pelo modelo de raciocínio distribuído e descentralizado.

5 TRABALHOS RELACIONADOS

Este capítulo apresenta em resumo os principais trabalhos relacionados, enfatizando as diferenças com a abordagem proposta nesta tese. Apenas trabalhos que propõem formas de raciocínio distribuído baseados em argumentação foram incluídos. Os trabalhos são avaliados principalmente no que diz respeito à forma como resolvem conflitos entre conclusões conflitantes, além de avaliar se realizam raciocínio totalmente distribuído a partir de BCs inter-relacionadas, se suportam a possibilidade de referenciar conhecimentos provenientes de fontes arbitrárias – o que permitiria a aplicação em ambientes abertos e dinâmicos – e se possuem mecanismos de compartilhamento de conhecimento (foco) em consultas.

A Seção 5.1 discute os sistemas de inferência par-a-par. A Seção 5.2 discute duas abordagens de argumentação distribuída baseadas em DeLP e programação lógica. A Seção 5.3 discute o sistema de argumentação contextual proposto por Brewka e Eiter (2009). A Seção 5.4 discute a Lógica Derrotável Contextual de Bikakis e Antoniou (2010). A Seção 5.5 discute os sistemas multicontexto distribuídos e dinâmicos de Dao-Tran *et al.* (2010). Finalmente, a Seção 5.6 apresenta uma tabela comparativa entre esses trabalhos e o presente trabalho.

5.1 SISTEMAS DE INFERÊNCIA PAR-A-PAR (P2P)

Os sistemas de inferência par-a-par (P2P, do Inglês, *peer-to-peer*) (CHATALIC *et al.*, 2006; BINAS; MCILRAITH, 2008) podem ser vistos como casos especiais de SMCs, pois consistem em entidades autônomas (*pares*) baseadas em lógica, similares aos agentes neste trabalho, que trocam informações locais. No entanto, em vez de usarem regras de ponte/mapeamento, é utilizado o conceito de arestas rotuladas com proposições em um grafo que inter-relaciona os pares no sistema. Se uma aresta entre os pares a e b contém uma dada proposição p, então a e b compartilham dessa mesma proposição, o que permite uma forma de fluxo de informações entre os pares.

O sistema de inferência proposicional par-a-par proposto por Chatalic *et al.* (2006) permite lidar com conflitos causados por fontes de informações mutuamente inconsistentes, detectando-os e eliminando-os, de modo que o raciocínio é realizado sem as informações que causam os conflitos. Isso é diferente da forma de raciocínio proposta na presente tese, que consiste em realizar o raciocínio sem eliminar as inconsistências. Além disso, Chatalic *et al.*

(2006) não incluem a ideia de preferências entre pares do sistema, que poderiam ser usadas para resolver conflitos causados por fontes de informações mutuamente inconsistentes. Binas e McIlraith (2008) propõem uma abordagem baseada em argumentação e um algoritmo de resposta a consultas que resolve inconsistências por meio de uma relação de preferência entre os pares derivada de um valor de prioridade atribuído a cada par. No entanto, o método seguido por Binas e McIlraith (2008) pressupõe uma relação de preferência global sobre os pares, que é compartilhada e usada por todos. Essa característica não é bem adequada à dimensão de perspectiva do raciocínio contextual, que deveria permitir que cada agente utilizasse sua própria relação de preferência baseada em seu próprio ponto de vista. A abordagem também não prevê a aquisição de conhecimentos de fontes arbitrárias, e não propõe formas de compartilhamento de conhecimento de foco em consultas.

5.2 ARGUMENTAÇÃO DISTRIBUÍDA BASEADA EM DELP E PROGRAMAÇÃO LÓGICA

Alguns trabalhos baseados em DeLP propõem algumas formas de argumentação distribuída em SMAs. Thimm e Kern-Isberner (2008) apresentam um SMA baseado em argumentação (ArgMAS, do Inglês, *argumentation-based multi-agent system*), no qual vários agentes propõem argumentos a favor ou contra uma dada sentença lógica. No entanto, o raciocínio em ArgMAS é feito de forma centralizada, e não totalmente distribuída, dependendo de um agente moderador que tem parte da responsabilidade de coordenar o processo de argumentação e analisar as estruturas de argumentação geradas pelos demais agentes.

Em (THIMM *et al.*, 2008) é apresentada uma extensão desse *framework* para apoiar a colaborações entre agentes, o que permite combinar as bases de conhecimento de diferentes agentes a fim de produzir argumentos. Portanto, essa abordagem apresenta uma forma de raciocínio com bases de conhecimentos inter-relacionadas, embora não utilize o conceito de regras de ponte/mapeamento. Em vez disso, os agentes em uma aliança de agentes constroem argumentos parciais que contêm conjuntos de literais livres, isto é, conjuntos de literais que ainda não são apoiados por outros argumentos. A partir disso, um meta-agente executa um algoritmo de encadeamento para trás tentando construir um argumento completo com base nos argumentos parciais.

No entanto, essa proposta apresenta algumas diferenças relevantes em comparação ao DDRMAS: (i) A colaboração entre agentes é restrita a um conjunto predefinido de agentes per-

tencentes a uma mesma aliança, enquanto no DDRMAS um agente pode colaborar com qualquer outro agente, sem a necessidade de ser criada uma aliança; (ii) um agente só pode participar de uma aliança por vez, enquanto nosso trabalho permite que um agente participe do raciocínio de múltiplos focos de consulta simultaneamente; (iii) os argumentos completos são gerados por um meta-agente, que centraliza boa parte do raciocínio, e não pelos agentes individuais de forma totalmente distribuída, como proposto neste trabalho; e (iv) a correspondência entre os literais de diferentes agentes é definida apenas pela correspondência exata, isto é, um argumento parcial com um literal livre x é encadeado a um literal que conclui exatamente o valor x, não sendo feita uma correspondência baseada em similaridade. Além disso, o trabalho não apresenta formas de compartilhamento de foco em consultas.

Um trabalho que se assemelha ao ArgMAS, no sentido de gerar argumentos colaborativamente, é apresentado por Sá e Alcântara (2012). Eles propõem que diferentes agentes construam argumentos condicionais, similares aos argumentos parciais do ArgMAS, derivados a partir do conhecimento dos agentes usando a linguagem dos programas lógicos disjuntivos estendidos (EDP, do Inglês, *Extended Disjunctive Programs*) sem cabeças disjuntivas. Os agentes, então, por meio de um processo dialético, tentam produzir argumentos não-condicionais (completos), adicionando-se apoio (subargumentos) às suas condições (similares aos literais livres), o que é chamado de reescrita de argumentos. Esse processo gera uma sequência de reescritas de argumentos, cada uma realizada por um agente, e cada uma resultando em um novo argumento. Se a sequência de reescritas terminar com um argumento não-condicional, então o argumento é elegível para ser considerado aceitável no *framework*. A principal diferença, em relação ao ArgMAS, portanto, é que a construção dos argumentos completos é feita por meio de um diálogo entre os agentes, e não centralizada em um meta-agente. No entanto, assim como no ArgMAS, essa abordagem não considera a similaridade entre literais heterogêneos e não propõe o compartilhamento de foco em consultas. Também não se propõe formas de resolução de conflitos explícitas, tais como as baseadas em confiança ou preferências.

Outro trabalho, baseado em DeLP, é o Planejamento Multiagente Sensível ao Contexto (CAMAP, do Inglês, *Context-Aware Multi-Agent Planning*) apresentado por Ferrando e Onaindia (2013), no qual argumentos são construídos com base nas informações de contexto conhecidas pelos agentes para apoiar a execução de planos de ação. Um protocolo de planejamento multiagente inclui um estágio de argumentação, nas quais um agente inicial, que inicia o raciocínio, e que também gerencia o diálogo argumentativo, envia seus argumentos iniciais em uma estrutura

chamada árvore PAD (do Inglês, *Plan Argument Dialogue*) para todos os demais agentes, na forma *broadcast*, e os agente receptores então constroem argumentos que atacam os argumentos existentes, devolvendo-os ao agente inicial. O agente inicial então anexa os argumentos recebidos na árvore PAD, e realiza uma nova rodada de envio dos argumentos. Novamente, com base nos argumentos já construídos, cada agente pode apresentar novos argumentos que atacam os atuais, devolvendo-os ao agente inicial. O processo é repetido pelo agente inicial até que nenhum novo argumento seja apresentado pelos demais agentes. O agente inicial, então, invoca o *procedimento de garantia* do DeLP para verificar quais argumentos são derrotados ou não.

Embora a abordagem seja interessante por não depender de um agente moderador fixo, ainda assim, é em parte centralizada, visto que existe um agente que coordena o processo de raciocínio e realiza o procedimento de verificação da aceitabilidade dos argumentos. Este agente é o mesmo que inicia o processo de raciocínio, sendo o agente que estabelece um objetivo a ser alcançado, o que pode ser comparado, de certa forma, a um agente que recebe uma consulta, no presente trabalho. No entanto, diferente do DDRMAS, no qual os agentes realizam consultas recursivas entre si a fim de buscar apoio para a justificação de um literal, o CAMAP centraliza a busca no agente inicial, que realiza repetidos envios do conjunto de argumentos construídos por todos os agentes por *broadcast*. Além disso, no CAMAP, os argumentos não são construídos colaborativamente, com base em dependências existentes nas bases de conhecimento inter-relacionadas de diferentes agentes. Logo, vários dos argumentos para o cenário apresentado na Seção 1.1.1 do Capítulo 1 não poderiam nem mesmo ser gerados, por necessitarem de informações provenientes de outros agentes para apoiá-los. Portanto, o CAMAP não se aplica ao problema proposto neste trabalho.

5.3 SISTEMA DE ARGUMENTAÇÃO CONTEXTUAL (ACS)

O sistema de argumentação contextual (ACS, do Inglês *Argumentation Context System*) de Brewka e Eiter (2009) propõe uma abordagem modular para argumentação abstrata, com *frameworks* (grafos) de argumentos divididos em diferentes módulos que podem definir relações entre si por meio de mediadores, tal que um mediador consiste em regras de ponte que associam argumentos de um *framework* a um contexto para outro *framework*. Um contexto para um *framework* consiste em um conjunto de expressões que determinam certas propriedades do *framework*, tais como preferências, ataques, argumentos aceitáveis e até mesmo modos de raciocínio e semântica de aceitabilidade. Portanto, por tratar de *frameworks* de argumentação

abstratos, a abordagem não fornece definições específicas para a construção de argumentos a partir de uma base de conhecimento. Além disso, o ACS tem por proposta modelar um tipo de meta-argumentação – que generaliza o conceito de argumentação hierárquica proposta por Modgil (2006), por meio da qual propriedades usadas em um *framework* de argumentação podem ser herdadas por um outro *framework* em um nível hierárquico diferente – sendo portanto uma forma de definir a interoperabilidade entre diferentes *frameworks* de argumentação, o que é diferente da proposta deste trabalho. Os autores também não propõem soluções para o problema de se referenciar conhecimentos de fontes arbitrárias, nem mecanismos de compartilhamento de conhecimento em consultas.

5.4 LÓGICA DERROTÁVEL CONTEXTUAL (CDL)

O trabalho que mais se aproxima do apresentado nesta tese, e no qual foi inspirado, é a *Lógica Derrotável Contextual* (CDL, do inglês *Contextual Defeasible Logic*) (BIKAKIS; ANTONIOU, 2010; BIKAKIS *et al.*, 2011; BIKAKIS; ANTONIOU, 2011), um *framework* de raciocínio derrotável e argumentativo totalmente distribuído baseado nos conceitos de SMC, com foco na aplicação em sistemas de Inteligência Ambiente. A CDL também é baseada na DL de Governatori *et al.* (2004), apresentada na Seção 2.2.2.1 do Capítulo 2. Um sistema CDL consiste em um conjunto de BCs distribuídas que podem ser inconsistentes entre si. A resolução dessas inconsistências, assim como o raciocínio sobre o conhecimento distribuído, é implementada por meio de *regras de mapeamento derrotáveis*, que permitem em seu corpo literais definidos por outras BCs, de modo que, para que a regra seja aplicável, o conhecimento deve ser adquirido de outras BCs. Para resolver os conflitos entre cadeias de raciocínio geradas a partir de regras contraditórias, um *framework* de argumentação baseado em DL é proposto, assim como um algoritmo distribuído de resposta a consultas.

Supondo que as BCs distribuídas de um sistema CDL sejam referentes às BCs de agentes em um SMA, a CDL fornece vários recursos necessários para a solução do problema proposto neste trabalho (ver Seção 1.2 do Capítulo 1), a saber: (i) lida com conhecimento parcial de bases de conhecimento distribuídas, por meio das regras de mapeamento; (ii) lida com conflitos entre conclusões derivadas das bases de conhecimento distribuídas, por meio da semântica de argumentação baseada em Lógica Derrotável; e (iii) lida com a questão dos diferentes graus de confiança entre agentes. No entanto, ele ainda não é suficiente, pois: (i) não suporta ambientes dinâmicos, visto que os mapeamentos definidos nas regras de mapeamento

devem ser definidos *a priori* na forma de antecedentes de regras que são literais estrangeiros; (ii) não lida com o conhecimento incerto baseado no grau de similaridade que pode haver entre bases de conhecimento cujos vocabulários são heterogêneos; e (iii) não lida com a questão de permitir consultas em que o foco do agente é compartilhado com os demais agentes. Isto se torna claro ao ser constatado que não é possível instanciar o cenário apresentado na Seção 1.1.1 do Capítulo 1 em CDL.

Quanto a partes específicas da solução, Bikakis e Antoniou propõem uma relação de preferência como uma relação de ordem total estrita sobre os agentes, o que não permite que se possua confiança equivalente em mais de um agente. Isso é resolvido em uma extensão apresentada em (BIKAKIS; ANTONIOU, 2011), que propõe uma relação de ordem parcial ampla por meio de uma estrutura de grafo direcionado acíclico, de modo que cada agente mantém internamente um grafo cujos vértices são referências a outros agentes no sistema, e as arestas indicam a preferência de um agente sobre outro. O presente trabalho adota uma abordagem diferente, que define indiretamente uma relação total ampla entre os agentes por meio de funções de confiança que mapeiam os agentes a valores contínuos em uma faixa de valores entre 0 e 1. Um grau de confiança padrão (e.g., com valor 0 ou $0{,}5$) é atribuído a cada agente para o qual não haja um grau de confiança específico ainda estabelecido. A relação é total pois todos os agentes são comparáveis, visto que tal relação de ordem provém da comparação entre valores numéricos que sempre existem (os valores provenientes da aplicação da função de confiança de um agente sobre cada agente).

Ainda referente à ordem de preferência proposta por Bikakis e Antoniou, esta é utilizada para resolver os conflitos entre argumentos, similarmente ao apresentado neste trabalho. No entanto, uma vez que não trabalham com um valor de confiança numérico, não é proposto um cálculo de força similar a uma média das forças dos subargumentos gerados pelos diferentes agentes que participaram na construção do argumento. Em vez disso, é usada a ideia de classificação (*rank*) de literais e de argumentos. Um literal com *rank* menor é definido por um agente mais confiável, e um com *rank* maior é definido por um agente menos confiável. O *rank* de um argumento é então definido como o maior *rank* dentre os *ranks* de todos os seus literais estrangeiros, isto é, como o *rank* que representa o literal estrangeiro menos preferido. A relação de derrota é definida então em termos do argumento com *rank* menor ou igual a outro argumento. Logo, a "força" do argumento sempre é proporcional à confiança que o agente tem no agente que ele menos confia, similar ao cálculo de força cético apresentado na Seção 3.3.4.

Bikakis *et al.* (2011) também apresentam um algoritmo distribuído chamado $P2P\_DR$. No entanto, não há preocupação em explicitar aspectos de paralelização. Além disso, diferentemente do algoritmo $Query$ proposto neste trabalho, que de fato constrói as estruturas dos argumentos, o $P2P\_DR$ trabalha com uma representação simplificada dos argumentos chamada de *conjunto de apoio*, que seria o equivalente ao conjunto de *ir-literais* em um argumento no algoritmo proposto nesta tese. Trabalhar com conjuntos de apoio é de fato suficiente em muitos cenários, incluindo o cenário dos coletores de cogumelos, se for utilizada uma estratégia de cálculo de força que não leva em consideração a profundidade e as diferentes ramificações de um argumento, como é o caso do cálculo de força ingênuo. No entanto, isso limita a possibilidade de se tirar proveito de diferentes métodos de cálculo de força que podem utilizar a estrutura de árvore dos argumentos, como é o caso do cálculo de força padrão proposto neste trabalho, que diminui o valor de força quando há argumentos gerados com base na confiança indireta entre agentes. Além disso, a construção dos argumentos pelo algoritmo distribuído, como é o caso do algoritmo $Query$ do DDRMAS, permite sua reutilização como insumo para possíveis mecanismos de revisão de crenças e geração de explicações que poderiam ser propostos em trabalhos futuros.

Além disso, o algoritmo $Query$ proposto nesta tese permite que os agentes raciocinem considerando múltiplos argumentos que concluem o mesmo *r-literal*. Dessa forma, os agentes podem aplicar uma gama ainda maior de estratégias possíveis, como é o caso de se considerar a soma das forças de vários argumentos, com a intuição de que, quanto mais razões existirem para se chegar a uma conclusão, mais forte será essa conclusão, como apresentado na Seção 3.3.5.1 do Capítulo 3. Isso não é possível no $P2P\_DR$, que considera somente os argumentos mais fortes a favor e contra um dado literal.

Em (BIKAKIS *et al.*, 2011), os autores propõem uma otimização baseada em *cache* para reduzir a complexidade computacional e o número de mensagens trocadas, mas eles propõem apenas uma *cache* mapeada por literal e histórico (equivalente ao $C_\alpha[(p, hist_p)]$ proposto neste trabalho), de modo que, se um agente for consultado por um literal, mas seu histórico for diferente do histórico da resposta em *cache*, ele terá que executar novamente a consulta. No entanto, eles propõem que, se é conhecido que um sistema não possui cadeias de regras circulares ou autodestrutivas na BC global, então a *cache* sempre poderia ser utilizada, independente do histórico. Contudo, isso só é possível se for conhecido previamente o fato de não haver esse tipo de cadeia de regras. Pressupondo um ambiente dinâmico, isso poderia ser um problema,

pois teria que haver algum mecanismo que verificasse a todo momento se tais cadeias foram introduzidas no escopo global do sistema, ou até mesmo algum mecanismo que evitasse que fossem introduzidas. A abordagem apresentada na presente tese, por outro lado, propõe a ideia de reaproveitar dinamicamente respostas em *cache* por meio da verificação e adaptação dessas respostas, o que funciona, até mesmo, para situações em que há tais cadeias de regras na BC global do sistema, a partir das quais são gerados argumentos falaciosos, como explicado na Seção 4.2.3.2.

5.5 SISTEMAS MULTICONTEXTO DISTRIBUÍDOS E DINÂMICOS (DDMCS)

Outras abordagens baseadas em SMC não propõem formas explícitas de resolução de conflitos (DAO-TRAN *et al.*, 2010) ou resolvem conflitos por meio do conceito de *reparo* (BREWKA *et al.*, 2018), que modifica as bases de conhecimento para manter um estado global livre de conflitos, o que pode ser indesejável em sistemas nos quais os agentes podem ter diferentes pontos de vista do ambiente que podem ser conflitantes entre si, mas que podem ser válidos em determinados contextos. No entanto, o trabalho de Dao-Tran *et al.* (2010) apresenta uma abordagem semelhante à que foi adotada para uma parte específica da solução proposta neste trabalho, denominado Sistema Multicontexto Distribuído e Dinâmico (DDMCS, do Inglês *Dynamic Distributed Multi-Context System*), que propõe o uso de regras de ponte esquemáticas com uma função de similaridade, e um algoritmo de *backtracking* para enumerar todas as substituições possíveis (instanciações) de átomos de ponte esquemática (semelhantes aos *er-literais* do presente trabalho) em um ambiente distribuído, a fim de instanciar um SMC comum, a partir do qual equilíbrios – possíveis conjuntos de crenças admissíveis – podem ser calculados.

Portanto, a proposta deles requer a pré-instanciação de um SMC comum para cada nova mudança no sistema, o que pode envolver cada base de conhecimento, visto haver dependências entre elas. Isso poderia resultar em uma sobrecarga desnecessária em ambientes altamente dinâmicos. Além disso, algum tipo de entidade centralizada seria necessária para instanciar e atribuir essas "bases de conhecimento instanciadas" para cada agente. A abordagem do presente trabalho, por outro lado, não requer nenhuma ação centralizada em resposta às mudanças. Quando um agente deve consultar outros agentes acerca de um *er-literal*, ele simplesmente envia uma mensagem em *broadcast* aos demais agentes, sem nem mesmo necessitar conhecer cada agente existente no sistema especificamente. Alterações no conhecimento dos agentes também não requerem qualquer intervenção. Quando um agente com conhecimento alterado é consultado, ele

simplesmente usará seu conhecimento atual, exceto no escopo de um foco de consulta específico, quando os argumentos resultantes de consultas anteriores a um agente são armazenados em *cache* e reutilizados durante o processamento daquele foco de consulta, conforme discutido na Seção 4.2.3.2.

Dao-Tran *et al.* (2010) também não propõem formas explícitas de resolver conflitos, o que, neste caso, seria o equivalente a realizar uma escolha entre os diversos equilíbrios possíveis. Também não se utiliza o conceito de argumentação, e não se propõe o compartilhamento de conhecimento de foco em consultas.

5.6 COMPARAÇÃO

O Quadro 2 apresenta uma comparação qualitativa entre essas abordagens com base em algumas características significativas com relação a este trabalho. Tais características são listadas a seguir:

Quadro 2 – Comparação entre trabalhos relacionados.

	Inferência P2P	**ArgMAS**	**Diálogos Cooperativos**	**CAMAP**	**ACS**	**CDL**	**DDMCS**	**DDRMAS**
a)	Sim	Sim	Não	Sim	Sim	Sim	Não	Sim
b)	Sim	Sim	Sim	Sim	Sim	Sim	Não	Sim
c)	Lógica Proposicional	DeLP	EDP	DeLP	Não possui	Lógica Derrotável	BCs heterogêneas	Lógica Derrotável
d)	Sim	Não	Sim	Não	Sim	Sim	Sim	Sim
e)	Sim	Sim	Sim*	Não	Não	Sim	Sim (DAO-TRAN *et al.*, 2015)	Sim
f)	Sim*	Sim*	Sim*	Não	Sim	Sim	Sim	Sim
g)	Não	Sim*	Sim*	Não	Não	Não	Sim	Sim
h)	Não	Não	Não	Não	Não	Não	Sim	Sim
i)	Não	Não	Não	Não	Não	Não	Não	Sim
j)	Não	Sim	Sim	Sim	Não	Não	Não	Sim

Fonte: Autoria própria.

a) *Resolução de conflitos explícita*: Indica se a abordagem formaliza maneiras com que argumentos, ou cadeias de raciocínio, com conclusões conflitantes (contraditórias), podem ser tratadas em abordagens não-monotônicas de raciocínio, no sentido de que apenas uma (ou nenhuma, nos casos em que não é possível resolver o conflito) das conclusões possa ser aceita. Todos os trabalhos que fornecem essa característica usam alguma forma de

relação de ordem, também chamada de relação de preferência, prioridade ou superioridade, sobre as regras, os agentes e/ou os argumentos. Essa característica é relevante pois este trabalho apresenta um formalismo não-monotônico que realiza a resolução de conflitos por meio de um cálculo de força de argumentos.

b) *Baseada em argumentação*: Indica se a abordagem apresenta um formalismo de raciocínio baseado em argumentação, o que permite resolver conflitos por meio de semânticas de aceitabilidade.

c) *Linguagem*: Indica a linguagem utilizada para a formalização do conhecimento nas BCs.

d) *Raciocínio totalmente distribuído*: Indica se o raciocínio é feito de forma totalmente distribuída, sem a dependência de alguma entidade que centraliza o raciocínio ou parte dele.

e) *Algoritmo de resposta a consultas*: Indica se a abordagem apresenta um algoritmo para resposta a consultas.

f) *Regras de ponte/mapeamento (BCs inter-relacionadas)*: Indica se a abordagem usa regras de ponte/mapeamento, ou algum tipo de recurso que inter-relaciona diferentes BCs.

g) *Regras esquemáticas (aquisição de fontes arbitrárias)*: Indica se a abordagem propõe algum tipo de regra de ponte/mapeamento que permita referenciar informações provenientes de fontes arbitrárias.

h) *Considera informações heterogêneas similares*: Indica se a abordagem utiliza informações de diferentes agentes com base na similaridade dessas informações, mesmo quando são heterogêneas.

i) *Compartilhamento de foco*: Indica se a abordagem propõe algum mecanismo que permite aos agentes compartilharem conhecimento de foco ao realizarem consultas a outros agentes.

j) *Estrutura de argumentação autoexplicativa operacional*: Indica se a solução computacional (algoritmo) da abordagem realiza a construção de argumentos (e contra-argumentos) estruturados cuja estrutura expressa com precisão a cadeia de raciocínio utilizada para o apoio à conclusão ou à negação da conclusão. Essa característica é relevante pois tal estrutura pode ser utilizada como insumo para possíveis outras funcionalidades, tais como aprendizagem, revisão de crenças e geração de explicações.

É interessante notar que os critérios de (g) a (i) estão relacionados às principais contribuições deste trabalho: a consideração de um ambiente aberto e dinâmico no qual os agentes possuem conhecimento imperfeito e estão em processo de descoberta desse ambiente.

Com exceção dos diálogos cooperativos de Sá e Alcântara (2012) e do DDMCS, todos apresentam resolução explícita de conflitos por meio de algum tipo de relação de preferência ou superioridade que possibilite a escolha entre diferentes respostas e argumentos conflitantes. O sistema de inferência P2P de Binas e McIlraith (2008) utiliza um valor inteiro positivo de prioridade atribuído a cada par, significando que um valor *menor* de prioridade implica que a prioridade é *melhor*. A partir disso, é calculada a classificação (*rank*) de um argumento como sendo a prioridade máxima dentre todos os pares utilizados para construir o argumento, adotando a ideia de que a classificação de um argumento é relativa à prioridade do pior par que contribuiu para com a construção do argumento. Os autores afirmam que as prioridades definem uma ordem de preferência total ou parcial ampla sobre os pares, embora não demonstrem como a ausência de prioridade é tratada, o que permitiria a definição de ordem parcial. A prioridade de cada par também é considerada globalmente, e não do ponto de vista de cada agente, como já foi mencionado.

Uma abordagem similar é adotada pela CDL, que, em vez de valores de prioridade, utiliza uma ordem de preferência total estrita entre BCs distribuídas. Cada BC possui sua própria ordem de preferência sobre as demais BCs no sistema, a partir da qual a classificação dos argumentos gerados é calculada como a classificação máxima de todos os literais estrangeiros utilizados no argumento, adotando assim a mesma intuição de Binas e McIlraith (2008) de que a classificação de um argumento deve ser proporcional à BC menos preferida que participou da geração do argumento.

O ACS define abstratamente preferências entre argumentos e valores de argumentos. Similarmente, O ArgMAS e o CAMAP, por serem baseados em DeLP, também definem preferências de forma abstrata.

Enfim, o DDRMAS utiliza um cálculo de força de argumentos, o qual acaba por gerar uma relação de ordem total ampla entre os argumentos, baseado numa função de confiança que um agente tem em relação aos demais e no grau de similaridade das informações adquiridas de outros agentes por meio das regras de mapeamento.

Apenas o DDMCS não é baseado em argumentação, e apenas o ACS é um *framework* de argumentação abstrata, logo não definindo a construção e estrutura dos argumentos com base

em uma BC.

Quanto à linguagem: o sistema P2P de Binas e McIlraith (2008) é baseado em lógica proposicional, o ArgMAS e o CAMAP são ambos baseados em DeLP, a abordagem de diálogos cooperativos de Sá e Alcântara (2012) é baseada em EDP, o ACS não define linguagem por ser um *framework* abstrato, a CDL e o DDRMAS são baseadas na DL, e o DDMCS suporta BCs distribuídas com lógicas heterogêneas.

Todos os trabalhos, exceto o ArgMAS e o CAMAP, apresentam raciocínio totalmente distribuído. Estes dois são distribuídos apenas em parte, no sentido de que diferentes agentes geram argumentos baseados em seu conhecimento local. No entanto, boa parte do raciocínio é centralizado: no caso do ArgMAS, em um agente moderador, e no CAMAP, no agente que inicia o processo de raciocínio.

Apenas o ACS e o CAMAP não apresentam algoritmo de resposta a consultas. O CAMAP, especificamente, modela a argumentação de agentes a favor e contra planos de ação, não sendo especificamente voltado para a resposta a consultas. A abordagem de diálogos cooperativos realiza o processo de argumentação a partir do estabelecimento de um objetivo, o que pode ser visto como uma forma de consulta.

Quanto a regras de ponte/mapeamento, o que caracteriza BCs inter-relacionadas, apenas o CAMAP não apresenta nenhum formalismo nesse sentido. Embora a inferência P2P de Binas e McIlraith (2008) não defina regras de ponte/mapeamento, ela define o conceito de arestas rotuladas em um grafo de relacionamentos entre pares. Quanto ao ArgMAS e ao diálogo cooperativo, embora não definam regras de mapeamento, propõem a construção de argumentos de forma colaborativa por meio da geração de argumentos parciais a partir das BCs de diferentes agentes, sendo, portanto, uma abordagem que considera BCs inter-relacionadas. No caso, esse inter-relacionamento é definido pelo fato de haver literais idênticos em diferentes BCs.

Apenas o DDMCS e o DDRMAS propõem, tanto o conceito de regras de ponte/mapeamento esquemáticas, quanto a ideia de vinculação de conhecimentos de diferentes BCs com base na similaridade entre as informações. No entanto, a forma como o DDMCS operacionaliza a vinculação entre as BCs com base em regras esquemáticas é diferente da forma como o DDRMAS o faz, sendo que o primeiro realiza uma pré-configuração do sistema a cada mudança ocorrida, enquanto o segundo realiza uma vinculação *online*, no momento da consulta. Embora o ArgMAS e a abordagem de diálogos cooperativos não proponham um tipo de regra esquemática, eles permitem a colaboração entre agentes sem que um agente precise saber *a*

priori quais literais os outros possuem em suas BCs. No entanto, no caso do ArgMAS, essa construção colaborativa não acontece exatamente entre agentes totalmente arbitrários, visto que só ocorre no escopo de uma aliança de agentes pré-determinada. Além disso, nem o ArgMAS nem os diálogos cooperativos propõem a vinculação entre literais heterogêneos pela similaridade entre eles.

Nenhum dos trabalhos, além do DDRMAS, propõem a ideia de compartilhamento de conhecimento de foco nas consultas. Uma abordagem que se aproxima da ideia é a de *consultas contextuais* a *DeLP-Servers*, conforme apresentado na Seção 2.6. No entanto, tal abordagem não foi incluída na comparação por não se tratar de uma forma de raciocínio distribuído. Isto fica claro ao se observar que o DDRMAS realiza uma propagação do conhecimento de foco à medida que consultas subsequentes a uma consulta inicial são realizadas em um mesmo foco de consulta, enquanto que as consultas contextualizadas a *DeLP-Servers* não realizam tal propagação, justamente pelo fato do raciocínio ser executado em uma única entidade.

Por fim, apenas o ArgMAS, o CAMAP, a abordagem de diálogos cooperativos e o DDRMAS realizam uma construção das estruturas de argumentação operacionalmente.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresenta uma abordagem totalmente distribuída de raciocínio baseada em argumentação. As entidades envolvidas são modeladas como agentes baseados em lógica, cada um possuindo uma base de conhecimento local, e as associações entre o conhecimento dos diferentes agentes como mapeamentos entre suas respectivas bases de conhecimento. As bases de conhecimento são representadas como teorias locais de Lógica Derrotável (DL) (GOVERNATORI *et al.*, 2004) estendidas por meio de regras de mapeamento derrotáveis, e com uma função de confiança individual do agente em relação a outros agentes. Uma vez que são considerados ambientes abertos e dinâmicos, os mapeamentos podem incluir a necessidade de aquisição de informações de fontes arbitrárias, não conhecidas *a priori*, sendo assim necessário que o raciocínio realize a vinculação entre informações de diferentes agentes em tempo de execução com base na similaridade entre as informações. Finalmente, ao enviar consultas a outros agentes acerca de um determinado literal, o agente pode compartilhar, junto à consulta, um conjunto de regras que especificam o foco, isto é, informações detidas pelo agente que podem ser relevantes de se compartilhar com os outros agentes para que estes sejam capazes de colaborar para com a busca de uma resposta à consulta.

O trabalho apresenta algumas contribuições específicas na área de argumentação, a saber: (1) a definição de uma estrutura de argumentação distribuída baseada no conhecimento de diferentes agentes e construída de forma colaborativa; e (2) uma semântica de argumentação baseada em uma força de argumentos derivada dos graus de confiança entre diferentes agentes e do grau de similaridade entre informações heterogêneas de diferentes agentes, assim como métodos alternativos de cálculo de força e de comparação de argumentos.

Também é apresentado um algoritmo distribuído eficiente que realiza computacionalmente o formalismo proposto, permitindo sua fácil implementação em linguagens de programação e tecnologias diversas. O algoritmo prevê os pontos em que deve ser executado com paralelismo, o que é importante em um contexto de SMAs que executam de forma distribuída e se comunicam por meio de trocas de mensagens. Diferente de outros trabalhos encontrados na literatura, o algoritmo apresentado também possui a vantagem de realmente gerar as estruturas de argumentação propostas, que podem ser persistidas e posteriormente utilizadas por outros mecanismos que podem delas tirar proveito, tais como mecanismos de aprendizagem, revisão de crenças e geração de explicações, o que também abre possibilidades para trabalhos futuros.

Dentre os possíveis trabalhos futuros vislumbrados, inclui-se realizar implementações a fim de testar na prática a proposta deste trabalho e possibilitar o desenvolvimento de aplicações. Uma vez que o algoritmo é apresentado como um pseudocódigo independente de tecnologias ou linguagens, é possível implementá-lo em qualquer linguagem de programação, sendo necessário que a tecnologia forneça um gerenciamento adequado de paralelismo. Especificamente, pode ser interessante a implementação do modelo utilizando o paradigma de programação orientado a agentes, seguindo a mesma linha dos trabalhos apresentados por Berariu (2014) e Panisson e Bordini (2016), que propõem implementações de raciocínio derrotável em agentes utilizando a plataforma Jason (BORDINI *et al.*, 2007). Tal plataforma provê uma implementação de agentes baseado no modelo BDI, incluindo o suporte nativo à representação de conhecimento por meio de regras de inferência. Além disso, as preocupações relacionadas à distribuição dos agentes são tratadas nativamente, incluindo a geração dos argumentos no *framework* JADE (BELLIFEMINE *et al.*, 1999), que já inclui diversas funcionalidades para SMAs, tais como protocolos de troca de mensagens, plataforma de agentes, serviço de páginas brancas e amarelas, etc.

Outra ideia, que foi discutida na Seção 2.6, é uma possível integração da abordagem proposta neste trabalho com o *framework* CArtAgO (RICCI *et al.*, 2011), no sentido de tirar proveito da capacidade dos agentes de se focarem em artefatos do ambiente nesse *framework*. Poder-se-ia substituir o envio de conhecimento de foco no conteúdo das mensagens de consulta por uma referenciação, na própria mensagem, indicando aos agentes quais artefatos do ambiente, que contêm o conhecimento de foco relevante para a consulta, devem ser focados.

A fim de permitir e facilitar a avaliação da abordagem proposta e de aplicações implementadas a partir dela, seria interessante também o desenvolvimento de modelos de verificação formal do sistema. Uma possibilidade, por exemplo, é o uso da linguagem Gwendolen (DENNIS; FARWER, 2008), uma linguagem de programação orientada a agentes que possibilita a verificação formal do comportamento dos agentes. Outra ideia é o desenvolvimento de *benchmarks* que permitam realizar análises empíricas por meio da execução das implementações, o que também permitiria a realização de comparações diversas, tais como comparações de tempo de execução.

Uma possível aplicação que poderia tirar proveito da abordagem proposta, além daquelas apresentadas no Capítulo 1, é a análise e detecção de notícias falsas (*fake news*). Estruturas de argumentos poderiam ser geradas com base na similaridade das informações e na confiabilidade das fontes, e assim permitir analisar como as *fake news* são comumente estruturadas.

Novas alternativas de cálculo de força poderiam também ser propostas e comparadas.

Por exemplo, poder-se-ia pensar em incluir a superioridade entre regras, ou atribuir graus de certeza que um agente possui em regras específicas, como proposto em (BORDINI *et al.*, 2007), o que poderia ser utilizado como mais um parâmetro no cálculo de força de argumentos. Outra ideia seria incluir a consideração de uma hierarquia de especialistas ou de autoridades, de modo que argumentos gerados a partir de conhecimentos vindos dessas autoridades fossem considerados mais fortes. Nesse mesmo sentido, poder-se-ia tirar proveito de um modelo organizacional para SMAs, como é o caso do Moise (HANNOUN *et al.*, 2000), que define uma estrutura organizacional em termos de grupos e papéis. É interessante também, para fins de implementação, o uso do *framework* JaCaMo (BOISSIER *et al.*, 2013), que inclui a linguagem de programação orientada a agentes Jason, o *framework* CArtAgO e o Moise. Por fim, para que seja possível realizar comparações entre os métodos de cálculo e comparação de força, é necessária também a definição de critérios específicos, o que pode ser também uma ideia para trabalhos futuros.

Uma investigação sobre a possibilidade de soluções híbridas – no sentido de serem em parte centralizadas – baseadas no modelo proposto também poderia ser realizada. Isso seria interessante, por exemplo, em um contexto de aplicação em cidades inteligentes, onde poderiam haver diferentes grupos de agentes, situados em diferentes regiões da cidade, capazes de se comunicar por meio de um agente mediador. Também é possível vislumbrar aplicações em que é necessário um grau de confiabilidade maior acerca de determinados assuntos ou decisões, o que possivelmente requereria a existência de um agente centralizado que detém a "palavra final" em determinados casos.

Outra possível oportunidade de trabalhos futuros seria integrar a abordagem apresentada a ontologias, derivando-se ou representando-se as regras locais com base nos indivíduos, axiomas e regras de inferência definidas na ontologia, integrando-se também regras de mapeamento, de modo similar ao que é feito em (JOSEPH *et al.*, 2016).

Conforme discutido no decorrer do texto, especialmente quanto à capacidade do algoritmo proposto de realmente construir as estruturas de argumentação durante o processo de raciocínio, seria também interessante estudar mecanismos de aprendizagem ou revisão de crenças que tirem proveito das estruturas de argumentação e dos valores de força de argumentos para atualizar as crenças dos agentes, assim como mecanismos que permitam construir explicações a partir de tais estruturas. Por exemplo, no cenário dos coletores de cogumelos, poder-se-ia pensar na possibilidade do agente da Alice aprender a identificar uma *amanita* da primavera com base em suas características e a saber que tal tipo de cogumelo é comestível, com base nos argumentos

que foram gerados a partir de sua consulta. No mesmo cenário, poder-se-ia pensar em como o agente da Alice, com base nos argumentos, poderia lhe explicar o porquê de ter sugerido que o cogumelo é comestível, de forma que a Alice possa compreender os motivos e assim também desenvolver seu conhecimento sobre cogumelos. Alguns trabalhos na linha de revisão de crenças que podem servir de base ou como trabalhos correlatos, são: (PAGLIERI; CASTELFRANCHI, 2004), (FALAPPA *et al.*, 2009), (DA COSTA PEREIRA *et al.*, 2011) e (DILLER *et al.*, 2018) . Alguns trabalhos na linha de geração de explicações são: (FAN; TONI, 2015) e (RAGO *et al.*, 2018).

No escopo do conceito de foco, estudos poderiam ser realizados a fim de definir mais concretamente esse conceito, visto que este trabalho apresenta apenas uma definição exemplificativa. Isso envolve estudos sobre que conhecimentos definem o foco de um agente, e qual sua relação com o estado e as atitudes mentais do agente, e com o ambiente e estrutura organizacional no qual o agente está inserido. Poderiam ser investigados também mecanismos que permitam delimitar, com base nas percepções e nas atitudes mentais dos agentes, o conhecimento que deve ser considerado como foco.

Uma comparação da abordagem proposta neste trabalho com pesquisas na área da Psicologia sobre o sistema de juiz-conselheiro (do Inglês, *judge-advisor system*) (SNIEZEK; BUCKLEY, 1995; SCHULTZE *et al.*, 2017) poderia ser frutífera no sentido de enriquecer o trabalho com ideias dessa área. Por exemplo, novos métodos de cálculo de força poderiam ser propostos com base em resultados e discussões desse tema de pesquisa.

No escopo da argumentação, poderiam ser propostos novos tipos de ataque e derrota. Neste trabalho, é proposto somente o ataque do tipo *rebutting*, que ocorre quando a conclusão de um argumento é contrária à conclusão de outro argumento. Um tipo diferente de ataque que poderia ser estudado é o *undercutting* conforme a noção apresentada no modelo ASPIC+: um ataque a uma regra de inferência (MODGIL; PRAKKEN, 2014). Também poderia ser estudada a consideração de diferentes naturezas de ataques, tais como a superfluidade, na qual dois ou mais argumentos levam à mesma conclusão ou servem para o mesmo fim, sendo possível assim descartar um ou mais deles, e a incompatibilidade instrumental, ou de recursos, no qual dois ou mais argumentos exigem o uso de recursos limitados compartilhados (MORVELI-ESPINOZA *et al.*, 2021).

Outras ideias interessantes a serem exploradas incluem realizar estudos comparativos entre o DDRMAS e outras abordagens de argumentação estruturada, tais como a DeLP e o

ASPIC+, buscando identificar correspondências entre eles e possíveis melhorias e evoluções em tais abordagens. Poder-se-ia também tentar realizar integrações ou conversões para *frameworks* de argumentação abstrata de Dung, o que permitiria estudar propriedades e o possível uso de uma gama maior de semânticas de argumentação no DDRMAS. Isto também abriria caminho para comparações e possíveis integrações com outras abordagens, tal como o ACS de Brewka e Eiter (2009), que adiciona à argumentação abstrata diversas características, incluindo meta-argumentação e argumentação hierárquica. Outra possibilidade é o estudo de alternativas de conjuntos de valores-verdade possíveis, tal como a inclusão do valor-verdade *unknown* para os casos em que um *r-literal* similar não é encontrado pelo agente, similarmente ao que é feito em DeLP.

REFERÊNCIAS

ADJIMAN, Philippe; CHATALIC, Philippe; GOASDOUÉ, François; ROUSSET, Marie-Christine; SIMON, Laurent. Distributed reasoning in a peer-to-peer setting: Application to the semantic web. **Journal of Artificial Intelligence Research**, v. 25, p. 269–314, 2006.

AMGOUD, Leila; CAYROL, Claudette. Inferring from inconsistency in preference-based argumentation frameworks. **Journal of Automated Reasoning**, Springer, v. 29, n. 2, p. 125–169, 2002.

ANDREOLI, Roberto; COROLLA, Angela; FAGGIANO, Armando; MALANDRINO, Delfina; PIROZZI, Donato; RANALDI, Mirta; SANTANGELO, Gianluca; SCARANO, Vittorio. A framework to design, develop, and evaluate immersive and collaborative serious games in cultural heritage. **Journal on Computing and Cultural Heritage (JOCCH)**, ACM New York, NY, USA, v. 11, n. 1, p. 1–22, 2017.

ANTONIOU, Grigoris; BILLINGTON, David; GOVERNATORI, Guido; MAHER, Michael J. A flexible framework for defeasible logics. *In*: KAUTZ, Henry A.; PORTER, Bruce W. (Ed.). **Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence**. Austin, Texas, USA: AAAI Press / The MIT Press, 2000. p. 405–410.

ANTONIOU, Grigoris; BILLINGTON, David; GOVERNATORI, Guido; MAHER, Michael J. Representation results for defeasible logic. **ACM Transactions on Computational Logic (TOCL)**, ACM New York, NY, USA, v. 2, n. 2, p. 255–287, 2001.

ANTONIOU, Grigoris; GROTH, Paul; HARMELEN, Frank van; HOEKSTRA, Rinke. **A Semantic Web Primer**. 3. ed. *[S.l.]*: MIT Press, 2012.

AZUMA, Shun ichi; YOSHIMURA, Ryota; SUGIE, Toshiharu. Broadcast control of multi-agent systems. **Automatica**, v. 49, n. 8, p. 2307–2316, 2013.

BAKER, Henry G.; HEWITT, Carl. The incremental garbage collection of processes. *In*: LOW, James (Ed.). **Proceedings of the 1977 Symposium on Artificial Intelligence and Programming Languages**. *[S.l.]*: ACM, 1977. p. 55–59.

BAO, Jie; VOUTSADAKIS, George; SLUTZKI, Giora; HONAVAR, Vasant. Package-based description logics. *In*: **Modular Ontologies**. *[S.l.]*: Springer, 2009. p. 349–371.

BECHHOFER, Sean; HARMELEN, Frank Van; HENDLER, Jim; HORROCKS, Ian; MCGUINNESS, Deborah L; PATEL-SCHNEIDER, Peter F; STEIN, Lynn Andrea *et al.* Owl web ontology language reference. **W3C Recommendation**, v. 10, n. 02, 2004.

BELLIFEMINE, Fabio; POGGI, Agostino; RIMASSA, Giovanni. Jade–a fipa-compliant agent framework. *In*: **Proceedings of PAAM**. *[S.l.: s.n.]*, 1999. v. 99, n. 97-108, p. 33.

BENCH-CAPON, Trevor. Persuasion in practical argument using value-based argumentation frameworks. **Journal of Logic and Computation**, Oxford University Press, v. 13, n. 3, p. 429–448, 2003.

BENERECETTI, Massimo; BOUQUET, Paolo; GHIDINI, Chiara. Contextual reasoning distilled. **Journal of Experimental & Theoretical Artificial Intelligence**, Taylor & Francis, v. 12, n. 3, p. 279–305, 2000.

BENFERHAT, Salem; DUBOIS, Didier; PRADE, Henri. Argumentative inference in uncertain and inconsistent knowledge bases. *In*: HECKERMAN, David; MAMDANI, E. H. (Ed.). **UAI '93: Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence**. Washington, DC, USA: Morgan Kaufmann, 1993. p. 411–419.

BERARIU, Tudor. An argumentation framework for bdi agents. *In*: **Intelligent Distributed Computing VII**. *[S.l.]*: Springer, 2014. p. 343–354.

BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora. The semantic web. **Scientific American**, JSTOR, v. 284, n. 5, p. 34–43, 2001.

BIKAKIS, Antonis; ANTONIOU, Grigoris. **Defeasible Contextual Reasoning in Ambient Intelligence**. 2009. Tese (Doutorado) — Computer Science Department, University of Crete, 2009.

BIKAKIS, A.; ANTONIOU, G. Defeasible contextual reasoning with arguments in ambient intelligence. **IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 11, p. 1492–1506, 2010.

BIKAKIS, Antonis; ANTONIOU, Grigoris. Partial preferences and ambiguity resolution in contextual defeasible logic. *In*: DELGRANDE, James P.; FABER, Wolfgang (Ed.). **Proceedings of the 11th International Conference, LPNMR 2011**. *[S.l.]*: Springer, 2011. (Lecture Notes in Computer Science, v. 6645), p. 193–198.

BIKAKIS, Antonis; ANTONIOU, Grigoris; HASAPIS, Panayiotis. Strategies for contextual reasoning with conflicts in ambient intelligence. **Knowledge and Information Systems**, Springer, v. 27, n. 1, p. 45–84, 2011.

BINAS, Arnold; MCILRAITH, Sheila A. Peer-to-peer query answering with inconsistent knowledge. *In*: **Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning**. Sydney, Australia: *[s.n.]*, 2008. p. 329–339.

BOISSIER, Olivier; BORDINI, Rafael H; HÜBNER, Jomi F; RICCI, Alessandro; SANTI, Andrea. Multi-agent oriented programming with jacamo. **Science of Computer Programming**, Elsevier, v. 78, n. 6, p. 747–761, 2013.

BORDINI, Rafael H.; HÜBNER, Jomi Fred; WOOLDRIDGE, Michael. **Programming Multi-Agent Systems in AgentSpeak Using Jason**. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2007.

BRACHMAN, Ronald; LEVESQUE, Hector. **Knowledge Representation and Reasoning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

BREWKA, Gerhard; EITER, Thomas. Equilibria in heterogeneous nonmonotonic multi-context systems. *In*: **Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence**. Vancouver, British Columbia, Canada: AAAI Press, 2007. p. 385–390.

BREWKA, Gerhard; EITER, Thomas. Argumentation context systems: A framework for abstract group argumentation. *In*: ERDEM, Esra; LIN, Fangzhen; SCHAUB, Torsten (Ed.). **Proceedings of the 10th International Conference, LPNMR 2009**. *[S.l.]*: Springer, 2009. (Lecture Notes in Computer Science, v. 5753), p. 44–57.

BREWKA, Gerhard; ELLMAUTHALER, Stefan; GONÇALVES, Ricardo; KNORR, Matthias; LEITE, João; PÜHRER, Jörg. Reactive multi-context systems: Heterogeneous reasoning in dynamic environments. **Artificial Intelligence**, Elsevier, v. 256, p. 68–104, 2018.

BREWKA, Gerhard; NIEMELä, Ilkka; TRUSZCZYńSKI, Mirosław. Chapter 6 nonmonotonic reasoning. *In*: van Harmelen, Frank; LIFSCHITZ, Vladimir; PORTER, Bruce (Ed.). **Handbook of Knowledge Representation**. *[S.l.]*: Elsevier, 2008, (Foundations of Artificial Intelligence, v. 3). p. 239–284.

BREWKA, Gerhard; ROELOFSEN, Floris; SERAFINI, Luciano. Contextual default reasoning. *In*: **Proceedings of the 20th International Joint Conference on Artifical Intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. (IJCAI'07), p. 268–273.

BROOKS, Rodney A. Intelligence without representation. **Artificial Intelligence**, Elsevier, v. 47, n. 1-3, p. 139–159, 1991.

BUVAC, Sasa; MASON, Ian A. Propositional logic of context. *In*: FIKES, Richard; LEHNERT, Wendy G. (Ed.). **Proceedings of the 11th National Conference on Artificial Intelligence**. *[S.l.]*: AAAI Press / The MIT Press, 1993. p. 412–419.

CAMINADA, Martin; AMGOUD, Leila. On the evaluation of argumentation formalisms. **Artificial Intelligence**, Elsevier, v. 171, n. 5-6, p. 286–310, 2007.

CASTELFRANCHI, Christiano; FALCONE, Rino. **Trust theory: A socio-cognitive and computational model**. *[S.l.]*: John Wiley & Sons, 2010. v. 18.

CHATALIC, Ph.; NGUYEN, G. H.; ROUSSET, M. Ch. Reasoning with inconsistencies in propositional peer-to-peer inference systems. *In*: **Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence**. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2006. p. 352–356.

CROSS, Valerie. Uncertainty in the automation of ontology matching. *In*: IEEE. **Fourth International Symposium on Uncertainty Modeling and Analysis. ISUMA 2003.** *[S.l.]*, 2003. p. 135–140.

DA COSTA PEREIRA, Célia; TETTAMANZI, Andrea; VILLATA, Serena. Changing one's mind: Erase or rewind? *In*: WALSH, Toby (Ed.). **Proceedings of the 22nd International Joint Conference on Artificial Intelligence. IJCAI 2011**. Barcelona, Catalonia, Spain: IJCAI/AAAI, 2011. p. 164–171.

DAO-TRAN, Minh; EITER, Thomas; FINK, Michael; KRENNWALLNER, Thomas. Dynamic distributed nonmonotonic multi-context systems. **Nonmonotonic Reasoning, Essays Celebrating its 30th Anniversary, Studies in Logic**, v. 31, 2010.

DAO-TRAN, Minh; EITER, Thomas; FINK, Michael; KRENNWALLNER, Thomas. Distributed evaluation of nonmonotonic multi-context systems. **Journal of Artificial Intelligence Research**, v. 52, p. 543–600, 2015.

DENNIS, Louise A; FARWER, Berndt. Gwendolen: A bdi language for verifiable agents. *In*: **Proceedings of the AISB 2008 Symposium on Logic and the Simulation of Interaction and Reasoning, Society for the Study of Artificial Intelligence and Simulation of Behaviour**. *[S.l.: s.n.]*, 2008. p. 16–23.

DILLER, Martin; HARET, Adrian; LINSBICHLER, Thomas; RÜMMELE, Stefan; WOLTRAN, Stefan. An extension-based approach to belief revision in abstract argumentation. **International Journal of Approximate Reasoning**, Elsevier, v. 93, p. 395–423, 2018.

DUNG, Phan Minh. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. **Artificial Intelligence**, Elsevier, v. 77, n. 2, p. 321–357, 1995.

DURFEE, Edmund H; LESSER, Victor R. Negotiating task decomposition and allocation using partial global planning. *In*: **Distributed Artificial Intelligence**. *[S.l.]*: Elsevier, 1989. p. 229–243.

ERENLI, Kai. Gamify your teaching-using location-based games for educational purposes. **International Journal of Advanced Corporate Learning (iJAC)**, Kassel University Press GmbH, v. 6, n. 2, p. 22–27, 2013.

EUZENAT, Jérôme; SHVAIKO, Pavel *et al.* **Ontology Matching**. *[S.l.]*: Springer, 2007. v. 18.

FALAPPA, Marcelo Alejandro; KERN-ISBERNER, Gabriele; SIMARI, Guillermo Ricardo. Belief revision and argumentation theory. *In*: **Argumentation in Artificial Intelligence**. *[S.l.]*: Springer, 2009. p. 341–360.

FAN, Xiuyi; TONI, Francesca. On computing explanations in argumentation. *In*: BONET, Blai; KOENIG, Sven (Ed.). **Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence**. Austin, Texas, USA: AAAI Press, 2015. p. 1496–1502.

FERBER, Jacques. **Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence**. 1. ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

FERRANDO, Sergio Pajares; ONAINDIA, Eva. Context-aware multi-agent planning in intelligent environments. **Information Sciences**, v. 227, p. 22 – 42, 2013.

FRINTROP, Simone. Towards attentive robots. **Paladyn J. Behav. Robotics**, v. 2, n. 2, p. 64–70, 2011.

GANDON, Fabien. **Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web**. 2002. Tese (Doutorado) — Université Nice Sophia Antipolis, 2002.

GARCÍA, Alejandro J; PRAKKEN, Henry; SIMARI, Guillermo R. A comparative study of some central notions of aspic+ and delp. **Theory and Practice of Logic Programming**, Cambridge University Press, v. 20, n. 3, p. 358–390, 2020.

GARCÍA, Alejandro J.; SIMARI, Guillermo R. Defeasible logic programming: An argumentative approach. **Theory Pract. Log. Program.**, Cambridge University Press, New York, NY, USA, v. 4, n. 2, p. 95–138, jan. 2004.

GARCÍA, Alejandro J; SIMARI, Guillermo R. Defeasible logic programming: Delp-servers, contextual queries, and explanations for answers. **Argument & Computation**, IOS Press, v. 5, n. 1, p. 63–88, 2014.

GHIDINI, Chiara; GIUNCHIGLIA, Fausto. Local models semantics, or contextual reasoning=locality+compatibility. **Artificial Intelligence**, v. 127, n. 2, p. 221 – 259, 2001.

GIUNCHIGLIA, Fausto. Contextual reasoning. **Epistemologia, special issue on I Linguaggi e le Macchine**, v. 16, p. 345–364, 1993.

GIUNCHIGLIA, Fausto; SERAFINI, Luciano. Multilanguage hierarchical logics, or: How we can do without modal logics. **Artificial Intelligence**, v. 65, n. 1, p. 29 – 70, 1994.

GIUNCHIGLIA, Fausto; SERAFINI, Luciano; GIUNCHIGLIA, Enrico; FRIXIONE, Marcello. Non-omniscient belief as context-based reasoning. *In*: **Proceedings of the 13th International Joint Conference on Artifical Intelligence - Volume 1**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. (IJCAI'93), p. 548–554.

GOVERNATORI, Guido; MAHER, Michael J; ANTONIOU, Grigoris; BILLINGTON, David. Argumentation semantics for defeasible logic. **Journal of Logic and Computation**, Oxford University Press, v. 14, n. 5, p. 675–702, 2004.

GROSZ, Barbara Jean. **The Representation and Use of Focus in Dialogue Understanding.** 1977. Tese (Doutorado), University of California, Berkeley, 1977.

GUHA, Ramanathan. **Contexts: A Formalization and Some Applications**. 1992. Tese (Doutorado), Stanford, CA, USA, 1992.

HANNOUN, Mahdi; BOISSIER, Olivier; SICHMAN, Jaime S; SAYETTAT, Claudette. Moise: An organizational model for multi-agent systems. *In*: **Advances in Artificial Intelligence**. *[S.l.]*: Springer, 2000. p. 156–165.

HARTIG, Olaf. Sparql for a web of linked data: Semantics and computability. *In*: SIMPERL, Elena; CIMIANO, Philipp; POLLERES, Axel; CORCHO, Oscar; PRESUTTI, Valentina (Ed.). **The Semantic Web: Research and Applications**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 8–23.

HENRICKSEN, Karen; INDULSKA, Jadwiga. Modelling and using imperfect context information. *In*: IEEE. **Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops**. Orlando, FL, USA, 2004. p. 33–37.

ITTI, Laurent; KOCH, Christof. Computational modelling of visual attention. **Nature Reviews Neuroscience**, Nature Publishing Group, v. 2, n. 3, p. 194–203, 2001.

JENNINGS, Nicholas R. On agent-based software engineering. **Artificial intelligence**, Elsevier, v. 117, n. 2, p. 277–296, 2000.

JOSEPH, Mathew; KUPER, Gabriel; MOSSAKOWSKI, Till; SERAFINI, Luciano. Query answering over contextualized rdf/owl knowledge with forall-existential bridge rules: Decidable finite extension classes. **Semantic Web**, IOS Press, v. 7, n. 1, p. 25–61, 2016.

KUTZ, Oliver; LUTZ, Carsten; WOLTER, Frank; ZAKHARYASCHEV, Michael. E-connections of abstract description systems. **Artificial Intelligence**, v. 156, n. 1, p. 1–73, 2004.

LAM, Ho-Pun; GOVERNATORI, Guido; RIVERET, Régis. On aspic$^+$ and defeasible logic. *In*: BARONI, Pietro; GORDON, Thomas F.; SCHEFFLER, Tatjana; STEDE, Manfred (Ed.). **Proceedings of Computational Models of Argument - COMMA 2016**. Potsdam, Germany: IOS Press, 2016. (Frontiers in Artificial Intelligence and Applications, v. 287), p. 359–370.

LORETO, Ines Di; MORA, Simone; DIVITINI, Monica. Collaborative serious games for crisis management: an overview. *In*: IEEE. **2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises**. Toulouse, France, 2012. p. 352–357.

MAARALA, A. I.; SU, X.; RIEKKI, J. Semantic reasoning for context-aware internet of things applications. **IEEE Internet of Things Journal**, v. 4, n. 2, p. 461–473, 2017.

MAHER, Michael J. Propositional defeasible logic has linear complexity. **Theory Pract. Log. Program.**, Cambridge University Press, USA, v. 1, n. 6, p. 691–711, 2001.

MCCARTHY, John. Programs with common sense. *In*: **Proceedings of the Teddington Conference on the Mechanization of Thought Processes**. London: Her Majesty's Stationary Office, 1959. p. 75–91.

MCCARTHY, John. Generality in artificial intelligence. **Commun. ACM**, ACM, New York, NY, USA, v. 30, n. 12, p. 1030–1035, dez. 1987.

MCCARTHY, John; BUVAC, Sasa. **Formalizing Context (Expanded Notes)**. Stanford University: Relatório técnico. Stanford, CA, USA, 1994.

MCEWAN, Gregor; GREENBERG, Saul. Supporting social worlds with the community bar. *In*: **Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work**. New York, NY, USA: ACM, 2005. p. 21–30.

MEKURIA, Dagmawi Neway; SERNANI, Paolo; FALCIONELLI, Nicola; DRAGONI, Aldo Franco. Smart home reasoning systems: a systematic literature review. **Journal of Ambient Intelligence and Humanized Computing**, Springer, p. 1–18, 2019.

MILLER, George A. Wordnet: a lexical database for english. **Communications of the ACM**, ACM New York, NY, USA, v. 38, n. 11, p. 39–41, 1995.

MINSKY, Marvin. **Society of Mind**. *[S.l.]*: Simon and Schuster, 1988.

MODGIL, S. Hierarchical argumentation. *In*: FISHER, Michael; HOEK, Wiebe van der; KONEV, Boris; LISITSA, Alexei (Ed.). **Logics in Artificial Intelligence**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 319–332.

MODGIL, Sanjay; PRAKKEN, Henry. The aspic+ framework for structured argumentation: a tutorial. **Argument & Computation**, IOS Press, v. 5, n. 1, p. 31–62, 2014.

MORVELI-ESPINOZA, Mariela; NIEVES, Juan Carlos; TACLA, Cesar Augusto. Dealing with conflicts between human activities: An argumentation-based approach. *In*: **AAAI-21 Workshop on Plan Activity and Intent Recognition (PAIR 2021), Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)**. Virtual: *[s.n.]*, 2021.

NAVARRO, Gonzalo. A guided tour to approximate string matching. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 33, n. 1, p. 31–88, 2001.

NUTE, Donald. Defeasible logic. *In*: **Proceedings of the 14th International Conference on Applications of Prolog, INAP 2001**. Tokyo, Japan: The Prolog Association of Japan, 2001. p. 87–114.

OGSTON, Elth; VASSILIADIS, Stamatis. Local distributed agent matchmaking. *In*: BATINI, Carlo; GIUNCHIGLIA, Fausto; GIORGINI, Paolo; MECELLA, Massimo (Ed.). **Proceedings of the 9th International Conference on Cooperative Information Systems, CoopIS 2001**. Trento, Italy: Springer, 2001. (Lecture Notes in Computer Science, v. 2172), p. 67–79.

PAGLIERI, Fabio; CASTELFRANCHI, Cristiano. Revising beliefs through arguments: Bridging the gap between argumentation and belief revision in MAS. *In*: RAHWAN, Iyad; MORAITIS, Pavlos; REED, Chris (Ed.). **First International Workshop on Argumentation in Multi-Agent Systems, ArgMAS 2004, Revised Selected and Invited Papers**. New York, NY: Springer, 2004. (Lecture Notes in Computer Science, v. 3366), p. 78–94.

PANISSON, Alison R.; BORDINI, Rafael H. Knowledge representation for argumentation in agent-oriented programming languages. *In*: **5th Brazilian Conference on Intelligent Systems, BRACIS 2016**. Recife, Brazil: IEEE Computer Society, 2016. p. 13–18.

PANISSON, Alison R.; MELO, Victor S.; BORDINI, Rafael H. Using preferences over sources of information in argumentation-based reasoning. *In*: **5th Brazilian Conference on Intelligent Systems, BRACIS 2016**. Recife, Brazil: IEEE Computer Society, 2016. p. 31–36.

PARSONS, S. Current approaches to handling imperfect information in data and knowledge bases. **IEEE Transactions on Knowledge and Data Engineering**, v. 8, n. 3, p. 353–372, 1996.

PARSONS, Simon; COHEN, Andrea; CHESÑEVAR, C. On the relationship between delp and aspic+. **Argumentation-based Proofs of Endearment. Essays in Honor of Guillermo R. Simari on the Occasion of his 70th Birthday**, College Publications, London, p. 293–323, 2018.

PARSONS, Simon; TANG, Yuqing; SKLAR, Elizabeth; MCBURNEY, Peter; CAI, Kai. Argumentation-based reasoning in agents with varying degrees of trust. *In*: SONENBERG, Liz; STONE, Peter; TUMER, Kagan; YOLUM, Pinar (Ed.). **Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)**. Taipei, Taiwan: IFAAMAS, 2011. p. 879–886.

PEDERSEN, Ted; PATWARDHAN, Siddharth; MICHELIZZI, Jason. Wordnet: Similarity - measuring the relatedness of concepts. *In*: MCGUINNESS, Deborah L.; FERGUSON, George (Ed.). **Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence**. San Jose, California, USA: AAAI Press / The MIT Press, 2004. p. 1024–1025.

PINYOL, Isaac; SABATER-MIR, Jordi. Computational trust and reputation models for open multi-agent systems: a review. **Artificial Intelligence Review**, Springer, v. 40, n. 1, p. 1–25, 2013.

POLLOCK, John L. Defeasible reasoning. **Cognitive Science**, Elsevier, v. 11, n. 4, p. 481–518, 1987.

PRAKKEN, Henry; VREESWIJK, Gerard. Logics for defeasible argumentation. **Handbook of Philosophical Logic**, Springer, p. 219–318, 2001.

RAGO, Antonio; COCARASCU, Oana; TONI, Francesca. Argumentation-based recommendations: Fantastic explanations and how to find them. *In*: LANG, Jérôme (Ed.). **Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018**. Stockholm, Sweden: *[s.n.]*, 2018. p. 1949–1955.

RAKIB, Abdur; UDDIN, Ijaz. An efficient rule-based distributed reasoning framework for resource-bounded systems. **Mobile Networks and Applications**, Springer, v. 24, n. 1, p. 82–99, 2019.

RAO, Anand S.; GEORGEFF, Michael P. BDI agents: From theory to practice. *In*: LESSER, Victor R.; GASSER, Les (Ed.). **Proceedings of the First International Conference on Multiagent Systems**. San Francisco, California, USA: The MIT Press, 1995. p. 312–319.

RICCI, Alessandro; PIUNTI, Michele; VIROLI, Mirko. Environment programming in multi-agent systems: an artifact-based perspective. **Autonomous Agents and Multi-Agent Systems**, Springer, v. 23, n. 2, p. 158–192, 2011.

ROBINSON, John Alan. A machine-oriented logic based on the resolution principle. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 12, n. 1, p. 23–41, 1965.

RODDEN, Tom. Populating the application: A model of awareness for cooperative applications. *In*: ACKERMAN, Mark S.; OLSON, Gary M.; OLSON, Judith S. (Ed.). **Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work**. Boston, MA, USA: ACM, 1996. p. 87–96.

RUSSELL, Stuart J; NORVIG, Peter. **Artificial Intelligence: a Modern Approach**. *[S.l.]*: Pearson Education Limited, 2016.

SÁ, Samy; ALCÂNTARA, João F. L. Cooperative dialogues with conditional arguments. *In*: HOEK, Wiebe van der; PADGHAM, Lin; CONITZER, Vincent; WINIKOFF, Michael (Ed.). **Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012**. Valencia, Spain: IFAAMAS, 2012. p. 501–508.

SCHULTZE, Thomas; MOJZISCH, Andreas; SCHULZ-HARDT, Stefan. On the inability to ignore useless advice. **Experimental Psychology**, Hogrefe Publishing, 2017.

SERAFINI, Luciano; BORGIDA, Alexander; TAMILIN, Andrei. Aspects of distributed and modular ontology reasoning. *In*: KAELBLING, Leslie Pack; SAFFIOTTI, Alessandro (Ed.). **Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI-05**. Edinburgh, Scotland: Professional Book Center, 2005. p. 570–575.

SERAFINI, Luciano; BOUQUET, Paolo. Comparing formal theories of context in ai. **Artificial Intelligence**, Elsevier, v. 155, n. 1, p. 41 – 67, 2004.

SHOHAM, Yoav. Agent-oriented programming. **Artificial intelligence**, Elsevier, v. 60, n. 1, p. 51–92, 1993.

SNIEZEK, Janet A.; BUCKLEY, Timothy. Cueing and cognitive conflict in judge-advisor decision making. **Organizational Behavior and Human Decision Processes**, v. 62, n. 2, p. 159–174, 1995.

SPERBER, Dan; WILSON, Deirdre. **Relevance: Communication and Cognition**. Cambridge, MA, USA: Harvard University Press, 1986.

STEINKUEHLER, Constance. Massively multiplayer online games as an educational technology: An outline for research. **Educational Technology Archive**, v. 48, p. 10–21, 2008.

SU, Xiang; LI, Pingjiang; RIEKKI, Jukka; LIU, Xiaoli; KILJANDER, Jussi; SOININEN, Juha-Pekka; PREHOFER, Christian; FLORES, Huber; LI, Yuhong. Distribution of semantic

reasoning on the edge of internet of things. *In*: IEEE. **Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)**. Athens, Greece, 2018. p. 1–9.

SYCARA, Katia; WIDOFF, Seth; KLUSCH, Matthias; LU, Jianguo. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. **Autonomous Agents and Multi-Agent Systems**, Springer, v. 5, n. 2, p. 173–203, 2002.

SYCARA, Katia P. Multiagent systems. **AI Magazine**, v. 19, n. 2, p. 79–79, 1998.

TANG, Yuqing; CAI, Kai; MCBURNEY, Peter; SKLAR, Elizabeth; PARSONS, Simon. Using argumentation to reason about trust and belief. **Journal of Logic and Computation**, OUP, v. 22, n. 5, p. 979–1018, 2012.

THIMM, Matthias; GARCIA, Alejandro J; KERN-ISBERNER, Gabriele; SIMARI, Guillermo R. Using collaborations for distributed argumentation with defeasible logic programming. *In*: **Proceedings of the 12th International Workshop on Non-Monotonic Reasoning (NMR'08)**. *[S.l.: s.n.]*, 2008. p. 179–188.

THIMM, Matthias; KERN-ISBERNER, Gabriele. A distributed argumentation framework using defeasible logic programming. *In*: **Proceedings of the 2008 Conference on Computational Models of Argument: COMMA 2008**. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2008. p. 381–392.

VEMULA, Anirudh; MUELLING, Katharina; OH, Jean. Social attention: Modeling attention in human crowds. *In*: **Proceedings of the 2018 IEEE International Conference on Robotics and Automation, ICRA**. Brisbane, Australia: IEEE, 2018. p. 1–7.

VIEIRA, Vaninha; TEDESCO, Patricia; SALGADO, Ana Carolina. Designing context-sensitive systems: An integrated approach. **Expert Systems with Applications**, Elsevier, v. 38, n. 2, p. 1119–1138, 2011.

WOOLDRIDGE, Michael J; JENNINGS, Nicholas R. Intelligent agents: Theory and practice. **The Knowledge Engineering Review**, v. 10, n. 2, p. 115–152, 1995.

ZAMBONELLI, Franco; JENNINGS, Nicholas R.; WOOLDRIDGE, Michael. Organisational abstractions for the analysis and design of multi-agent systems. *In*: CIANCARINI, Paolo; WOOLDRIDGE, Michael J. (Ed.). **Agent-Oriented Software Engineering**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 235–251.

APÊNDICES

APÊNDICE A – PROVAS

A.1 TEOREMAS 1 E 2 – COERÊNCIA E CONSISTÊNCIA DO *FRAMEWORK* DE ARGUMENTAÇÃO

As provas apresentadas são baseadas nas dadas por (GOVERNATORI *et al.*, 2004), uma vez que a semântica de argumentação em si é muito similar à apresentada em DL.

Teorema 1. Dado um DDRMAS $\mathcal{S}$ e um foco de consulta α:

- Um argumento não pode ser tanto justificado quanto rejeitado
- Um *r-literal* não pode ser tanto justificado quanto rejeitado

Prova: Suponha que exista um argumento que é tanto justificado quanto rejeitado. Seja n o menor índice tal que, para um argumento A, $A \in RArgs_{\mathcal{S}\alpha}$ ($JArgs_{\mathcal{S}\alpha}$) e $A \in J_n^{\mathcal{S}\alpha}$.

Pelas definições, existe um argumento B, apoiado por $JArgs_{\mathcal{S}\alpha}$, que derrota A, e B é *undercut* por $J_{n-1}^{\mathcal{S}\alpha}$. Logo existe um argumento C, apoiado por $J_{n-1}^{\mathcal{S}\alpha}$, que derrota um subargumento próprio B' de B. Visto que $B \in JArgs_{\mathcal{S}\alpha}$, então $B' \in JArgs_{\mathcal{S}\alpha}$ também, logo C é *undercut* por $JArgs_{\mathcal{S}\alpha}$, isto é, existe um argumento E, apoiado por $JArgs_{\mathcal{S}\alpha}$, que derrota um subargumento próprio C' de C. $C' \in J_{n-1}^{\mathcal{S}\alpha}$, visto que C' é um subargumento de um argumento apoiado por $J_{n-1}^{\mathcal{S}\alpha}$. Além disso, C' é rejeitado, pois é derrotado por um argumento (E) que é apoiado por $JArgs_{\mathcal{S}\alpha}$. No entanto, isso contradiz a minimalidade que foi pressuposta para n. Logo, a suposição original é falsa, e nenhum argumento pode ser tanto justificado quanto rejeitado.

A segunda parte do teorema segue facilmente do primeiro: se p é justificado, então existe um argumento para p em $JArgs_{\mathcal{S}\alpha}$. Pela primeira parte do teorema, esse argumento está em $Args_{\mathcal{S}\alpha} \setminus RArgs_{\mathcal{S}\alpha}$, pois está em $JArgs_{\mathcal{S}\alpha}$. Logo, se p é justificado, então não é rejeitado.

△

Teorema 2. Se o conjunto de argumentos justificados em um DDRMAS $\mathcal{S}$, $JArgs_{\mathcal{S}\alpha}$, contém dois argumentos com conclusões conflitantes, então ambos são argumentos estritos.

Prova. Sejam dois argumentos A e B com conclusões conflitantes. Suponha que B é estrito. Então, para que A seja aceitável n.q.d.r. a qualquer conjunto S, A deve ser estrito (caso contrário, B derrotaria A, e B não pode ser *undercut*, por ser estrito). Assim, por simetria, ou A e B são ambos estritos – o que, pelo pressuposto de consistência dos conjuntos de regras estritas, nunca deveria acontecer – , ou nenhum deles é estrito. Suponha então que ambos são argumentos derrotáveis válidos (uma vez que o argumento deve ser válido para que seja aceitável

por qualquer conjunto S), e B derrota A. Então A deveria ser rejeitado por ser derrotado por um argumento justificado (B). Pelo Teorema 2, sabe-se que isso não é possível: A não pode ser tanto justificado quanto rejeitado. Logo, nenhum par de argumentos derrotáveis justificados podem ter conclusões conflitantes.

$\triangle$

A.2 TEOREMA 3 – EQUIVALÊNCIA COM LÓGICA DERROTÁVEL

As provas apresentadas são baseadas nas dadas por Governatori *et al.* (2004), com suas devidas adaptações para a abordagem apresentada neste trabalho. Recomenda-se consultar a Seção 2.2.2.1, na qual constam as regras para derivação dos literais etiquetados Δ e ∂ em DL.

Teorema 3. Dado um *r-literal* $p \in V^R_{\mathcal{DL}(\mathcal{S},\alpha)}$:

1. $\exists A \in JArgs_{S\alpha}$ t.q. A é um argumento estrito e $Conc(A) = p$ se e somente se $\mathcal{DL}(\mathcal{S},\alpha) \vdash +\Delta p$.

2. $\exists A \in JArgs_{S\alpha}$ t.q. A é um argumento estrito e $Conc(A) = p$ se e somente se $\mathcal{DL}(\mathcal{S},\alpha) \vdash -\Delta p$.

3. $\exists A \in JArgs_{S\alpha}$ t.q. A é um argumento derrotável e $Conc(A) = p$ se e somente se $\mathcal{DL}(\mathcal{S},\alpha) \vdash +\partial p$.

4. $\exists A \in JArgs_{S\alpha}$ t.q. A é um argumento derrotável e $Conc(A) = p$ se e somente se $\mathcal{DL}(\mathcal{S},\alpha) \vdash -\partial p$.

Prova:

Caso 1 ($\Rightarrow$):

Base indutiva. Seja A um argumento estrito para p com altura 1, isto é, A é formado por uma regra r com corpo vazio; deste modo, é imediato que $+\Delta p$.

Passo indutivo. Suponhamos que a proposição seja verdadeira para argumentos de altura menor que n, e seja A um argumento estrito para p de altura n. Seja a regra que forma o argumento A uma regra estrita $r : p \leftarrow Body(r)$. Por construção, para cada $q \in Body(r)$, temos um argumento estrito de altura menor que n, portanto todos esses *r-literais* são justificados, e, pela hipótese indutiva, temos $+\Delta q$; portanto, as condições para derivar $+\Delta p$ são satisfeitas.

Caso 1 ($\Leftarrow$):

Base indutiva. A prova tem uma única entrada $P(1) = +\Delta p$, o que significa que existe uma regra r para p com corpo vazio. Portanto, existe um argumento base A tal que $Conc(A) = p$.

Passo indutivo. Suponhamos que a proposição seja verdadeira para teorias de prova de tamanho menor que n, e que $P(n+1) = +\Delta p$. Isto significa que existe uma regra estrita r para p tal que $\forall a \in Body(r), +\Delta p \in P(1..n)$. Pela hipótese indutiva, temos argumentos estritos para cada $q \in Body(r)$. Seja A o argumento com conclusão p e subargumentos cujas conclusões sejam cada $q \in Body(r)$. A partir disso, é imediato verificar que A é um argumento para p.

Caso 2 ($\Rightarrow$).

Prova pela contrapositiva: Suponhamos que $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\Delta p$. Construiremos, então, o argumento para p. Sabe-se que, se, para toda regra estrita com cabeça q, existe $q \in Body(r)$ t.q. $\mathcal{DL}(\mathcal{S},\alpha) \vdash -\Delta q$, então $\mathcal{DL}(\mathcal{S},\alpha) \vdash -\Delta p$, o que contradiz a suposição original desta prova. Logo, existe uma regra r estrita com cabeça q t.q., para todo $q \in Body(r)$, $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\Delta q$. Podemos, assim, construir um argumento parcial para p que começa com r e no qual toda premissa q satisfaz $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\Delta q$, e podemos fazer o mesmo para cada nó-folha para construir um argumento parcial mais profundo, até que se possa construir um argumento completo.

Caso 2 ($\Leftarrow$).

Base indutiva. A prova consiste de um único registro $P(1) = -\Delta p$. Isso é possível apenas se não existem regras estritas para p. Logo, é impossível que exista um argumento para p.

Passo indutivo. Suponhamos que a proposição seja verdadeira para provas de tamanho menor que n, e que $P(n+1) = -\Delta p$. Isto significa que, para cada regra estrita r para p, existe um *r-literal* $q \in Body(r)$ tal que $-\Delta q \in P(1..n)$. Pela hipótese indutiva, não existe argumento estrito para q, e, portanto, r não pode ser usada para construir um argumento estrito para p, e isso ocorre para toda regra estrita para p. Logo, um argumento estrito para p não pode ser construído.

Caso 3 ($\Rightarrow$). Para provar este caso, temos que usar uma indução dupla. A indução externa é no estágio de justificação dos argumentos, a partir do qual é verificada a indução sobre a altura das árvores que representam os argumentos no mesmo estágio de justificação.

Base indutiva. Começamos considerando os argumentos aceitáveis n.q.d.r. a $J_0^{\mathcal{S}\alpha}$, cuja altura é 1. Tais argumentos A são baseados em uma única regra r para p com corpo vazio. Se o argumento A for estrito, então r é estrita e, portanto, $\mathcal{DL}(\mathcal{S},\alpha) \vdash +\Delta p$ e, consequentemente, $\mathcal{DL}(\mathcal{S},\alpha) \vdash +\partial p$. Se o argumento for derrotável, então r é uma regra derrotável, e nesse caso sabemos que todo argumento que derrota A é *undercut* por $J_0^{\mathcal{S}\alpha}$, que é um conjunto vazio. Seja B um argumento que derrota A que é baseado em uma regra s para $\sim p$. Uma vez que A é

aceitável n.q.d.r. a $J_0^{\mathcal{S}\alpha}$, B é *undercut* por $J_0^{\mathcal{S}\alpha}$; portanto existe um argumento C que derrota um subargumento de B. O argumento C é, portanto, apoiado por $J_0^{\mathcal{S}\alpha}$. Logo, C é baseado em uma única regra para $\sim q$ com corpo vazio, para algum q que é premissa de B. Neste ponto, pode-se verificar que as condições para provar $-\partial q$ são satisfeitas, o que é uma condição para que s seja ∂-descartável (condição 2.3.1 de $+\partial$).

Demonstramos agora que, para toda regra s com cabeça $\sim p$, existe $q' \in Body(s)$ t.q $\mathcal{DL}(\mathcal{S},\alpha) \vdash -\partial q'$. Suponhamos que, para obter uma contradição, existe uma regra s com cabeça $\sim p$ t.q. para todo $q' \in Body(s)$, $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\partial q'$. Podemos construir um argumento para $\sim p$ como segue.

Inicialmente, o argumento parcial – ainda sem seus subargumentos construídos – possui conclusão $\sim p$. Para cada $q' \in Body(s)$ tal que existe um *r-literal* q'' similar o suficiente a q' tal que $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\partial q''$, sabemos, pelas condições de $-\partial$ (ver Seção 2.2.2.1), que, se $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\partial q''$, então:

(1) $\mathcal{DL}(\mathcal{S},\alpha) \vdash -\Delta q''$, ou

(2) $\exists r \in KB_{\mathcal{S}\alpha}$ t.q. $Head(r) = q''$ t.q. $\forall q_r \in Body(r)$, $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\partial q_r$

Se (1), então, pelo caso 2 do teorema, existe um argumento estrito B para q''. Pode-se, então, adicionar o argumento B como subargumento do argumento parcial para $\sim p$. Caso contrário, (2) implica que existe uma regra r para q'' tal que $\forall q_r \in Body(r)$, $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\partial q_r$. Assim, é possível criar um argumento com conclusão q'' e nós-filhos referentes a cada $q_r \in Body(r)$, e adicioná-lo como subargumento do argumento parcial para $\sim p$.

Podemos, assim, repetir essa construção para cada antecedente de cada regra que pode ser aplicada. Logo, é possível construir um argumento para $\sim p$ tal que todo *r-literal* q'' que é premissa desse argumento satisfaz $\mathcal{DL}(\mathcal{S},\alpha) \nvdash \partial q''$.

No entanto, como já observamos, para todo argumento B existe um argumento C que derrota um subargumento de B que conclui q, e que $\mathcal{DL}(\mathcal{S},\alpha) \vdash -\partial q$. Assim, temos uma contradição, e, consequentemente, $\forall s \in KB_{\mathcal{S}\alpha}$ t.q. $Head(s) = \sim p$, $\exists q' \in Body(s)$ t.q. $\mathcal{DL}(\mathcal{S},\alpha) \vdash -\partial q'$. Considerando, então, o argumento A para p, temos que $\mathcal{DL}(\mathcal{S},\alpha) \vdash +\partial p$.

Para completar a prova da base indutiva, é necessário mostrar que a proposição é verdadeira para argumentos arbitrários em $J_1^{\mathcal{S}\alpha}$. Portanto, assumimos, por indução, que a proposição é verdadeira para *r-literais* que possuem argumentos em $J_1^{\mathcal{S}\alpha}$ cuja altura é menor que h. Seja A um argumento para p com altura h. Todos os subargumentos e argumentos na árvore de

argumentação cuja raiz é A possuem altura menor que h. Portanto, se A é baseado numa regra r, por indução temos que $\forall q_r \in Body(r)$, $\mathcal{DL}(\mathcal{S},\alpha) \vdash +\partial q_r$.

Passo indutivo. É possível repetir a prova da base indutiva observando que os argumentos que realizam *undercut* são apoiados por $J_n^{\mathcal{S}\alpha}$, e que todo argumento em $J_n^{\mathcal{S}\alpha}$ é justificado. Assim, para todo *r-literal* q no corpo das regras a partir das quais são gerados os argumentos que realizam *undercut*, temos que $\mathcal{DL}(\mathcal{S},\alpha) \vdash +\partial q$.

Caso 3 ($\Leftarrow$).

Prova por indução sobre o tamanho das derivações em $\mathcal{DL}(\mathcal{S},\alpha)$. Seja P uma derivação em $\mathcal{DL}(\mathcal{S},\alpha)$.

Base indutiva. Seja $P(1) = +\partial p$. Isto significa que existe uma regra que conclui p com corpo vazio. Se a regra é estrita, um argumento estrito para p pode ser gerado a partir dela, que é sempre justificado. Se a regra for derrotável então um argumento derrotável A para p pode ser gerado. Além disso, é necessário satisfazer a condição (2.3) de $+\partial$. Isso é possível apenas se não houver regras para $\sim p$, o que implica que não há argumentos para $\sim p$, e a única forma que A pode ser derrotado é por um argumento para $\sim p$. Portanto, neste caso, $A \in J_1^{\mathcal{S}\alpha}$, e, portanto, $A \in JArgs_{\mathcal{S}\alpha}$.

Passo indutivo. Assumimos que a proposição é verdadeira para derivações com tamanho menor ou igual a n.

Seja $P(n+1) = +\partial p$. Consideramos apenas os casos diferentes daqueles vistos na base indutiva. Por definição, existe uma regra $r \in KB_{\mathcal{S}\alpha}$ com cabeça p tal que $\forall q_r \in Body(r)$, $+\partial q_r \in P(1..n)$. Pela hipótese indutiva temos que há argumentos justificados para todo q_r, o que implica que existe um argumento para p apoiado por argumentos justificados, que chamaremos de A. Além disso, $\forall s \in KB_{\mathcal{S}\alpha}$ com cabeça $\sim p$, $\exists q_s \in Body(s)$ tal que $-\partial q_s \in P(1..n)$. Pela hipótese indutiva, todos os q_s são, portanto, rejeitados por $JArgs_{\mathcal{S}\alpha}$, isto é, ou não existem argumentos para os q_s ou tais argumentos são derrotados por argumentos apoiados por $JArgs_{\mathcal{S}\alpha}$. Assim, considerando um argumento B para $\sim p$ que derrota A, só pode ser o caso de B ser *undercut* por $JArgs_{\mathcal{S}\alpha}$, uma vez que algum subargumento de B é rejeitado. Logo, todo argumento que derrota A é *undercut*, e, assim, A é aceitável n.q.d.r. a $JArgs_{\mathcal{S}\alpha}$, e, portanto, justificado.

Caso 4 ($\Rightarrow$).

Provamos o contrapositivo. Sabemos que, pelas condições para ∂ apresentadas na Seção 2.2.2.1, se $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\partial p$, logo: (1) $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\Delta p$; ou (2) $\exists r \in KB_{\mathcal{S}\alpha}$ t.q. $Head(r) = p$,

$\forall q \in Body(r)$, $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\partial q$ e $\forall s \in KB_{\mathcal{S}\alpha}$, t.q. $Head(r) = \sim p$, $\exists q \in Body(s)$ t.q. $\mathcal{DL}(\mathcal{S},\alpha) \nvdash +\partial q$.

Se (1), então, pelo Caso 2, existe um argumento estrito B para p, logo B não é rejeitado.

Se (2), então existe uma regra r para p tal que $\forall q \in Body(r)$, $\mathcal{DL}(\mathcal{S},\alpha) \nvdash -\partial q$. Assim, é possível construir argumentos para cada $q \in Body(r)$, e assim subsequentemente para cada antecedente das regras utilizadas. Assim, podemos construir um argumento C para p.

Considere um argumento D que derrota C em q. Se D satisfaz o caso (1), então é parte de um argumento estrito que não pode ser derrotado. Logo, D só pode satisfazer (2). Logo, $\forall s \in KB_{\mathcal{S}\alpha}$, t.q. $Head(r) = \sim p$, $\exists q \in Body(s)$ t.q. $\mathcal{DL}(\mathcal{S},\alpha) \nvdash +\partial q$. Pelo Caso 3 deste teorema, portanto, q não é justificado. Logo, D não é apoiado por $JArgs_{\mathcal{S}\alpha}$, e assim C não é *undercut*. Logo, C não é rejeitado.

Caso 4 ($\Leftarrow$).

Base indutiva. Seja $P(1) = -\partial p$. Isso só é possível se não houverem regras aplicáveis que concluem p. Logo não há argumentos para p em $Args_{\mathcal{S}\alpha}$, logo p é rejeitado.

Passo indutivo. Seja $P(n+1) = -\partial p$. Por definição, $\forall r \in KB_{\mathcal{S}\alpha}$ t.q. $Head(r) = p$, temos que, ou (1) $\exists q_r \in Body(r)$ tal que $-\partial q_r \in P(1..n)$, ou (2) $\exists s \in KB_{\mathcal{S}\alpha}$ t.q. $Head(s) = \sim p$ tal que $\forall q_s \in Body(s)$, $+\partial q_s \in P(1..n)$.

No caso (1), pela hipótese indutiva temos que todo argumento A para p possui algum subargumento próprio que é rejeitado. Logo, todos os argumentos para p são rejeitados, e assim também p é rejeitado.

No caso (2), pela hipótese indutiva, todo argumento para p é derrotado por um argumento apoiado por $JArgs_{\mathcal{S}\alpha}$. Assim, todo argumento para p é rejeitado, e também o é p.

$\triangle$

A.3 TEOREMA 4 – TERMINAÇÃO DO ALGORITMO

Teorema 4. O algoritmo termina em tempo finito retornando um dos valores $true$, $false$ ou $undec$ para o ***r-literal*** consultado p e conjuntos de argumentos para p e $\sim p$.

Prova. A cada chamada recursiva, o algoritmo faz uma chamada para $Find\_Similar\_RLiterals$. Para cada ***r-literal*** similar encontrado que não esteja no histórico $hist_p$, são feitas no máximo duas chamadas para $Local\_Ans$ (para p' e $\sim p'$), duas chamadas para $Find\_Def\_Args$ (para p' e $\sim p'$) e uma chamada para $Compare\_Def\_Args$.

$Find\_Similar\_RLiterals$ realiza uma pesquisa simples passando pela cabeça de cada

regra em $KB_{a\alpha}$, que é garantidamente um conjunto finito, induzindo assim um número finito de operações. $Local\_Ans$ é um algoritmo de busca sobre as regras estritas em KB_a, que por pressuposto não contém cadeias de regras cíclicas. Logo, $Local\_Ans$ sempre termina.

$Find\_Def\_Args$ busca respostas para todos os *r-literais* nos corpos de todas as regras com cabeça p' (ou $\sim p'$) de um agente, mas, diferentemente de $Local\_Ans$, não está confinado às regras estritas da BC de um agente, e, considerando a possibilidade de que possam haver ciclos na base de conhecimento global estendida $KB_{\mathcal{S}\alpha}$, não é trivial provar seu término. O número de regras com cabeça p' em $KB_{\mathcal{S}\alpha}$ é finito, assim como o número de *r-literais* em seus corpos. Portanto, uma chamada para $Find\_Def\_Args$ induz a um número finito de chamadas para $Query\_Agents$ para cada *r-literal* do corpo de cada regra com cabeça p'. Por definição, um DDRMAS possui um conjunto finito de agentes. Portanto, $Query\_Agents$ recebe um conjunto finito de agentes, enviando assim um conjunto finito de mensagens $Query$. Cada mensagem $Query$ enviada na função $Query\_Agents$ pode ser vista como chamadas recursivas para $Query$. Pressupõe-se também que tenha sido estabelecido um valor finito de tempo que um agente aguarda pela resposta, após o qual o algoritmo prossegue caso a resposta não seja retornada.

Considerando a possibilidade de ciclos na base de conhecimento global, as chamadas recursivas a $Query$ levam à possibilidade de se encontrar outro *r-literal* no corpo de outra regra, possivelmente definida por outro agente, que equivale a um *r-literal* já avaliado em chamadas anteriores a $Query$. Isso poderia levar a um laço infinito no qual chamadas infinitas para $Query$ para o mesmo *r-literal* são emitidas. Isso é evitado por meio do histórico $hist_p$, que é passado em cada consulta. Em cada nova consulta, a função $Find\_Def\_Args$ é chamada apenas para *r-literais* similares que não estão em $hist_p$ (ver linhas 5 a 7 do Algoritmo 1), evitando assim chamadas recursivas para *r-literais* repetidos. Além disso, o histórico é aumentado com cada novo *r-literal* p' a ser consultado em cada chamada para $Find\_Def\_Args$ (linha 3 de $Find\_Def\_Args$). Portanto, nenhuma chamada para $Find\_Def\_Args$ pode induzir chamadas recursivas infinitas para $Query$.

$Query$ também inclui uma chamada para a função $Compare\_Def\_Args$, que recebe dois conjuntos de argumentos, realiza a comparação e marcação dos argumentos, e retorna um valor-verdade. Visto que todo argumento é uma estrutura de dados finita, $Compare\_Def\_Args$ resulta em um conjunto finito de operações.

$\triangle$

A.4 TEOREMAS 5 E 6 – CORRETUDE E COMPLETUDE DO ALGORITMO EM RELAÇÃO AO *FRAMEWORK* DE ARGUMENTAÇÃO

Teorema 5. Para um SMA $\mathcal{S} = (Ags, F_Q, \Theta, st)$, um ***r-literal*** p, e um agente $a \in Ags$, tal que existe $p' \in V^R_{a\alpha}$ similar o suficiente a p, $Local\_Ans(p')$ retorna

1. $true$ se e somente se existe um argumento estrito $A \in Args_a$ para p'
2. $false$ se e somente não existe um argumento estrito $A \in Args_a$ para p'

Prova:

Caso 1 ($\Rightarrow$).

Usamos indução no número de chamadas de $Local\_Ans$ que são necessárias para produzir a resposta para p'.

Base indutiva. Suponha que $Local\_Ans$ retorne $true$ em uma única chamada. Isto significa que existe uma regra estrita local com cabeça p', r, tal que $Body(r) = \emptyset$. Usando r, podemos construir um argumento base e estrito A para p'.

Passo indutivo. Suponha que $n + 1$ chamadas a $Local\_Ans$ sejam necessárias para calcular $true$. Isso significa que há uma regra local estrita com cabeça p' (digamos r) tal que $\forall q \in Body(r)$, $Local\_Ans$ retorna $true$ em n ou menos chamadas. Pela hipótese indutiva, para cada q há um argumento local estrito para q . Usando os argumentos para q e a regra r, podemos construir um argumento local estrito para p'.

Caso 1 ($\Leftarrow$).

Usamos indução sobre a altura de argumentos locais estritos para p' .

Base indutiva. Suponha que haja um argumento local estrito para p' (digamos A) com altura 1. Isso significa que existe uma regra local estrita com cabeça p' e com corpo vazio em a; portanto, $Local\_Ans$ retorna $true$.

Passo indutivo. Suponha que A seja um argumento local estrito para p' com altura $n + 1$. Então, há uma regra estrita com cabeça p' (r) em a, de modo que para cada ***r-literal*** q em seu corpo há um argumento estrito com altura $\leq n$. Pela hipótese indutiva, $Local\_Ans$ retorna $true$ para todo $q \in Body(r)$. Consequentemente $Local\_Ans$ irá retornar $true$.

Caso 2 ($\Rightarrow$).

Pela definição de $Local\_Ans$, é trivial verificar que $Local\_Ans$ não pode retornar $true$ e $false$ ao mesmo tempo como uma resposta para um ***r-literal*** p'. Suponha que $Local\_Ans$

retorne $false$. Suponha que haja um argumento estrito para p' . Então (pelo Caso 1 da Proposição) a resposta é $true$, o que contradiz nossa hipótese original. Consequentemente, não há nenhum argumento estrito para p' .

Caso 2 ($\Leftarrow$).

Similarmente à prova para $\Rightarrow$, supomos que não haja nenhum argumento local estrito para p' . Supondo que $Local\_Ans$ retorne $true$, concluímos (pelo Caso 1 da Proposição) que existe um argumento local estrito para p' , o que contradiz nossa hipótese original.

$\triangle$

Teorema 6. Para um SMA $\mathcal{S} = (Ags, F_Q, \Theta, st)$, um *r-literal* p, um agente $a \in Ags$ e um foco de consulta $\alpha = (p,a',KB_\alpha^{\mathcal{F}}) \in F_Q$, $Query(p,\alpha,hist_p)$ retorna

1. $true$ se e somente se existe $p' \in V_{a\alpha}^R$ similar o suficiente a p que é justificado em $KB_{\mathcal{S}\alpha}$
2. $false$ se e somente, para todo $p' \in V_{a\alpha}^R$ similar o suficiente a p, p' é rejeitado em $KB_{\mathcal{S}\alpha}$
3. $undec$ se e somente se não existe $p' \in V_{a\alpha}^R$ similar o suficiente a p que é justificado em $KB_{\mathcal{S}\alpha}$, mas existe $p' \in V_{a\alpha}^R$ que não é nem justificado nem rejeitado em $KB_{\mathcal{S}\alpha}$.

Prova:

Caso 1 ($\Rightarrow$).

Provamos usando indução nas chamadas de $Query$.

Base indutiva.

$Query$ retorna $true$ em uma única chamada. Isso ocorre quando (a) $Local\_Ans(p')$ retorna $true$ para algum p' similar o suficiente a p. Logo, pela Proposição 5, há um argumento local estrito A para um p' similar o suficiente a p. Portanto, $A \in JArgs_{\mathcal{S}\alpha}$ e assim p' é justificado; ou (b) existe uma regra local derrotável r em a tal que $Body(r) = \emptyset$ e $Head(r) = p'$ e não há regra com cabeça $\sim p'$ em a. Portanto, há um argumento A para p' formado a partir de r e não há nenhum argumento que ataca A. Como A não tem subargumentos próprios e não é atacado por nenhum argumento, $A \in JArgs_{\mathcal{S}\alpha}$; portanto, p' é justificado.

Passo indutivo.

$Query$ retorna $true$ em $n + 1$ chamadas. As seguintes condições devem ser válidas:

(a) existe uma regra r com cabeça p' em a, de modo que, para todos os *r-literais* em seu corpo ($q \in Body(r)$), a resposta $true$ é retornada por $Query$ em no máximo n chamadas. Pela hipótese indutiva, para todo q há um argumento A_q com conclusão q em $JArgs_{\mathcal{S}\alpha}$. Portanto, para cada A_q, então A_q ou é um argumento base ou todos os seus subargumentos estão em $JArgs_{\mathcal{S}\alpha}$.

Usando o argumento A_q e a regra r, podemos construir um argumento A para p' de modo que todo subargumento próprio de A esteja em $JArgs_{\mathcal{S}\alpha}$ – em outras palavras, A é apoiado por $JArgs_{\mathcal{S}\alpha}$.

(b) $Local\_Ans(\sim p')$ retorna $false$ – pela Proposição 5, não existem argumentos estritos para $\sim p'$.

(c) para toda regra s com cabeça $\sim p'$ em a: ou (c.1) existe um *r-literal* q' no corpo de s para o qual $Query$ retorna $false$ em n chamadas. Pela hipótese indutiva, q' é rejeitado, o que significa que os argumentos para q' ou seus subargumentos são derrotados por um argumento apoiado por $JArgs_{\mathcal{S}\alpha}$. Consequentemente, todo argumento B baseado na regra s para $\sim p'$ é *undercut* por $JArgs_{\mathcal{S}\alpha}$; ou (c.2) $\forall q' \in Body(s)$, $Query$ retorna $true$ ou $undec$ como uma resposta para q' (em no máximo n chamadas) juntamente com um conjunto de argumentos para q', a partir dos quais é possível construir argumentos para $\sim p'$. Neste caso, então, deve existir pelo menos um argumento para p' que não é derrotado por nenhum desses argumentos para $\sim p'$, e assim, pela definição, p' é justificado.

Caso 1 ($\Leftarrow$).

Usamos indução no estágio de aceitabilidade dos argumentos com conclusão p' em $Args_{\mathcal{S}\alpha}$.

Base indutiva. Suponhamos que um argumento A para p' em $Args_{\mathcal{S}\alpha}$ é aceitável n.q.d.r. a $J_0^{\mathcal{S}\alpha}$. Isto significa que: ou (a) A é um argumento estrito para p', caso em que, pela Proposição 5, será retornado $true$ por meio de $Local\_Ans$; ou (b) A é um argumento derrotável baseado em uma regra r tal que $Body(r) = \emptyset$, o que implica que a função $Find\_Def\_Args(p',...)$ retorna um conjunto de argumentos $Args_{p'}$ contendo o argumento A, e: ou (b.1) nenhum argumento para $\sim p'$ pode ser gerado, de modo que nenhum argumento derrote A. Logo, $Find\_Def\_Args(\sim p',...)$ retorna um conjunto vazio de argumentos $Args_{\sim p'}$. Assim, $Query$ retornará $true$. Ou (b.2), para todo argumento B para $\sim p'$ que pode ser gerado, B não derrota A, ou B derrota A mas B já foi marcado como rejeitado. Logo, $Find\_Def\_Args(\sim p',...)$ retorna um conjunto de argumentos não vazio $Args_{\sim p'}$ para $\sim p'$, tal que nenhum argumento não rejeitado em $Args_{\sim p'}$ derrota A. Assim, $Query$ retornará $true$.

Passo indutivo. Suponha que um argumento A para p' em $Args_{\mathcal{S}\alpha}$ é aceitável n.q.d.r. a $J_{n+1}^{\mathcal{S}\alpha}$. Isto significa que: ou (a) A é um argumento estrito para p', caso em que, pelo Teorema 5, será retornado $true$ por meio de $Local\_Ans$; ou (b) A é apoiado por $J_{n+1}^{\mathcal{S}\alpha}$ e todo argumento que derrota A é *undercut* por $J_{n+1}^{\mathcal{S}\alpha}$. O fato de A ser apoiado por $J_{n+1}^{\mathcal{S}\alpha}$ significa que todo

subargumento próprio de A é aceitável n.q.d.r. a $J_n^{\mathcal{S}\alpha}$. Pela hipótese indutiva, existe uma regra r com cabeça p' tal que $\forall q \in Body(r)$, $Query(q,...)$ retorna $true$. Logo, na chamada de $Query(p,...)$, $Find\_Def\_Args(p',...)$ retorna um conjunto de argumentos $Args_{p'}$ tal que $A \in Args_{p'}$.

Suponha que exista um argumento B que derrote A. Logo, pela definição, B é *undercut* por $J_{n+1}^{\mathcal{S}\alpha}$, isto é, existe um argumento C para um *r-literal* q que é apoiado por $J_{n+1}^{\mathcal{S}\alpha}$ e derrota algum subargumento próprio de B. Uma vez que C é apoiado por $J_{n+1}^{\mathcal{S}\alpha}$, todo subargumento de C é aceitável n.q.d.r. a $J_n^{\mathcal{S}\alpha}$. Suponhamos que C derrote B' em q, onde B' é um subargumento próprio de B. Pela hipótese indutiva, então, existe uma regra t para q tal que $\forall q' \in Body(t)$, $Query(q',...)$ retorna $true$. Isto implica que a chamada para $Find\_Def\_Args(\sim p',...)$, na consulta inicial, induz a uma chamada $Query(q,...)$, na qual $Find\_Def\_Args(q,...)$ retorna um conjunto de argumentos não vazio $Args_q$ (do qual B' faz parte) e $Find\_Def\_Args(\sim q,...)$ retorna um conjunto de argumentos não vazio $Args_{\sim q}$ (da qual C faz parte). Uma vez que supomos que C derrota B', temos que $Query(q,...)$ retorna $false$, o que faz com que B' seja marcado como rejeitado, de modo que B também será marcado como rejeitado. Assim, não restando argumentos não rejeitados que derrotem A, a consulta inicial ($Query(p,...)$) retorna $true$.

Caso 2 ($\Rightarrow$).

Base indutiva. $Query$ retorna $false$ em uma única chamada. Isso ocorre quando, para todo p' similar o suficiente a p, $Local\_Ans(p')$ retorna $false$ (pelo Teorema 5), i.e., não há nenhum argumento estrito para nenhum p', e (a) $Local\_Ans(\sim p')$ retorna $true$, isto é, há um argumento estrito B para $\sim p'$, que, por definição, é justificado e derrota qualquer argumento não estrito para p' e não é *undercut*. Logo, pelas linhas 19 e 20 de $Query\_Agents$, todo argumento para p' é rejeitado; ou (b) existe uma regra derrotável local s com cabeça $\sim p'$ tal que $Body(s) = \emptyset$. Portanto, existe um argumento B para $\sim p'$. B não tem subargumentos próprios – portanto, é apoiado e não *undercut* por $JArgs_{\mathcal{S}\alpha}$ – e $StArg(B) = 1$ – portanto, B derrota qualquer argumento não estrito para p'. Como não há nenhum argumento local estrito para p', todo argumento para p' é derrotado por B; portanto, p' é rejeitado.

Passo indutivo. $Query$ retorna $false$ em $n+1$ chamadas. As duas condições a seguir devem ser satisfeitas: (a) $Local\_Ans(p')$ retorna $false$; logo, não existe nenhum argumento estrito para p'; e (b) para toda regra r com cabeça p', ou (b.1) existe um *r-literal* $q \in Body(r)$ de modo que $Query$ retorna $false$ em no máximo n chamadas. Pela hipótese indutiva, isso

significa que q é rejeitado e, portanto, todo argumento A baseado em r contém um subargumento que é derrotado por um argumento apoiado por $JArgs_{\mathcal{S}\alpha}$, de modo que os argumentos baseados em r são rejeitados; ou (b.2) existe uma regra s para $\sim p'$ tal que $Query$ retorna $true$ para todo *r-literal* $q' \in Body(s)$, e tal que para todo *r-literal* $q \in Body(r)$, $Query$ retorna $true$ ou $undec$ em no máximo n chamadas, e todos os argumentos baseados em r são derrotados pelo argumento baseado em s. Portanto, para todo argumento A para p', segue que, ou A ou algum subargumento de A são derrotados por um argumento apoiado por $JArgs_{\mathcal{S}\alpha}$. Portanto, p' é rejeitado por $JArgs_{\mathcal{S}\alpha}$.

Caso 2 ($\Leftarrow$). Usa-se prova por contradição.

Suponha que para um *r-literal* p' que é rejeitado, segue que $Query$ retorna $true$ ou $undec$ como uma resposta para p'. Por definição: ou (a) $Local\_Ans(p')$ retorna $true$, o que significa que existe um argumento estrito para p', levando à conclusão de que p' é justificado, o que, pela propriedade de coerência (Teorema 1), contradiz a suposição inicial de que p' é rejeitado por $JArgs_{\mathcal{S}\alpha}$; ou (b) existe pelo menos um argumento para p' que não possui subargumentos rejeitados e que não é derrotado por um argumento que não é *undercut*. Logo, existe pelo menos um argumento não rejeitado para p', o que contradiz a suposição inicial. Portanto, $Query$ retornará $false$ como resposta para p'.

Caso 3 ($\Rightarrow$).

Base indutiva. Pelo menos três chamadas a $Query$ são necessárias para que se retorne $undec$. As seguintes condições devem ser verdadeiras: (a) $Local\_Ans(p') = false$, o que, pela Teorema 5, implica que não há nenhum argumento estrito para p'; (b) não há regras com cabeça $\sim p'$, o que significa que não há argumentos que derrotam os argumentos para p' em sua raiz; e (c) há apenas uma regra r com cabeça p', com um *r-literal* q em seu corpo, para a qual é verdadeiro que: (c.1) não há regra com cabeça $\sim q$, e (c.2) há apenas uma regra r' com cabeça q, de modo que p' é o único *r-literal* no corpo de r'. Portanto, o único argumento para p' (A) pode ser obtido usando a regra r, e o único argumento para q (A') pode ser obtido usando a regra r', tal que A' tem um único subargumento A'' que é um nó-folha falacioso p'!. Assim , nenhum dos argumentos é justificado nem rejeitado, e, portanto, p' não é justificado nem rejeitado.

Passo indutivo. $Query$ retorna $undec$ em $n + 1$ chamadas. As seguintes condições devem ser satisfeitas:

(a) $Local\_Ans(p')$ e $Local\_Ans(\sim p')$ ambos retornam $false$. Pelo Teorema 5, não há argumentos estritos para p' e $\sim p'$ em $Args_{\mathcal{S}\alpha}$;

(b) para toda regra r com cabeça p': ou (b.1) há um *r-literal* q no corpo de r de modo que $Query$ retorna $false$ ou $undec$ em n ou menos chamadas; ou (b.2) para todo q no corpo de r, $Query$ retorna $true$ como uma resposta para q em n ou menos chamadas, mas há uma regra s com cabeça $\sim p'$, de modo que, para cada *r-literal*, q' no corpo de s, $Query$ retorna $true$ ou $undec$ como uma resposta para q' em n ou menos chamadas e todos os argumentos para p' são derrotados pelos argumentos para $\sim p'$. Para o caso descrito em (b.1), pela hipótese indutiva, q não é justificado; portanto, não existe argumento para p' que seja apoiado por $JArgs_{\mathcal{S}\alpha}$. Para o caso de (b.2), pela hipótese indutiva, existe um argumento B baseado em s para $\sim p'$, tal que, para todo argumento A baseado em r para p', B derrota A. Podemos então provar que existe um argumento B', que também é baseado na regra s, e que é mais forte que B, de forma que nenhum subargumento de B seja derrotado por um argumento apoiado por $JArgs_{\mathcal{S}\alpha}$. Assim, B' não é *undercut* por $JArgs_{\mathcal{S}\alpha}$ e derrota todo argumento derrotável baseado em regras com cabeça p'. Uma vez que não há argumento estrito para p', então nenhum argumento para p' é apoiado por $JArgs_{\mathcal{S}\alpha}$. Assim, chega-se à conclusão de que para todo argumento A para p', A não é apoiado por $JArgs_{\mathcal{S}\alpha}$ ou é derrotado por um argumento em $Args_{\mathcal{S}\alpha}$ que não é *undercut* por $JArgs_{\mathcal{S}\alpha}$; e, portanto, p' não é justificado.

(c) para toda regra s com cabeça $\sim p'$: ou (c.1) existe um *r-literal* q' no corpo de s, de modo que $Query$ retorna $false$ ou $undec$ como resposta para q' em n ou menos chamadas; ou (c.2) para todo q', $Query$ retorna $true$ como resposta para q' em n ou menos chamadas, mas existe uma regra r com cabeça p', de modo que, para cada *r-literal* q no corpo de r, $Query$ retorna $true$ ou $undec$ em n ou menos chamadas, e, para algum argumento baseado em r com conclusão p', tal argumento não é derrotado por nenhum argumento para $\sim p'$. Da mesma forma que antes, podemos chegar à conclusão de que existe um argumento A baseado em uma regra r, de modo que não existe argumento B que seja apoiado por $JArgs_{\mathcal{S}\alpha}$ e derrote A ou um subargumento de A. Portanto, p' não é rejeitado por $JArgs_{\mathcal{S}\alpha}$.

Caso 3 ($\Leftarrow$). Isso é trivial de se provar usando os casos 1 e 2 do teorema. Para um *r-literal* p' que não é justificado nem rejeitado, suponha que $Query$ retorna $true$. Pelo Caso 1, tem-se que p' é justificado (contradição). Suponha então que $Query$ retorna $false$. Pelo Caso 2, isso significa que p' é rejeitado (contradição). Portanto, para p', $Query$ só pode retornar $undec$.

$\triangle$