Tiago Luiz Schmitz

Modelo Anímico para Raciocínio Normativo Organizacional de um Agente BDI

Florianópolis, Brasil
8 de março de 2016

Tiago Luiz Schmitz

Modelo Anímico para Raciocínio Normativo Organizacional de um Agente BDI

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Doutor.

Universidade Federal de Santa Catarina - UFSC

Orientador: Jomi F. Hübner

Florianópolis, Brasil
8 de março de 2016

,
Modelo Anímico para Raciocínio Normativo Organizacional de um Agente BDI : / Tiago Luiz Schmitz; orientador, Jomi F. Hübner. - Florianópolis, Brasil 2016.
133 p.

- Universidade Federal de Santa Catarina - UFSC, Centro Tecnológico - CTC. Programa de Pós - Graduação em Engenharia de Automação e Sistemas - PPGEAS.

Inclui Referências

1.Normas. 2.Desejos. 3.racíocinio. 4.multiagentes. I. , . II. Universidade Federal de Santa Catarina - UFSC. Programa de Pós - Graduação em Engenharia de Automação e Sistemas - PPGEAS. II. Modelo Anímico para Raciocínio Normativo Organizacional de um Agente BDI.

Esta Tese foi julgada adequada como **TESE DE DOUTORADO** no Curso de Doutorado em Engenharia de Automação e Sistemas e aprovada em sua forma final pela Banca Examinadora designada.

Dr. Rômulo Silva de Oliveira
Coordenador do PPGEAS

Banca Examinadora:

Dr. Jomi Fred Hübner
Orientador

Dr. Rafael H. Bordini

Dr. Antônio Carlos da Rocha Costa

Dra. Jerusa Marchi

Dr. Eduardo Camponogara

Dr. Marcelo Ricardo Stemmer

Para meus pais e irmã,
Luiz, Lisana e Luíza.

AGRADECIMENTOS

Era dezembro de 1999 e aquele guri novo tinha o sonho de ser doutor. Na sua mente, ele tinha sempre os mesmos versos transpassando seus pensamentos: "... nem tão perto que eu possa tocar // nem tão longe que eu não possa crer que um dia chego lá... ". Quinze anos e alguns meses depois estes versos se modificaram na mente deste guri já não tão jovem: "... nem tão perto que eu possa acreditar // que o dia já chegou...".

Com a certeza de que o dia chegou, como a aurora de um dia traz luz à sombra da noite, percebo nessa viagem que o início e o fim são importantes, mas que na viagem foi onde toda a magia aconteceu. Viajar sozinho não tem graça e ninguém consegue seguir rumo sem uma mão amiga ajudando a guiar o barco nas noites sem estrelas. Este é o fim da jornada e muitas mãos foram necessárias para manter o rumo.

Primeiramente, agradeço a minha família, em especial a meus pais e irmã que me apoiaram de todas as maneiras possíveis. Neste ponto cabe dizer que a saudade é um veneno traiçoeiro e poderoso que nos drena a energia e só se manifesta quando não há antídoto disponível.

Quando perdido no mar das ideias a mão experiente de um navegante que já singrou por essas águas é de grande valia ao aventureiro de primeira viagem. Muito injusto seria se não deixasse nestas poucas linhas um agradecimento a todos que foram meus professores. Em especial, ao Jomi que acreditou em mim e de forma brilhante e dedicada me guiou nesta jornada.

Se eu digo que viajar sozinho é chato é porque viajar com amigos é muito bom. São eles que te fazem rir, oferecem um ombro amigo quando precisa chorar e aconselham nos momentos de dúvida, ou seja, fazem o papel da família quando esta não pode estar presente. Em especial, agradeço aos amigos que estão junto comigo na nau do mar das ideias, André, Andreu, Eduardo, Lie, Renan, Ríad e Toscano.

Durante essa viagem muita coisa aconteceu. Houve obstáculos que forçaram a busca por outros caminhos não planejados, mas como em toda boa viagem na hora mais sombria sempre surge um anjo que

serena a mente e acalenta o coração. Nessa viagem não foi diferente. Pouco antes do grande obstáculo, eu conheci o amor. Hoje olhando os rumos tomados tenho certeza que a providência divina a colocou ao meu lado. A viagem com ela se tornou diferente, nas noites escuras e frias o brilho dela me iluminava e aquecia, revigorando as minhas forças para continuar. Agradeço a ti, Daniela, pelo carinho, amor e apoio.

Em qualquer história de aventura existe um termo de fé, esta não é diferente. Agradeço a nossa Senhora de Caravaggio pela força nos momentos em que fraquejei. Nestas linhas rôtas que contam um pouco da minha história, personagens vieram e foram tornando-a única. A todos que participaram os meus sinceros agradecimentos, vocês tornaram a viagem maravilhosa.

Não posso esquecer de agradecer a quem financiou boa parte dessa viagem. Minha gratidão ao CNPq que forneceu recursos para que eu pudesse viajar tão longe e conquistar o meu sonho.

O que sabemos é uma gota, o que ignoramos é um oceano.
(Isaac Newton)

RESUMO

Em sistemas multiagentes abertos dotados de sistemas normativos é necessário que os agentes sejam capazes de deliberar sobre as normas do sistema e seus desejos pessoais. Considerando que os agentes normativos têm recursos limitados, é necessário raciocinar também sobre os recursos disponíveis e se eles são suficientes para alcançar o objetivo implicado por uma norma. Nesta direção, o objetivo principal da tese é desenvolver uma arquitetura e um processo de raciocínio que juntos são capazes de deliberar sobre normas e desejos, levando em consideração os recursos finitos.

Para atingir esse objetivo, o conceito de ânimo (THAYER, 1989) foi tomado como ponto de partida para propor uma arquitetura capaz de representar as informações necessárias para o agente deliberar sobre as normas e desejos. O processo de deliberação proposto mapeia essas informações em um problema de programação inteira mista e através de um solver obtém o conjunto de desejos e normas, que produz o melhor benefício ao agente. Como cenário de aplicação dos conceitos estabelecidos nessa tese, foi desenvolvido o planejamento de trajetória on-line para um veículo aéreo não tripulado. Conferindo a este autonomia para deliberar sobre as rotas a serem percorridas.

Os resultados obtidos nessa tese são: uma arquitetura de agente capaz de raciocinar sobre normas e desejos considerando recursos limitados e uma implementação da arquitetura proposta.

Palavras-chaves: Sistemas Multiagentes. Normas. Raciocínio.

SUMÁRIO

1 INTRODUÇÃO

Sistemas multiagente abertos (SMAa) são uma classe especial de sistemas multiagente caracterizada pela variação populacional dos agentes, uma vez que esses podem entrar e sair do sistema a qualquer momento. Agentes necessitam coordenar-se e cooperar entre si para cumprir as metas do sistema. Contudo, a variação populacional oriunda dos SMAa pode adicionar um custo elevado à essas ações fazendo com que exista uma perturbação no cumprimento dos objetivos do sistema.

Um sistema multiagente aberto pode ser visto como uma sociedade artificial (FERBER; GUTKNECHT; MICHEL, 2004) e, como tal, sua coordenação pode ser mediada por uma organização. Essas sociedades artificiais são compostas por grupos com propósitos específicos, possuindo diferentes organizações e, por conseguinte, diferentes sistemas normativos.

Um sistema normativo pode ser definido como um conjunto de normas impostas aos agentes com finalidade de conduzir seu comportamento para atingir os objetivos do grupo (BOELLA; TORRE; VERHAGEN, 2006). A capacidade de interpretar sistemas normativos são características desejáveis aos agentes que pertencem aos grupos. Para que um agente tenha essas características, é necessário que esse seja capaz de raciocinar sobre as normas. Nessa direção, Foucault define o raciocínio como uma das maneiras de se **obter** crenças ou comportamentos considerando as explanações e justificativas acerca de um evento, fenômeno ou comportamento. O raciocínio humano está intimamente relacionado com a habilidade da autoconsciência do ser em **modificar crenças, comportamentos**, tradições e instituições (FOUCAULT, 2003). A alteração do comportamento de um indivíduo dentro de uma sociedade não é trivial. Whisner declara que a mudança de comportamento é um processo complexo, pois o indivíduo necessita fazer ponderações entre valores individuais e sociais (WHISNER, 2003). O raciocínio de um agente possui uma semelhança com essa definição, pois são autônomos por definição e, portanto, é desejável que os agentes sejam capazes de compreender e utilizar as normas da organização

para ajustar o seu comportamento à sociedade na qual estão inseridos. Para tal, o agente deve ponderar sobre as normas e desejos pessoais.

Observa-se que os agentes dotados de raciocínio adquirem crenças e modificam seu comportamento ao longo de sua existência, tendo que ponderar sobre atribuições individuais e sociais. O raciocínio sobre as atribuições individuais é um assunto bastante pesquisado, como no caso do raciocínio prático do BDI (*Belief - Desire - Intention*) (BRATMAN, 1999). O raciocínio sobre as atribuições sociais do ponto de vista computacional é recente, começando a ser explorado nas últimas duas décadas. Em Torre e Tan (1999), o raciocínio normativo é definido como o processo de pensar sobre as normas para tomar decisões.

Agentes com capacidade de raciocínio enfrentam situações de conflito e a mais explorada na bibliografia é o **conflito entre os desejos do agente e as normas da organização**. Agentes com essa característica enfrentam situações, nas quais os objetivos do agente podem conflitar com os objetivos da sociedade. Por exemplo, o agente A assume o papel X na sociedade I, que tem como objetivo atingir o estado P, ao mesmo tempo que tem um desejo de não atingir P.

Contudo, se considerarmos que os agentes possuem recursos finitos, um novo tipo de conflito pode surgir: um **conflito por uso de recursos**, que até então não foi muito considerado na literatura. Ao agente, ter recursos finitos implica que ele tem de fazer a gerência dos mesmos para que eles sejam consumidos produzindo benefício ao agente. O fato de a percepção de novos objetivos e normas ser imprevisível, dificulta o gerenciamento dos recursos. Por exemplo, um agente se compromete a cumprir a norma N_1, o que consumirá 10 unidades do recurso r. Na metade do cumprimento de N_1, uma norma N_2 é assumida e consumirá 3 unidades de r. Contudo, o agente possui apenas 11 unidades do recurso. Com as duas normas sendo cumpridas em paralelo, poderá ocorrer o não cumprimento de nenhuma (N_1 consumiu 9 e N_2 consumiu 2), ou o cumprimento de apenas uma. Nesse caso, o agente deve determinar qual norma irá melhor beneficiá-lo e se comprometer apenas com ela.

Em conjuntos de normas e objetivos maiores que no exemplo

apresentado, o problema ganha complexidade, sendo necessário um raciocínio capaz de determinar o melhor conjunto de normas e objetivos que, uma vez cumpridos, possam maximizar o benefício do agente, respeitando os recursos disponíveis e solucionando as situações de conflito direto entre desejos e normas. Esse problema será abordado neste trabalho.

A partir de uma busca por estudos relacionados ao raciocínio normativo nos principais eventos da área de sistemas multiagente (COIN, BRACIS, AAMAS) e em bases digitais (IEEEXplorer, Springer), encontrou - se trabalhos como Alechina, Dastani e Logan (2012), Carabelea, Boissier e Castelfranchi (2005), Criado, Argente e Botti (2011), Meneguzzi e Luck (2009), Boella e Torre (2004a), Broersen e Torre (2007), Joseph et al. (2010), Gaertner e Toni (2008), Andrighetto et al. (2007), Sadri, Stathis e Toni (2006), Kollingbaum e Norman (2006) que podem fornecer indícios de como resolver esses conflitos. Esses trabalhos utilizam técnicas de alocação de tarefas, poder social, implementações de modelos de raciocínio oriundos da sociologia, processo argumentativo e outras encontradas na mesma literatura. Por outro lado, nenhum desses trabalhos utiliza, de forma explícita, os recursos finitos de um agente como critério para realizar esse tipo de raciocínio.

1.1 OBJETIVOS

O objetivo principal desta tese é desenvolver uma arquitetura e um processo de raciocínio que, juntos, são capazes de deliberar sobre normas, levando em consideração os recursos do agentes. Para atingir esse objetivo, devem ser atingidos os seguintes objetivos específicos:

1. propor e validar uma arquitetura de agente que permita aos agentes representarem desejos, normas e recursos:
    a) desenvolver uma arquitetura capaz de representar as informações necessárias para o raciocínio normativo;
    b) essa arquitetura deve ser flexível e permitir a representação de diferentes critérios de raciocínio;

2. propor um processo de deliberação que permita aos agentes raciocinar sobre objetivos e normas. Para tal, é necessário propor e validar mecanismos que permitam ao agente:

    a) raciocinar sobre o cumprimento e a relevância das normas;

    b) resolver conflitos diretos entre desejos e normas;

    c) resolver conflitos de recursos;

    d) revisar o cumprimento e relevância das normas.

1.2 METODOLOGIA

A primeira etapa para atingir a meta principal desta tese é o estudo acerca de conceitos psicológicos e econômicos, em busca de inspiração para modelar as estruturas dos agentes. Essa abordagem é adotada devido à visão do SMA como uma sociedade artificial, na qual os agentes são indivíduos dessa sociedade. Nessa etapa, também será observado como os trabalhos relacionados ao raciocínio apresentam tais estruturas. As estruturas definirão como o agente representará internamente as normas e como as informações transitarão pela mente do agente, assim, cumprindo o objetivo específico 1a. Com a intenção de permitir a expansão do modelo de raciocínio a outros elementos, será adotada uma estrutura modular que permite cumprir o objetivo específico 1b. O cumprimento dessa primeira etapa tem como resultado uma arquitetura que permite aos agentes representarem objetivos e normas. A validação dessa etapa será feita através de uma avaliação experimental, na qual a arquitetura será incorporada em uma linguagem de programação de agentes e através de cenário de testes, analisado a facilidade de representação das informações e a capacidade de expansão do modelo.

Com as estruturas estabelecidas, será usado uma modelagem *MIP - Mixed integer programming* para cumprir os objetivos 2a, 2b e 2c. Nessa fase é elaborado um cenário de testes para verificar qual o impacto dessa modelagem no tempo de resposta do agente. Para atingir o objetivo 2d, é necessário elaborar técnicas que permitam ao

agente continuar ou abandonar o cumprimento de normas, levando em consideração as consequências dessas ações.

1.3 ORGANIZAÇÃO DO DOCUMENTO

O restante deste documento está dividido em quatro capítulos. No segundo capítulo é feita uma revisão bibliográfica para definir o significado dos conceitos adjacentes ao raciocínio normativo. O terceiro capítulo apresenta a classificação do estado da arte do raciocínio normativo, apresentado os benefícios e limitações dos trabalhos relacionados. O quarto capítulo propõe o modelo de raciocínio normativo baseado em ânimo, sua evolução durante essa pesquisa e uma comparação com o arcabouço teórico da literatura. Por fim, este documento apresenta um cenário de testes que exemplifica o uso da teoria e ferramenta desenvolvidas nesta tese.

2 FUNDAMENTAÇÃO TEÓRICA

> "A lei é poderosa; mais poderosa, porém, é a necessidade."
>
> Johann Wolfgang von Goethe

Neste capítulo são apresentados os conceitos adjacentes ao raciocínio normativo. Primeiramente, é apresentada uma definição de sistema normativo. A seguir, uma classificação dos tipos de agente. Por fim, uma seção apresentando brevemente os conceitos de ânimo utilizado como subsídio no raciocínio normativo.

2.1 SISTEMA NORMATIVO

Nesta seção, será apresentado o conceito de sistema normativo utilizado no presente trabalho. O sistema normativo é responsável por avaliar se as obrigações foram completamente atingidas ou se proibições foram descumpridas. Ainda é responsabilidade do sistema normativo observar as violações de *deadlines* das obrigações e proibições e impor as correspondente sanções (BOELLA; TORRE, 2004b). Dessa maneira, o trabalho da organização é continuamente:

- monitorar o comportamento dos agentes;
- avaliar o efeito de cada agente no ambiente;
- aplicar sanções quando as mesmas forem violadas.

Normas são mecanismos de coordenação com o objetivo de reforçar comportamentos adequados e dissuadir comportamentos inadequados dentro de uma organização. Na sociologia, existem muitos trabalhos (RAWLS, 1955; SEARLE, 1969; SEARLE, 1997) que descrevem diferentes tipos de normas em um mesmo sistema normativo. Especificamente, existem dois tipos principais de normas: constitutivas e deônticas. Nesta tese serão abordadas apenas as normas deônticas.

Normas deônticas são expressas em termos de obrigações, permissões e proibições. As normas constitutivas definem a realidade institucional (fatos institucionais), construída em termos de ações ou estado das coisas do mundo real (fatos brutos). A especificação normativa (N) de um sistema multiagente pode ser definida como:

$$N = N_{deontica} \cup N_{constitutiva}$$

onde:

- $N_{deontica}$ é o conjunto de normas deônticas que define o que é considerado proibido, obrigatório ou permitido.
- $N_{constitutiva}$ é o conjunto de normas constitutivas que define a ligação entre o mundo real e a realidade institucional.

2.1.1 Normas deônticas

As normas deônticas representam a responsabilidade, direitos e deveres dos agentes. Elas são divididas em três modalidades: a obrigação, a proibição e a permissão. O comportamento de cada uma dessas modalidades segue um padrão definido. Obrigações são estados globais que devem ser atingidos por um agente. Proibições são estados globais que não devem ser atingidos. Permissões são estados globais que podem ser atingidos.

Normas geralmente não são aplicadas todo o tempo, logo sua especificação tem condições de ativação e expiração. A condição de ativação de uma norma define quando ela será instanciada, dessa forma, o agente terá a ciência de que, a partir dessa condição, ele deve cumpri-la. A condição de expiração define o período de validade da norma.

Outro elemento componente da norma é o mecanismo de reforço. Nesse mecanismo a norma ganha dois componentes: a sanção e a recompensa. A sanção corresponde à punição que o agente recebe por não cumprir a norma. A recompensa é o reforço positivo recebido pelo agente por ter cumprido a norma. As normas deônticas utilizadas nesse trabalho são representadas pela tupla n_d (PACHECO, 2012):

$$\langle D, C, T, A, E, S, R \rangle$$

Onde:

- $D \in \{O, F, P\}$ é a modalidade deôntica da norma, obrigação (O), proibição (F) e permissão (P);
- C é o estado controlado pela norma;
- T é o agente alvo da norma;
- A é a condição de ativação da norma;
- E é a condição de expiração da norma;
- S é a sanção imposta caso o agente descumpra a norma;
- R é a recompensa recebida caso a norma seja cumprida.

Por exemplo, a norma abaixo obriga o motorista conduzir o veículo devagar quando chove. O agente alvo dessa norma é o motorista. A norma é ativada quando está chovendo e cessa quando não está chovendo. O motorista é sancionado com uma multa (sanção = 100) caso não a cumpra. Caso o motorista obedeça a norma ele não recebe nenhum benefício (recompensa = 0).

$$\langle O, devagar, motorista, chuva, \neg chuva, 100, 0 \rangle$$

2.2 AGENTES

Para compreender as características e o comportamento dos agentes normativos, primeiramente precisamos estudar as propriedades e os problemas ligados aos agentes em geral. De início, é apresentada uma definição de agente que se enquadra no escopo do trabalho para, posteriormente, definir os tipos de agente.

O termo agente é ainda controverso na sua definição (FIPA. . . , 2013). Todas as tentativas de unificar diferentes teorias sobre os agentes ou de encontrar um denominador comum acabaram por falhar (WOOLDRIDGE; JENNINGS, 1995). Muitas definições (WOOLDRIDGE;

JENNINGS, 1995; SHOHAM, 1993) são proposta sobre o domínio dos SMA, assim como Ferber, para o qual o agente é uma entidade física ou virtual (FERBER, 1995):

- capaz de agir sobre ele mesmo e sobre o ambiente;
- capaz de perceber seu ambiente (com uma representação parcial);
- capaz de comunicar-se com outros agentes;
- que possui objetivos individuais;
- que possui experiência e eventualmente pode reproduzi-la;
- no qual o comportamento é consequência de todas as propriedades acima citadas.

A autonomia é um dos conceitos principais na definição do comportamento do agente. Ela foi inicialmente utilizada na robótica para descrever a capacidade do robô de reagir de maneiras adequadas aos eventos inesperados que podem aparecer no ambiente físico real. Com o surgimento dos sistemas multiagentes, que estudam a interação entre os muitos agentes heterogêneos que compõem o sistemas, os quais se denominam agentes autônomos. Por exemplo, para Demazeau e Müller (1991), um agente autônomo é um agente cuja existência não se justifica pela existência de outros. Por Castelfranchi (1995), um agente autônomo se enquadra na definição de Ferber, mas ele adiciona a noção de que o agente age sem a intervenção dos humanos ou outros agentes e possui um controle sobre suas ações e seu estado interno.

Autonomia é definida como a capacidade de o agente executar uma ação por sua própria iniciativa (pró - atividade). Os agentes podem controlar seu comportamento a fim de atender os seus objetivos e decidir se ajudam ou impedem os outros de realizarem seus objetivos. A autonomia é representada em graus de liberdade, o que equivale a dizer que um agente é livre para decidir o que ele fará e como perceberá o mundo.

A Reatividade de um agente é definida como a capacidade de perceber e de reagir sobre modificações ocorridas no ambiente. O tipo de reatividade é tratado de diferentes maneiras em função da forma com que os agentes são construídos.

Sociabilidade significa que os agentes são capazes de interagir com outros agentes, ou até mesmo humanos. Esse conceito conduz os estudos dos sistemas multiagentes através da perspectiva social. Os agentes possuem uma representação não apenas do ambiente e de suas próprias características, mas também de outros agentes. A sociabilidade é uma propriedade importante a ser preservada, mas é importante ressaltar que essa propriedade limita o comportamento do agente e, por vezes, entra em contradição com seus objetivos pessoais.

A noção de agente se refere por definição a qualquer um que age. A classificação dos tipos de agentes, baseada em seu módulo de decisão, gera três classe de agentes: **cognitivos, reativos e híbridos** (WOOLDRIDGE; JENNINGS, 1995). Todos os modelos seguem uma mesma arquitetura geral com processos de observar, decidir e agir. Nesta tese, buscamos um agente capaz de raciocinar, portanto nos focaremos no tipo cognitivo.

Os agentes do tipo cognitivo seguem um paradigma proposto pela IA simbólica. Esse tipo de agente pode ser criado reunindo diversos módulos cognitivos baseados em uma representação simbólica do mundo. A execução de um agente é caracterizada por um ciclo reativo (WOOLDRIDGE; JENNINGS, 1995) do tipo **percepção - deliberação - ação**. O modulo de deliberação simbólica é capaz de, a partir desse modelo, tomar a decisão correspondente à ação que será executada a seguir.

As arquiteturas desse tipo mais comuns são as BDI (Belief - Desire - Intention) (BRATMAN, 1999). As crenças, os desejos e as intenções de um agente são as noções cognitivas primárias e caracterizam o estado do agente (BRATMAN, 1999). O modelo de um agente BDI é representado como um sistema composto de estruturas dinâmicas de dados correspondentes aos princípios cognitivos e um conjunto de eventos os quais o agente pode tratar. Os eventos são considerados

como as entradas do sistema e são produzidos por diferentes fontes de informação, podendo ser externas (modificações do ambiente) ou internas (modificações do estado do agente). As saídas do sistema são ações atômicas executadas por intermédio de uma função *execute*. A partir do seu estado corrente e da fila de eventos, o sistema seleciona e executa as opções que correspondem aos procedimentos, regras de produção, tarefas, planos ou autômatos de estados finitos. As ações, uma vez executadas, podem gerar o início de um novo ciclo, por produzirem novos eventos. A execução de um agente é descrita por um interpretador contendo uma execução em laço apresentado no algoritmo 1. Uma parte dos sistemas que implementam o modelo BDI utiliza o raciocínio prático (RAO; GEORGEFF, 1995), como o 2-APL (DASTANI, 2008) e o Jason (BORDINI; HÜBNER; WOOLDRIDGE, 2007).

Algorithm 1: BDI-Interpreter (RAO; GEORGEFF, 1995)

```
initialize-state();
repeat
    selected-events := event-selector (even-queue);
    options := option-generator(selected-events,B,G,I);
    selected-options := deliberate(options,B,G,I);
    update-intentions (selected-options,I);
    execute(I);
    get-new-external-events();
    drop-successful-attitudes(B,G,I);
    drop-impossible-attitudes(B,G,I);
until quit;
```

2.3 ÂNIMO

O conceito de ânimo de (THAYER, 1989) foi uma inspiração para o processo de deliberação proposto. Thayer (1989) considera que o ânimo é uma relação entre *energia*, *tensão* e *benefício* (THAYER, 1996). O estado energético de uma pessoa está entre os extremos "total-

mente disposta" e "totalmente cansada". O estado de tensão está entre o "calmo" e o "estressado". De acordo como Thayer (1989), esses dois estados são combinados, sendo que, uma pessoal se sente bem quando se encontra em um estado disposto e calmo, e ela se sente mal quando está em um estado tenso e cansado.

Thayer (1989) afirma que existem diferentes elementos que podem regular o ânimo. Estes elementos são capazes de alterar o ânimo pois eles oferecem um benefício. O benefício produz satisfação e relaxa a tensão do indivíduo. Por exemplo, Thayer identificou uma conexão entre a comida e o ânimo quando uma pessoa, frequentemente, usa a comida para regular o ânimo. Essa pessoa sabe que a compulsão alimentar não é saudável (comer gera tensão), todavia continua a comer, pois a satisfação de comer é maior do que a tensão gerada pelo ato. Outro elemento que pode regular o ânimo é a atividade física. Por exemplo, grandes quantidades de hormônios são produzidos durante uma caminhada; por outro lado, a movimentação do corpo gera um gasto de energia. No entanto, os hormônios que ela produz podem neutralizar a tensão causando um bom estado de ânimo.

O indivíduo necessita encontrar um meio termo entre os elementos regulatórios do ânimo (comida, exercício, e outros) para atingir o estado perfeito de ânimo. Portanto, *o indivíduo precisa escolher atingir objetivos que evitam tensões e produzem mais benefícios dentro das suas limitações energéticas.*

3 TRABALHOS RELACIONADOS

> "... é semelhante a um homem que edificou uma casa sobre a terra sem alicerces; na qual a torrente deu com ímpeto, e logo caiu; e foi grande a ruína daquela casa."
>
> Lucas 6:49

Este capítulo tem como objetivo apresentar os trabalhos relevantes para a área do raciocínio normativo e como a proposta desse trabalho está localizada no estado da arte. O capítulo será divido em três seções. A primeira apresenta de forma genérica os elementos utilizados no raciocínio normativo. A segunda apresenta os trabalhos relevantes categorizados pelos elementos descritos na primeira seção. A última seção apresenta uma discussão sobre o estado da arte apresentado.

3.1 PRINCIPAIS ELEMENTOS UTILIZADOS NO RACIOCÍNIO NORMATIVO

Durante o reconhecimento do estado da arte, foram identificados diversos trabalhos relevantes ao raciocínio normativo. Esses trabalhos utilizam diferentes elementos associados às normas para realizar o raciocínio normativo. Tais elementos são a preferência, a dependência, a sanção, o deadline e os recursos. Esse último é foco do presente trabalho. A seguir, será apresentada uma definição desses elementos que servirá para classificar os trabalhos relevantes descritos na seção posterior.

A **preferência** é o elemento mais utilizado pelos trabalhos relacionados. A representação desse elemento ocorre de três maneiras diferentes: ordenação total, ordenação parcial e ordenação parcial condicional.

Na ordenação total, o agente possui uma estrutura que atribui um grau de preferência a cada norma. Dessa maneira, uma norma N_1 com grau de preferência maior que N_2, N_3 e N_4 tem o seu cumprimento prioritário em relação a N_2, N_3 e N_4.

Na ordenação parcial, o agente possui estruturas que representam a preferência entre duas normas e não há necessariamente uma ordem global para todas as normas. Por exemplo, a norma N_1 tem preferência de cumprimento entre N_1, N_2 e N_3. Contudo, não existe uma representação da preferência entre N_2 e N_3.

A **sanção** é outro elemento utilizado pelos trabalhos relacionados. Como descrito no capítulo anterior, uma sanção é um dos componentes da norma e representa a penalização imposta ao agente pelo seu não cumprimento. O uso de sanções é fundamentado pelo behaviorismo de Skinner (MOREIRA, 1999), que destaca o uso de reforços para modelar o comportamento de indivíduos de uma sociedade. Os trabalhos que utilizam a sanção como elemento de avaliação consideram que uma norma N_1, com uma sanção mais danosa ao agente que N_2, tem prioridade de cumprimento sobre N_2.

O **deadline** é um elemento utilizado apenas por um dos trabalhos relacionados. Raciocinar sobre o deadline de uma norma significa que o processo de deliberação observa se há tempo hábil para cumprir a norma e determina quais serão cumpridas. Por exemplo, o agente cumpre todas as normas que ele consegue cumprir antes de ultrapassar os respectivos deadlines. Para tanto, é possível utilizar algoritmos de escalonamento para fazer a escolha das normas que serão cumpridas.

A **dependência** é um elemento observado em dois trabalhos relacionados. O raciocínio baseado nesse elemento leva os agentes a raciocinar sobre as relações de poderes (CASTELFRANCHI, 2002). Uma relação de dependência existe quando um agente depende de outro para cumprir seus objetivos e normas. Por exemplo, um agente A_1 deve cumprir uma norma N_1, porque o não cumprimento afeta negativamente o agente A_2. Agente do qual A_1 é dependente, porque, para cumprir uma norma N_2, o agente A_1 precisa do auxílio de A_2.

O **recurso** é o elemento que não é explorado por nenhum dos trabalhos relacionados. Esse elemento é foco do raciocínio proposto por este trabalho. Um agente capaz de raciocinar baseado nesse elemento pode determinar quais normas serão cumpridas de forma a respeitarem os recursos disponíveis.

3.2 MODELOS DE RACIOCÍNIO NORMATIVO BASEADO EM PREFERÊNCIA

3.2.1 Dignum et al. (2000)

Dignum é um dos primeiros autores a propor uma arquitetura de agente normativo. Para tal, em Dignum et al. (2000) é proposto uma extensão da arquitetura clássica BDI para considerar normas. A primeira contribuição desse trabalho é o uso da norma para inferir as intenções do agente. Para tanto, o algoritmo clássico do BDI foi modificado em muitos passos, considerando a existência de crenças normativas e a ocorrência de novos eventos relacionados à ativação de normas. Nessa abordagem, o comportamento dos agentes é determinado pela norma. O agente não é capaz de negar o cumprimento de uma norma, ou seja, se o agente receber uma norma ele tentará cumpri-la. Os agentes seguem cegamente as normas e os conflitos são resolvidos de forma estática, de acordo com a preferência estabelecida pelas intenções.

3.2.2 Broersen et al. (2001)

Broersen et al. propõe um modelo estendido da arquitetura BDI com uma noção explícita de obrigação. Os agentes do tipo BOID são formados por quatro componentes: (B)Crenças, (O)Obrigações, (I) Intenções e (D) Desejos. Conflitos entre esses componentes são resolvidos através de uma função de ordenação estática. Dessa maneira, o agente não pode decidir cumprir ou violar as normas de acordo com a circunstância.

Em Broersen et al. (2001) são apresentados quinze diferentes tipos de conflitos, caracterizados em duas categorias: interna e externa. Os conflitos internos podem ser caracterizados pelos quatros subtipos unários (B; O; I; D).

Conflito B: o agente possui duas crenças conflitantes. Por exemplo, eu acredito que a luz da varanda está desligada, porque eu pedi para minha irmã desligá-la; eu acredito que a luz está ligada, pois, a última vez que a vi, ela estava ligada.

Conflito O: o agente possui dois objetivos conflitantes. Por exemplo, é obrigatório ser honesto. É obrigatório ser polido. Se eu for honesto sobre algo, e esse algo for indelicado.

Conflito I: o agente possui duas intenções conflitantes. Por exemplo, eu tenho a intenção de terminar o artigo no domingo. Eu tenho intenção de ir à praia no sábado. Se eu for à praia no sábado, eu não consiguirei terminar o artigo no domingo.

Conflito D: o agente possui dois desejos conflitantes. Por exemplo, eu desejo fumar. Eu desejo ser saudável. Todavia, fumar não é saudável.

Os conflitos externos podem ser descritos em seis conjuntos binários (BO; BI; BD; OI; OD; ID), quarto conjuntos ternários (BOI; BOD; BID; OID) e um conjunto quaternário (BOID).

Conflito BO: o agente possui uma crença e uma obrigação conflitantes. Por exemplo, é obrigatório ver minha madrasta nesse fim de semana. Mas eu acho que não tenho tempo para tal.

Conflito BI: o agente possui uma crença e uma intenção conflitantes. Por exemplo, eu tenho planos de ver minha madrasta nesse fim de semana. Todavia, eu acho que vai ser impossível, pois meu carro quebrou.

Conflito BD: o agente possui uma crença e um desejo conflitantes. Por exemplo, eu gostaria de ter um longo período de férias. Eu precisaria me afastar do trabalho por um tempo para tirar longas férias. Mas, eu não posso me afastar do trabalho.

Conflito OI: o agente possui uma obrigação e uma intenção conflitantes. Por exemplo, é obrigatório ver minha madrasta nesse fim de semana. Mas, eu tenho planos de ir a uma conferência. Se eu for à conferencia, eu não poderei ver minha madrasta.

Conflito OD: o agente possui uma obrigação e um desejo conflitantes. Por exemplo, é obrigatório não fumar em uma área fechada. Eu desejo fumar no meu escritório. Todavia, meu escritório é uma área fechada.

Conflito ID: o agente possui uma intenção e um desejo conflitantes. Por exemplo, eu gostaria de tirar uma soneca. Mas, eu tenho

intenção de pegar o avião.

Conflito BOI: o agente possui um conflito na composição de uma crença, uma obrigação e uma intenção. Por exemplo, se eu fumo, eu devo ir a uma área de fumantes. Eu tenho intenção de fumar. Todavia, eu sei que essa é uma área de não fumantes.

Conflito BOD: o agente possui um conflito na composição de uma crença, uma obrigação e um desejo. Por exemplo, se eu fumo, eu devo ir a uma área de fumantes. Eu desejo fumar. Todavia, eu sei que essa é uma área de não fumantes.

Conflito BID: o agente possui um conflito na composição de uma crença, uma intenção e um desejo. Por exemplo, eu tenho intenção de ir a uma conferência. Eu desejo que o custo dessa viagem não seja caro. Eu sei que se for a essa conferência o custo da viagem vai ser muito caro.

Conflito OID: o agente possui um conflito na composição de uma obrigação, uma intenção e um desejo. Por exemplo, eu tenho intenção de ir a uma conferência. Eu desejo ficar em um hotel luxuoso. Todavia, se eu for à conferência é obrigatório que não fique em um hotel luxuoso.

Conflito BOID: o agente possui um conflito na composição de uma crença, uma obrigação, uma intenção e um desejo. Por exemplo, eu tenho intenção de ir a uma conferência. É obrigatório que eu não gaste muito dinheiro. Para tanto, eu tenho de escolher entre um hotel caro e um voo barato ou um hotel barato e um voo caro. Eu desejo ficar em um hotel caro. Mas, a secretária comprou uma passagem para um voo caro.

A arquitetura proposta em Broersen et al. (2001) apresenta um comportamento parametrizado por uma função de ordenação p para resolver os conflitos. Dependendo do tipo de agente, os valores adotados em p seguem uma ordem fixa. O BOID define quatro tipos de agente.

O primeiro é o agente simplório social, no qual a ordem decrescente de prioridade são crenças (b), intenções (i), obrigações (o) e desejos (d).

$$b < i < o < d$$

O segundo tipo é o agente simplório egoísta, no qual a ordem decrescente de prioridade são crenças (b), intenções (i), desejos (d) e obrigações (o).

$$b < i < d < o$$

O terceiro é o agente mente aberta social, no qual a ordem decrescente de prioridade são crenças (b), obrigações (o), desejos (d) e intenções (i).

$$b < o < d < i$$

O último, por sua vez, é o agente mente aberta egoísta, no qual a ordem decrescente de prioridade são crenças (b), desejos (d), obrigações (o) e intenções (i).

$$b < d < o < i$$

A arquitetura BOID propõe um mecanismo para resolver conflitos entre crenças, obrigações, intenções e desejos. Para tal, utiliza de um algoritmo de ordenação que estabelece as prioridades de cada um dos elementos.

3.2.3 Sadri, Stathis e Toni (2006)

No trabalho de Sadri, Stathis e Toni (2006) é proposto uma extensão do KGP (*knowledge-goal-plan*) que adiciona noções sobre obrigações, proibições e papéis. O agente considera relevante todas as normas que afetam o papel desempenhado por ele. Outra característica desse modelo é que o agente tem motivação interna e objetivos próprios. Sendo assim, em caso de conflito entre normas e objetivos, os agentes escolherão sempre seguir o comportamento especificado pela norma. O principal foco dessa proposta é, através da consideração das normas, planejar e decidir qual a ação do agente no próximo passo.

No modelo KGP, o estado interno do agente é descrito pela tupla:

$$\langle KB, Goals, Plan, TCs \rangle.$$

Nessa tupla, KB é a base de crenças do agente, ou seja, o que ele sabe sobre ele mesmo e o ambiente. A base é segmentada para permitir o raciocínio sobre diferentes capacidades.

- KB_{plan}, para *planning*;
- KB_{pre}, para a identificação de precondições das ações;
- KB_{react}, para reatividade;
- KB_0, para representar o conhecimento do agente sobre o ambiente em que se encontra, incluindo comunicações passadas e percepções.

$Goals$ é um conjunto de propriedades que o agente quer atingir. Os objetivos estão associados com elementos temporais. $Plan$ é um conjunto de ações organizadas para satisfazer objetivos. Os planos também são associados elementos temporais. TCs é um conjunto de restrições temporais.

A capacidade de raciocínio do KPG pode ser dividida em três partes *planning*, reatividade e mudança de percepção. O *Planning* gera planos parciais para um conjunto de objetivos. Para tal, cria subobjetivos e ações para atingir os objetivos assumidos. A Reatividade opera sobre as percepções do ambiente. A mudança na percepção do agente incorre na alteração dos objetivos ou planos.

Para incorporar as normas no KGP, é necessário adaptar o modelo. Para tal, Sadri adapta a fase de *planning* e reatividade do modelo KPG. A reatividade alterada permite aos agentes estarem conscientes de suas obrigações e proibições, através das observações de suas obrigações no ambiente em que se situam. Isso força o agente a cumprir qualquer norma relevante do sistema normativo. *Planning* é usado para escolher a sequência de ações que permitirão ao agente cumprir obrigações e, se possível, evitar proibições.

3.2.4 Gaertner e Toni (2008)

A arquitetura proposta por Gaertner é mais um exemplo de agente orientado à norma, na qual ela sempre é obedecida. Esse modelo é uma extensão do *Multi-context BDI*, na qual as obrigações e proibições são consideradas e, para tanto, interpretadas como intenções. Como solução para conflito entre normas, o modelo propõe utilizar uma abordagem baseada em argumentação, além da função de preferência. A argumentação pode servir como uma ferramenta computacional efetiva e uma abstração viável para várias atividades dos agentes, em particular o raciocínio (GAERTNER; TONI, 2008).

O conflito entre normas e objetivos é resolvido em Gaertner e Toni (2008) através do cruzamento de informações (normas, crenças, desejos e intenções) e reforçado pela preferência do agente. As normas, quando internalizadas nesse modelo, recebem uma preferência, utilizada na resolução dos conflitos. A argumentação baseada em hipóteses é o método de argumentação utilizado no trabalho de Gaertner. Método esse tido como um poderoso mecanismo para compreender similaridades e diferenças sobre muitos frameworks de raciocínio legal (KOWALSKI; TONI, 1996), de raciocínio prático e epistêmico (GAERTNER; TONI, 2007) e para serviço de seleção e composição (TONI, 2007).

O trabalho de Gaertner raciocina sobre o comprometimento do agente com normas, crenças, desejos e intenções. Gaertner argumenta que utilizando preferências é possível, por exemplo, priorizar algumas crenças sobre algumas normas ou certas normas sobre desejos. Essas preferências, segundo Gaertner, podem ser chamadas de personalidade normativa de um agente. As preferências do agente podem ser estabelecida de três maneiras:

- Preferências como ordem total - Nesse caso, uma função que extrai de regras de ligação e elementos de teorias, axiomas e regras de inferência números racionais que representam a preferência. Esses valores quanto mais baixos, mais alta a preferência da informação em questão.

- Preferências como ordem parcial – Nesse caso, a função dá lugar a um conjunto de preferências do tipo $pref(\mu_i, \mu_j)$, onde μ_i tem preferência sobre μ_j. Na ordem parcial, os elementos não são classificados por uma preferência quantitativa, mas considerando um elemento de cada vez.

- Preferências dinâmicas via meta-regras - Como no caso da ordem parcial, a preferência é dada por um conjunto de preferências do tipo $pref(\mu_i, \mu_j)$, contudo meta-regras pondem alterar a ordem da preferência. Por exemplo, $pref(\mu_i, \mu_j) \leftarrow sol$ ou $pref(\mu_j, \mu_i) \leftarrow chuva$, permitindo o agente alterar a preferência entre dois elementos dependendo do clima.

3.3 MODELOS DE RACIOCÍNIO NORMATIVO BASEADO EM SANÇÃO

3.3.1 López e Marquez (2004)

A autonomia de um agente ainda é um elemento de discussão. A maior parte dos pesquisadores concorda que a autonomia é a propriedade que permite o agente tomar decisões (WOOLDRIDGE, 2002) e, para tanto, os agentes fazem uso de suas preferências ou **motivações**, que se relacionam com os seus objetivos.

De acordo com d'Inverno e Luck (2001), motivação é um componente de alto nível, não derivável, que permite raciocinar sobre fazer alguma coisa. Definindo, dessa maneira, a motivação como qualquer desejo ou preferência, que podem levar o agente a gerar e adotar objetivos. A motivação, ainda, afeta os resultados do raciocínio ou comportamento para satisfazer esses objetivos.

Em López e Marquez (2004) a autonomia relacionada aos objetivos do agente está associada a um conjunto de motivações. Dessa maneira, as preferências individuais geram uma importância diferente para cada objetivo. Assim, associando pesos aos objetivos na forma de valores, onde quanto maior o valor, mais relevante é o objetivo.

A figura 1 apresenta a arquitetura proposta em López e Marquez (2004). Observa-se que à arquitetura BDI são adicionados três processos: adoção de normas, cumprimento da norma e atualização do agente com as decisões normativas tomadas. Em paralelo, ocorrem três novas atitudes mentais: a instância das normas, seleção das normas e rejeição de normas. A seguir cada um desses processos será descrito.

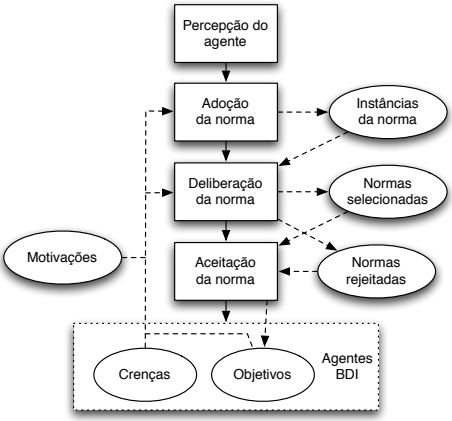

Figura 1 – Arquitetura abstrata de agente normativo

O **processo de adoção de normas** pode ser definido como o processo pelo qual os agentes reconhecem suas responsabilidades para com os outros através da internalização das normas. Para um agente adotar uma norma é necessário:

- o agente se reconhecer como endereço destino da norma;
- a norma ainda não deve ter sido adotada;
- a norma deve ser emitida por uma autoridade conhecida;
- o agente deve ter motivos para permanecer na sociedade.

O **processo de deliberação das normas** pode ser definido como o processo pelo qual os agentes analisam a relevância do cumpri-

mento das normas instanciadas. Para deliberar sobre normas os agentes seguem três passos.

No primeiro passo, um conjunto de normas ativas é selecionado do conjunto de normas adotadas (instâncias de normas). Normas ativas são as que, por um estado mental específico, o agente acredita ser necessário cumprir.

No segundo passo, o agente divide as normas ativas em não conflitantes e conflitantes. Uma norma ativa é não conflitante se o seu cumprimento não causa conflito com os outros objetivos do agente. As conflitantes, por sua vez, quando são cumpridas impedem a obtenção de outros objetivos do agente.

No terceiro passo, para cada um desses conjuntos de normas, o agente deve decidir qual executar. Dessa maneira, depois de deliberar sobre as normas, dois grupos são formados. O grupos das normas intencionadas, contendo as normas que o agente deseja cumprir, e o grupo das rejeitadas, contendo as normas que não serão cumpridas.

O **processo de cumprimento das normas** deve ser iniciado, a fim de atualizar a meta de um agente, de acordo com as decisões que ele tenha feito na deliberação das mesmas. Assim, o agente é influenciado de duas formas, dependendo se a norma foi intencionada ou rejeitada. Dessa forma, quatro situações podem ser observadas.

Na primeira, todos os objetivos normativos intencionados pela norma são adicionados no conjunto de objetivos do agente, pois em processo anterior foi decidido pelo cumprimento desses.

Na segunda, alguns objetivos são impedidos pelos objetivos normativos. Esses objetivos não podem ser atingidos porque o agente prefere cumprir com a norma. Consequentemente, o conjunto de objetivos impedidos é removido dos objetivos do agente.

Na terceira, alguns objetivos se beneficiam das recompensas obtidas de uma norma intencionada. Recompensas contribuem para a satisfação desses objetivos sem a necessidade do agente fazer um esforço extra. Como resultado desse benefício, o agente remove o objetivo e adiciona o objetivo normativo cumprindo-o e ainda respeitando a norma.

Na quarta, as normas rejeitadas apenas afetam o conjunto de objetivos, impedindo objetivos associados a sanções. Esse conjunto é removido dos objetivos do agente. Isso ocorre porque essa é a maneira do agente normativo lidar com as consequências impostas pela norma.

O trabalho de López e Marquez (2004) apresenta uma arquitetura que permite agentes BDI raciocinar sobre normas adicionando três novos processos. O primeiro decide quais normas serão adotadas, o segundo se as normas adotadas serão cumpridas e o terceiro atualiza a base de crença e objetivos do agente.

3.3.2 Criado, Argente e Botti (2011)

Em Criado, Argente e Botti (2011) é discutido como o agente pode deliberar sobre a aceitação de uma norma de acordo com seus interesses sobre uma motivação racional. Para tanto, Criado utiliza como base uma arquitetura de agente BDI normativa (n-BDI). A principal contribuição desse trabalho é a construção de estratégias para decidir quais normas os agentes devem aceitar ou rejeitar. As estratégias utilizam funções de utilidade baseada em recompensas e sanções.

Em sistemas multiagente abertos, onde os participantes tendem a ser heterogêneos, com confiança limitada, com objetivos individuais em conflito e com um elevado grau de incertezas no sistema, as normas representam uma ferramenta efetiva para realizar a coordenação e a cooperação dos membros do sistema (CRIADO; ARGENTE; BOTTI, 2011). Essas normas podem mudar com o decorrer do tempo para se adaptarem às mudanças do ambiente e o comportamento da sociedade de agentes. Segundo Criado, Argente e Botti (2011) estratégias para decidir quais normas os agentes devem aceitar e rejeitar permitem que os agentes se adaptem às normas, adequando-se ao sistema, modificando-o ou deixando-o.

Em Criado, Argente e Botti (2011) é discutido como o agente pode deliberar sobre a aceitação de uma norma de acordo com seus interesses sobre uma motivação racional. Para tal, Criado utiliza como base uma arquitetura de agente BDI normativa (n-BDI) composta por

7 módulos e representada na figura 2.

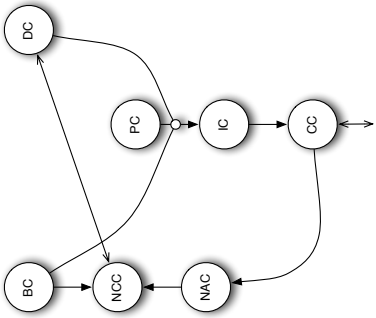

Figura 2 – Arquitetura n-BDI (CRIADO; ARGENTE; BOTTI, 2011)

Os módulos da arquitetura n-BDI são:

- Contexto da crença (BC). É composta por um conjunto de proposições lógicas do tipo (B_γ, β_γ), onde B_γ representa uma crença do agente e β_γ representa o grau de confiança na crença com um intervalo de 0 até 1.
- Contexto da intenção (IC). É formado por preposições lógicas do tipo (I_γ, ι_γ), onde I_γ representa uma intenção do agente e ι_γ representa o grau de confiança com um intervalo de 0 até 1.
- Contexto dos desejos (DC). É formado por preposições lógicas do tipo $(D*_\gamma, \delta*_\gamma)$, onde $D*_\gamma$ representa um desejo do agente e $\delta*_\gamma$ representa o grau de confiança com um intervalo de 0 até 1. $* \in \{+, -\}$ representa desejos positivos e negativos, respectivamente.
- Contexto de planejamento (PC). Esse módulo permite determinar a sequência de ações para atender as intenções e atingir os objetivos. Esse módulo é composto por preposições lógicas do tipo $plan(\Sigma)$, onde Σ é um conjunto de ações
- Contexto de comunicação(CC). Esse módulo permite ao agente comunicar-se com o ambiente.

- Contexto de aquisição de normas (NAC). Esse módulo é responsável por compreender as normas da sociedade. A aquisição pode ocorrer de modo direto ou por inferência. Na primeira, a norma é informa ao agente pelo sistema. Na segunda, o agente possui um sistema cognitivo que é capaz de inferir normas através da observação de comportamentos da sociedade.

- Contexto de aceitação da norma (NCC). Esse módulo é responsável pela aceitação das normas. As regras que o NCC segue são:

  – O contexto mental injeta fórmulas no NCC;

  – O NCC realiza um processo de inferência que define quais normas vão ser obedecidas considerando o estado mental atual;

  – O contexto BDI é modificado de acordo com as novas proposições mentais derivadas das normas aceitas.

Nos trabalhos Castelfranchi (1999a), Conte, Castelfranchi e Dignum (1999), López (2003) são apresentados diferentes técnicas de aceitação de normas. Um dos primeiros trabalhos que analisaram a motivação para aceitação de normas sobre a ótica dos agentes é apresentado por Castelfranchi (1999a). Nesse estudo, é determinado que a norma não requer apenas um comportamento do agente, mas também um estado mental.

Em Criado, Argente e Botti (2011) é proposto uma função de aceitação que suporta todas as categorias descritas em Castelfranchi (1999a). Essa função de aceitação é descrita como:

$$f_{compliance}(\delta^*_C, \delta^*_S, \delta^*_R, \rho_S, \rho_R) = \{0,1\}$$

Onde:

- $* \in \{+, -\}$, onde $+$ é o aspecto positivo e $-$ é o aspecto negativo;

- δ^*_C representa a importância da norma para o agente;

- δ_S^* representa a importância da sanção aplicada pelo descumprimento da norma;
- δ_R^* representa a importância da recompensa aplicada pelo cumprimento da norma;
- ρ_S representa a probabilidade de ocorrer S;
- ρ_R representa a probabilidade de ocorrer R;
- por fim a função retorna 0 ou 1 respectivamente rejeitando ou aceitando a norma.

O trabalho Criado, Argente e Botti (2011) classificou suas estratégias através do trabalho López (2003), no qual as estratégias são subdivididas em simples, motivacional, social e mistas.

Estratégias simples consideram apenas o efeito da aceitação que uma norma pode ter sobre os objetivos do agente e podem ser categorizadas como:

- Estratégia automatizada - todas as normas são aceitas, correspondendo a função de aceitação:

$$f_{compliance}(\delta_C^*, \delta_S^*, \delta_R^*, \rho_S, \rho_R) = 1 \tag{3.1}$$

- Estratégia rebelde - todas as normas são rejeitadas, correspondendo a função:

$$f_{compliance}(\delta_C^*, \delta_S^*, \delta_R^*, \rho_S, \rho_R) = 0 \tag{3.2}$$

- Estratégia temerosa - aceita todas as normas que possuem sanção:

$$f_{compliance}(\delta_C^*, \delta_S^*, \delta_R^*, \rho_S, \rho_R) = \begin{cases} 1 & \text{se } \delta_S^+ + \delta_S^- > 0 \\ 0 & \text{caso contrário} \end{cases} \tag{3.3}$$

- Estratégia gananciosa - aceita todas as normas que possuem recompensa:

$$f_{compliance}(\delta_C^*, \delta_S^*, \delta_R^*, \rho_S, \rho_R) = \begin{cases} 1 & \text{se } \delta_R^+ + \delta_R^- > 0 \\ 0 & \text{caso contrário} \end{cases} \quad (3.4)$$

Estratégias de motivação são baseadas no utilitarismo (MILL; SHER, 2001), no qual se procura maximizar a utilidade. Dessa maneira, o desejo do cumprimento da norma e a violação da norma são critérios de cumprimento.

- Estratégia egoísta - aceita as normas que beneficiam os objetivos do agente:

$$f_{compliance}(\delta_C^*, \delta_S^*, \delta_R^*, \rho_S, \rho_R) = \begin{cases} 1 & \text{se } \delta_C^+ > 0 \\ 0 & \text{caso contrário} \end{cases} \quad (3.5)$$

- Estratégia por pressão - aceita a norma caso os efeitos negativos da sanção sejam maiores que o efeito negativo do cumprimento do agente:

$$f_{compliance}(\delta_C^*, \delta_S^*, \delta_R^*, \rho_S, \rho_R) = \begin{cases} 1 & \text{se } \delta_S^- > \delta_C^- \\ 0 & \text{caso contrário} \end{cases} \quad (3.6)$$

- Estratégia oportunista - aceita as normas que têm um efeito positivo da recompensa maior que o efeito negativo do cumprimento da norma :

$$f_{compliance}(\delta_C^*, \delta_S^*, \delta_R^*, \rho_S, \rho_R) = \begin{cases} 1 & \text{se } \delta_R^+ > \delta_C^- \\ 0 & \text{caso contrário} \end{cases} \quad (3.7)$$

Estratégias sociais são baseadas no bem estar social, no qual se procura maximizar a qualidade do sistema. Nesse sentido, o desejo social do cumprimento da norma e a violação da norma são critérios utilizados.

- Estratégia cooperativa - aceita as normas que o agente considera benéficas à sociedade. Essa estratégia pode ser implementada definindo os objetivos sociais como desejos do agente.

- Estratégia benevolente - aceita as normas que o agente considera benéficas a um agente (alvo) que ele queira favorecer.

Estratégias mistas são baseadas nas estratégias anteriores e se utilizam de múltiplos componentes.

- Estratégia mista - aceita as normas que têm um efeito positivo maior na sua aceitação do que efeito negativo na sua rejeição:

$$f_{compliance}(\delta_C^*, \delta_S^*, \delta_R^*, \rho_S, \rho_R) = \begin{cases} 1 & \text{se } \delta_R^+ + \delta_C^+ + \delta_S^- > \\ & \delta_R^- + \delta_C^- + \delta_S^+ \\ 0 & \text{caso contrário} \end{cases} \tag{3.8}$$

- Estratégia mista ponderada - aceita as normas que têm um efeito positivo maior na sua aceitação do que efeito negativo na sua rejeição, fazendo uma ponderação através da probabilidade de ocorrência:

$$f_{compliance}(\delta_C^*, \delta_S^*, \delta_R^*, \rho_S, \rho_R) = \begin{cases} 1 & \text{se } (\rho_R * \delta_R^+) + \delta_C^+ + \\ & (\rho_S * \delta_S^-) > (\rho_R * \delta_R^-) + \\ & \delta_C^- + (\rho_S * \delta_S^+) \\ 0 & \text{caso contrário} \end{cases} \tag{3.9}$$

O trabalho Criado, Argente e Botti (2011) apresenta como uma arquitetura n-BDI pode auxiliar o agente a adquirir, deliberar e tomar decisões sobre as normas, utilizando uma definição de estratégia discutida em trabalhos anteriores. Um dos destaques de Criado é o grau de significância das normas que permite aos agentes considerarem não apenas se as normas são benéficas ou não, mas determinar qual a intensidade do efeito causado por ela. Como projeto futuro, o trabalho, propõe a análise da motivação não racional, a fim de atribuir emoções às estratégias de aceitação de normas.

3.4 MODELOS DE RACIOCÍNIO NORMATIVO BASEADO EM DEPENDÊNCIAS

3.4.1 Carabelea, Boissier e Castelfranchi (2005)

A proposta de Carabelea, Boissier e Castelfranchi (2005) utiliza a teoria de poder social (CASTELFRANCHI, 2002) para permitir um agente raciocinar sobre a entrada em um grupo. Segundo Carabelea, Boissier e Castelfranchi (2005), diversos autores propõem a noção de poder social como paradigma para poder explicar o comportamento dos agentes (por exemplo, Castelfranchi (2002) e Jones e Sergot (1996)). Um dos usos dessa teoria é tornar possível para o desenvolvedor e o observador do sistema multiagente analisar e predizer o comportamento dos agentes no grupo. Outro uso é habilitar o agente a raciocinar sobre seus poderes e dos outros agentes presentes no grupo. Após o agente ingressar na organização, ele segue cegamente as normas impostas a ele.

Conforme Castelfranchi (2002) argumenta, a noção de poder não é intrinsecamente social e não faz apenas referência à teoria social ou pelo menos da ação social. Carabelea baseia o seu trabalho em Sichman et al. (1994), utilizando as noções básicas de ação, recurso, plano e objetivo (respectivamente a, r, pl, o), para descrever o comportamento do agente. Essas noções são utilizadas para definir o **poder individual** do agente. Em Carabelea, Boissier e Castelfranchi (2005) são usados os poderes individuais para identificar as relações de poder entre agentes, que ele denomina como **poder social**.

Um agente pode se valer de diversos tipos de poderes individuais dentro de um grupo. Tais poderes podem ser classificados como: poderes de execução (can_do), poderes deônticos ($entitled\_to$) e poderes totais ($Power$). Os poderes de execução representam as ações que o agente pode realizar. Por exemplo, um agente pode fazer uma ação se ele tiver todos os recursos, habilidades e conhecimentos necessários para executar a ação, ou seja, diz-se que um agente pode atingir um objetivo se ele tiver um plano ou possa obtê-lo e o agente tem todos os

recursos necessários e pode executar todos os sub-planos e ações.

A noção de poder social é relativamente comum em sistemas multiagente onde encontrarmos situações nas quais o agente possui apenas alguns componentes do poder. Desse modo, é necessário que os agentes cooperem entre si, construindo uma rede de dependências. Por exemplo, um agente Y pode fornecer uma ação para X de modo que X possa atingir um objetivo. Nessa situação, o agente X depende do agente Y, a qual podemos descrever como $depends\_on(X,Y,ag)$.

O foco central do trabalho Carabelea, Boissier e Castelfranchi (2005) é proporcionar aos agentes a habilidade de raciocinar sobre as relações entre eles e a sociedade que os envolve. Permitindo, assim, que os agentes sejam capazes de raciocinar sobre o seu ingresso ou não em um grupo. Para atingir esses objetivos, os autores utilizaram a teoria de poder.

3.4.2 Kollingbaum e Norman (2006)

Na arquitetura de Kollingbaum e Norman (2006), as obrigações são motivações e as proibições restringem as escolhas de atividades dos agentes. Basicamente, os agentes conhecem as ativações e expirações da norma e determinam quais normas são mais relevantes em um determinado momento. Nessa arquitetura, os agentes não têm motivação interna. Logo, eles tentam sempre cumprir as normas. Sendo a principal causa de violação, o conflito de normas. O trabalho de Kollinbaum foca-se na definição de algoritmos e procedimentos para detectar e resolver conflito de normas.

No modelo proposto por Kollingbaum, as normas possuem uma condição de ativação. Quando essa condição é satisfeita, as normas são adicionadas a um conjunto de normas instanciadas. Ao habilitar as normas, é atualizado o conjunto de planos candidatos que conduzirá o agente ao seu cumprimento. Essas normas podem ser conflitantes e, portanto, é necessária sua resolução antes da execução dos planos.

Os algoritmos para resolução de conflito observados no trabalho de Kollingbaum são sete: decisão arbitrária, recência, antiguidade, cau-

tela, audácia, poder social e negociação com o emissor da norma. Essas estratégias de resolução de conflito são aplicadas na fase de deliberação do agente, auxiliando - o a dividir o conjunto de planos candidatos em planos permitidos e proibidos.

A estratégia de decisão arbitrária pode ser utilizada como uma forma simplificada de decisão. Os planos em conflitos não levam em consideração nenhuma característica sobre a situação do conflito, além da preferência preestabelecida no agente.

A recência e a antiguidade requerem que sejam registrados estampilhas de tempo com o momento de ativação da norma. As normas ativadas são ordenadas por suas estampilhas de tempo e selecionadas conforme a recência ou antiguidade para sua execução.

Quando um agente utiliza a estratégia cautelosa, a proibição sempre prevalece sobre a permissão e o objetivo, ao contrário do que ocorre com a audaz, onde permissões sobrepõem-se a proibições.

Em outra estratégia, o agente pode decidir renegociar uma norma com a organização, podendo, assim, cancelar proibições ou receber permissões adicionais que sobreponham as proibições.

Por fim, a última estratégia abordada leva em consideração o poder social. Nessa abordagem, a norma com maior poder ganha o conflito. A teoria de poder social (SICHMAN et al., 1994) foi descrita em maiores detalhes na seção que apresenta o trabalho de Carabelea, Boissier e Castelfranchi (2005).

3.5 MODELOS DE RACIOCÍNIO NORMATIVO BASEADO EM DEADLINES

3.5.1 Alechina, Dastani e Logan (2012)

A proposta do trabalho de Alechina apresenta a N-2APL, uma linguagem de programação baseada no paradigma BDI, que suporta agentes capazes de raciocinar sobre as normas. A N-2APL adiciona à 2APL (DASTANI, 2008) o suporte a conceitos normativos como obrigações, proibições, sanções, prazos e durações. A deliberação ocorre com

a utilização de um algoritmo de escalonamento e leva em consideração os prazos e durações, além do critério de prioridade.

A cada ciclo o sistema normativo gera N-eventos que são mensagens em broadcast sobre as obrigações e proibições. A obrigação recebida por um agente através de N-eventos segue uma estrutura $obligation(\iota, o, d, s)$. Lendo-se, o agente ι tem que cumprir a obrigação o no deadline d sob pena de sofrer a sanção s. Já a proibição recebida segue uma estrutura $prohibition(\iota, p, s)$. Lendo-se, o agente ι é proibido de realizar ações que satisfaçam p, sob pena de sofrer a sanção s.

Nesse caso, onde obrigações e proibições são dotadas de prazos e duração, é provável que ocorram situações nas quais nem todas as obrigações e proibições possam ocorrer dentro do prazo especificado pelo sistema. No trabalho de Alechina, Dastani e Logan (2012), para solucionar essa situação são classificadas as obrigações e proibições, utilizando as recompensas e sanções impostas ao agente. O peso da recompensa e da sanção é estabelecido a critério do agente. Por exemplo, o agente ι recebe 2 n-eventos contendo uma obrigação X com recompensa L e uma proibição Y com sanção K, onde o peso de K é maior que a recompensa L. Logo, a prioridade de cumprir a proibição é maior que a da obrigação.

Além de ordenar as normas impostas, o agente deve raciocinar sobre a relação de conflito entre os seus objetivos pessoais e as suas obrigações perante o grupo. Nesse caso, o trabalho de Alechina, Dastani e Logan (2012) segue o modelo BOID (RIEMSDIJK et al., 2005), onde os agentes atribuem a prioridade conforme a sua especificação. Por exemplo, um agente social é caracterizado por dar prioridade às suas obrigações, em contrapartida agentes egoístas atribuem maior prioridade aos seus objetivos pessoais.

Se os objetivos do agente e suas obrigações não são conjuntamente atingíveis, um agente normativo racional tem que escalonar suas intenções para atingir os objetivos e obrigações de maior prioridade. Essa programação do agente não é trivial. Ela requer que os planos que o agente deve executar para atingir os objetivos e obrigações sejam

executados dentro do deadline e que os mesmos não violem proibições.

A linguagem N-2APL é apresentada como uma linguagem normativa racional, sendo uma extensão da linguagem 2APL que acrescenta os conceitos normativos como obrigação, proibição, sanção, deadline e duração, além de ser capaz de tratar conjuntos de objetivos e obrigações não atingíveis simultaneamente.

A linguagem N-2APL

A linguagem 2APL (DASTANI, 2008) usa um sistema de regras para planejar a execução dos planos e, assim, atingir os objetivos do agente. Esse sistema de regras da linguagem é nomeado como PG-Rules, na qual a linguagem N-2APL adiciona a questão temporal formalizando-a como:

$$\kappa \rightarrow \beta | \pi : t.$$

Onde κ é a meta desejada, β são as crenças que indicam o estado do agente no qual o plano é válido, π é o plano a ser executado para atingir κ e t é o tempo que o plano requerido leva para executar π.

As obrigações e proibições informadas pelo sistema normativo através de n-eventos são adicionadas aos agentes quando os mesmos percebem os eventos. As proibições são adicionadas à base de crenças e as obrigações são adicionadas aos objetivos do agente. A determinação dos objetivos e sanções mais prioritárias são definidas pelo projetista do sistema que utiliza relações binárias como:

$$pref(x, y),$$

onde x é mais prioritário que y. Por fim, para possibilitar o raciocínio sobre as proibições, é utilizado o componente $effect\{\pi\}$, que contém os efeitos causados pela execução do plano π.

Para um agente tratar conjuntos de objetivos e obrigações não atingíveis simultaneamente, é necessário que o agente trate de questões temporais. No trabalho de Alechina, Dastani e Logan (2012) esse tratamento é feito deliberando sobre quais planos devem ser executados

seguindo uma política de escalonamento para atingir os objetivos mais prioritários.

O agente, nessa abordagem, determina que um conjunto viável de planos obedecem a uma ordem de preferência. Se não for possível executar todos os planos devido aos seus deadlines, serão descartados os planos que possuírem objetivos menos prioritários em preferência dos mais prioritários. Da mesma maneira, os planos são eliminados se violam proibições de prioridade mais alta que o objetivo.

O modelo de programação para agente normativo, proposto em Alechina, Dastani e Logan (2012), apresenta a adição dos conceitos da organização para o raciocínio dos agentes. Esse modelo faz uso de um escalonador de planos para tratar os deadlines presentes nas normas. Atribuindo uma ordem de preferência às obrigações e às sanções, permite ao escalonador executar o algoritmo preference-maximal. Nesse modelo, a atribuição da preferência aos planos e efeitos causados são fixos, sendo esses informados pelo projetista do sistema. O uso de um conjunto conflitante de normas proveniente de múltiplas organizações não é tratado pelo modelo.

3.6 OUTROS MODELOS DE RACIOCÍNIO NORMATIVO

3.6.1 Castelfranchi (1999b)

Castelfranchi (1999b) é um dos primeiros trabalhos a observar a necessidade de um agente capaz de raciocinar sobre normas. Esse trabalho propõe como uma arquitetura de um agente pode ser estendida com uma noção explícita de norma. Essa arquitetura permite aos agentes representarem normas e deliberarem quando seguirão ou a violaram. Esse trabalho apresenta uma arquitetura para agentes normativos, e detalhes como tarefas e deliberações ficam a cargo do agente. Os autores não especificam como essas tarefas são executadas, tornando o modelo intuitivo e abstrato. A proposta não formaliza como são as conexões lógicas entre normas e os estados mentais.

De acordo com Castelfranchi, estratégias para aceitação da norma

são classificadas como:

- Aceitação incondicional implica o agente não ter capacidade para rejeitar a norma. Sendo assim, ele sempre irá cumpri-la.
- Aceitação instrumental implica o agente ter um grande nível de autonomia, uma vez que ele adota uma norma caso a considere benéfica aos seus objetivos.
- Aceitação cooperativista implica o agente aceitar as normas que ele considera capazes de ajudar a sociedade a atingir seus objetivos.
- Aceitação benevolente implica o agente aceitar as normas que beneficiam outros agentes que ele considera necessário favorecer.

3.6.2 Meneguzzi e Luck (2009)

Em Meneguzzi e Luck (2009) é proposto uma arquitetura para agentes BDI que permite ao agente modificar seu comportamento baseado em normas. A proposta de Meneguzzi não tem intenção de apresentar resolução de conflitos entre normas. Há apenas o interesse em como o agente altera o seu comportamento perante as normas aceitas.

Quando um agente entra em um novo ambiente, é possível que ele encontre um novo conjunto de normas diferente das quais ele foi designado. No momento em que ele toma consciência dessas normas, ele deve ajustar seu comportamento para ficar de acordo com o novo conjunto de normas. Em Meneguzzi e Luck (2009), é proposta uma arquitetura para agentes BDI que permite ao agente modificar seu comportamento baseado em normas.

Como primeiro passo, em Meneguzzi e Luck (2009), a norma é representada por:

$$norm(Activation, Expiration, Norm),$$

na qual os dois primeiros termos são as condições da norma e o último termo é o conteúdo da norma. Por exemplo,

$$norm(in(britain), \neg in(britain), obligation(driveOn(left))).$$

Quando um agente percebe uma nova norma, ele deve decidir se aceita e modifica seu comportamento ou se rejeita e sofre uma sanção. O processamento das normas, apresentado em Meneguzzi e Luck (2009), é representado na figura 3, onde o ambiente fornece normas que são aceitas ou recusadas pelos agentes. Caso aceitas, as normas passam por um processo de verificação de consistência, e, após a resolução dos conflitos, há o processo de mudança de comportamento.

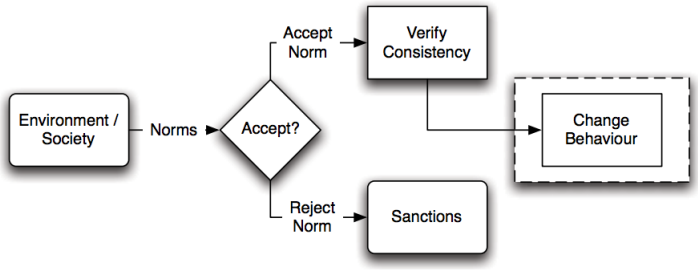

Figura 3 – Processamento das normas (MENEGUZZI; LUCK, 2009)

Após um agente aceitar uma norma, ele precisa ter certeza que essa não está em conflito com o conjunto de normas já aceitas. Por exemplo, uma norma n_1 proíbe uma ação α e uma norma n_2 obriga a execução de α, logo n_1 e n_2 estão em conflito. Dessa maneira, um agente deve tomar cuidado com as normas aceitas para não obter esse tipo de conflito. Esse tipo de conflito não é tratado no trabalho (MENEGUZZI; LUCK, 2009).

Em Meneguzzi e Luck (2009), quando uma norma é aceita, ela é adicionada à base de crenças do agente na forma de crença ou objetivo. Por outro lado, se a norma não é aceita, ela não é adicionada. Sendo assim, quando há uma modificação na base de crenças, uma nova norma pode ter sido aceita ou removida, necessitando de uma modificação de

comportamento. Essa modificação ocorre de diferentes maneiras para obrigações e proibições.

Quando ocorre uma alteração da base de crenças por um evento (adição ou exclusão) originado por uma obrigação O, são aceitas três ações:

- ignorar a norma. Acontece quando a condição de término é verdadeira. Uma vez que não há sentido em executar uma obrigação fora do prazo de validade;
- adotar o objetivo O. Acontece quando a condição de ativação é verdadeira e a de término é falsa;
- adicionar o plano que atinge O. Acontece quando a condição de ativação e de término são falsas.

Para cumprir uma obrigação, primeiramente, sua condição de término deve ser falsa. Passando por esse teste, é adicionado um plano que atinja O à biblioteca de planos. Dessa forma, quando a condição de ativação estiver presente na base de crenças, o plano que atinge O é executado. Por fim, quando a condição de término é verdadeira, um segundo plano é ativado e remove os planos referentes à obrigação O.

Por outro lado, quando uma proibição P gera uma modificação na base de crenças, é possível que 3 ações ocorram:

- ignorar a norma. Acontece quando a condição de término é verdadeira, uma vez que não há sentido em executar uma obrigação fora do prazo de validade;
- eliminar as intenções que atingem P. Acontece quando a condição de ativação é verdadeira e a de término é falsa;
- adicionar o plano que suprime planos que atingem P. Acontece quando a condição de ativação e a de término são falsas.

Para cumprir uma proibição, primeiramente, sua condição de término deve ser falsa. Passando por esse teste, é adicionado um plano que remove todas as intenções que levem à obtenção de P e a supressão

dos planos que atingem P. Assim sendo, quando a condição de ativação estiver presente na base de crenças, o plano é executado impedindo P. Por fim, quando a condição de término é verdadeira um segundo plano é ativado e remove os planos criados pela proibição.

Para implementar tal arquitetura, em Meneguzzi e Luck (2009), foi apresentado uma toolkit, que permite executar os algoritmos descritos anteriormente, adicionando funcionalidades de análise da biblioteca de planos à linguagem AgentSpeak(L). O framework apresentado em Meneguzzi e Luck (2009) é suficientemente genérico e pode ser extendido por um BDI tradicional, comprovado pela implementação descrita no trabalho.

3.6.3 Joseph et al. (2010)

A arquitetura proposta por Joseph permite ao agente raciocinar sobre a aceitação da norma. O agente participa de um processo de argumentação, propondo, aceitando ou rejeitando obrigações. O critério de aceitação ou rejeição de uma norma é a coerência entre elas. Esse trabalho foca-se exclusivamente na aplicação da coerência para o dilema da aceitação de normas.

A teoria de coerência utilizada por Joseph, quando usada para explicar o raciocínio humano, propõem que esses humanos aceitem ou rejeitem uma cognição (externa ou interna), dependendo como essa contribui para a maximização das restrições impostas pelas situações e outras cognições.

De acordo com a teoria de coerência, existem relações de coerência ou incoerência entre peças de informações dependendo se elas se apoiam ou se contradizem uma com as outras. Se duas peças de informação não são relacionadas, então não há coerência (restrição) entre elas. Joseph apresenta essa relação através de um grafo onde os nós são as peças de informação e as arestas são as restrições entre elas. O mecanismo de Thagard utilizado por Joseph computa a coerência global do grafo baseado na maximização da satisfação das restrições entre os pares de nós.

A coerência formalizada por Thagard (2002) utilizada por Joseph segue a noção básica de que um conjunto de peças de informação é representado como nodos de um grafo com peso nas arestas, ou restrições, entre esses nodos. Assim, algumas dessas restrições são positivas (representando coerência) e outras negativas (representando incoerência), e, associado a cada restrição, há um número que indica o peso da restrição. Com essa definição, a maximização da coerência é formulada como um problema de particionamento do conjunto de nodos em dois conjuntos: os aceitos e os rejeitados. Nesse sentido, para maximizar a aceitação, são seguidas duas condições de coerência:

- se a aresta $\{v, w\}$ é positiva, então $v \in A$ se e somente se $w \in A$
- se a aresta $\{v, w\}$ é negativa, então $v \in A$ se e somente se $w \in R$

Se uma aresta cumpre uma das condições acima, então, Thagard define-a como uma restrição satisfeita. O problema de coerência é resolvido pela maximização da soma dos pesos das restrições satisfeitas.

No trabalho de Joseph et al. (2010), escolheu-se apenas trabalhar com um tipo de coerência a qual denominou coerência dedutiva, porque o teorema de dedução lógica, a partir do qual é derivado, é bem compreendido. Thagard apresenta mais 5 outros tipos de coerência: explanatória, conceitual, analógica, perceptiva e deliberativa, cada uma com seu próprio conjunto de elementos e restrições.

A solução apresentada por Joseph converte o problema de maximização de coerência em um problema de *max-cut*. O *Max-cut* pertence à categoria de problemas conhecida como NP-completo.

Joseph estendeu a implementação de Thagard incorporando um meta interpretador baseado em Prolog para extrair provas de cada uma das sentenças de um agente BDI. As provas obtidas possibilitam definir os valores entre as peças de informação. Joseph usa um algoritmo de programação de aproximação semidefinida para resolver o problema de *Max-cut* para avaliar a coerência do gráfico e para determinar os nós do conjunto aceite.

3.7 DISCUSSÃO

Nesta seção, são discutidos os trabalhos apresentados anteriormente, abordando suas principais características e problemas. A tabela 1 classifica tais trabalhos seguindo como critério os elementos utilizados pelo raciocínio normativo. Nesta tabela, o X corresponde ao critério utilizado pelo modelo.

Três dos modelos observados não se encontram na tabela 1. O modelo apresentado em Castelfranchi (1999b) não está presente, pois trata-se de um modelo mais intuitivo e menos formal. O trabalho apresentado em Meneguzzi e Luck (2009) não é exatamente um trabalho sobre raciocínio, apesar de apresentar um modelo que permite o agente mudar o seu comportamento. Logo, não possui uma classificação quanto a critérios de avaliação de raciocínio. Por fim, o trabalho de Joseph et al. (2010) não é apresentado na tabela, pois usa um sistema de coerência para determinar quais normas serão aceitas, para tal ele faz uso dos conceitos apresentados em Thagard (2002).

Modelo	Pref.	Sanção	*Deadline*	Depend.	Recursos
Dignum	X				
Boella	X				
BOID	X				
Carabelea				X	
López		X			
Noa	X		X	X	
N. KPG	X				
Gaertner	X				
Criado		X			
Alechina	X		X		

Tabela 1 – Modelos X critérios de avaliação

Na grande maioria dos modelos analisados, o critério utilizado para o raciocínio é uma estrutura de preferência. Essa estrutura é está-

tica ou dinâmica, dependendo do modelo. Para um agente utilizar uma estrutura de preferência estática a fim de deliberar sobre normas restringe sua autonomia, uma vez que o agente é condicionado a sempre agir de forma a respeitar a ordem preferencial definida em sua concepção. O uso de uma estrutura dinâmica permite ao agente ajustar seu comportamento durante a sua execução, garantindo um grau de autonomia maior. Outro quesito a ser observado nos modelos que utilizam uma estrutura de preferência é a falta de definição de como essa estrutura é construída e quais os componentes que são utilizados.

A grande maioria dos modelos destaca que a sanção é um critério relevante na deliberação normativa, contudo a sanção é efetivamente utilizada como critério de raciocínio normativo em apenas dois modelos López e Marquez (2004) e Pacheco (2012). Desses modelos, o trabalho de López e Marquez (2004) é o que melhor caracteriza o uso das sanções. Nele, a sanção determina qual a motivação do agente em cumprir a norma.

Dos trabalhos observados, apenas dois utilizam o *deadline* como critério de raciocínio. Desses dois trabalhos, o mais relevante é o Alechina, Dastani e Logan (2012), pois trata especificamente do raciocínio normativo baseado em *deadlines*. Por sua vez, Kollingbaum e Norman (2006) utiliza diversos critérios e o tratamento dos *deadlines* são simplificados. Em Alechina, Dastani e Logan (2012) é proposto uma linguagem N-2APL, que habilita o agente a reconhecer as normas aceitas e verificar quanto tempo uma determinada norma demora para ser cumprida. O modelo proposto por Alechina, Dastani e Logan (2012), além do *deadline*, utiliza o critério de preferência para determinar quais as normas que serão cumpridas a cada ciclo do agente. O principal problema dessa abordagem é a complexidade de determinar o tempo que o agente demora para cumprir uma norma, uma vez que esses modelos se utilizam de arquiteturas não determinísticas.

O uso da teoria de poder (dependência) ocorre em dois modelos. Dentre eles, o mais relevante é o trabalho de Carabelea, Boissier e Castelfranchi (2005). Nesse trabalho, em específico, o agente se compromete com os objetivos e proibições ao entrar na organização, contudo,

após entrar, o agente pode vir a não cumprir esse acordo no decorrer da execução. Tal situação ocorre quando o agente atinge um estado que o leve a infringir uma norma, ou um estado que não lhe permita atingir todos os objetivos em tempo hábil. Sendo assim, esse modelo não garante que o agente vá cumprir suas obrigações e proibições em tempo de execução.

Os modelos apresentados utilizam diferentes critérios para aceitar as normas, mas nenhum deles utiliza como critério no raciocínio normativo os recursos utilizados pelos agentes. Assim, essas abordagens possuem dificuldades de trabalhar com agentes com recursos finitos. A natureza do recurso finito pode ser das mais diversas. Para fins de exemplificação, um recurso finito pode ser uma fonte de energia (e.g. bateria), um elemento do ambiente disponível ao agente (e.g. ponto de extração de minérios para um agente minerador), ou um item deteriorável do agente (e.g. pneus).

Um agente capaz de raciocinar sobre normas observando seus recursos consegue gerenciá-los de forma a obter o melhor resultado para o sistema e para os seus objetivos pessoais dentro das suas limitações. Os agentes móveis são um exemplo de aplicação para o raciocínio normativo baseado em recursos, uma vez que esses têm a necessidade de agir em conformidade com o sistema, bem como a necessidade de gerenciar seus recursos.

Para suprir essa lacuna no estado da arte, no próximo capítulo será proposto um modelo de raciocínio capaz de deliberar sobre normas, levando em consideração os recursos disponíveis e as implicações da aceitação (recompensa) ou recusa (sanção) da norma para o agente.

4 MODELO DE RACIOCÍNIO BASEADO EM ÂNIMO - HUGINN

> "O resultado do pensamento não tem de ser o sentimento mas a atividade."
>
> Vincent Van Gogh

Este trabalho tem como objetivo propor uma arquitetura mental e um processo de raciocínio que permita a um agente deliberar sobre normas e objetivos, levando em consideração os recursos disponíveis aos agentes. O modelo proposto foi nomeado como Huginn[1].

A figura 4 representa, de forma genérica, o contexto do trabalho. O agente reconhece os seus desejos pessoais e obtêm recursos e normas do ambiente, delibera sobre eles e, por fim, altera seu comportamento de forma a buscar um resultado melhor.

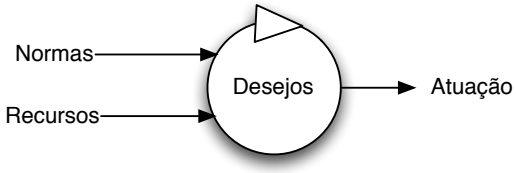

Figura 4 – Comportamento do agente

Para que um agente possa se comportar desta maneira, é necessário que ele raciocine sobre as normas, desejos e recursos disponíveis. Logo, é necessário que o agente:

- **obtenha** normas, desejos e recursos;
- tenha **representação mental** das normas, desejos e recursos;
- **delibere** sobre as normas, desejos e recursos;

[1] Na mitologia nórdica, Huginn (do nórdico antigo "pensamento") e Muninn (do nórdico antigo "memória" ou "mente") são um par de corvos que voam por todo o mundo conhecido como Midgard, trazendo informações ao deus Odin (BULFINCH, 1998).

- **atue** com o ambiente.

O presente trabalho foca-se em duas dessas características, a **representação** e a **deliberação**. Todavia, a arquitetura que é apresentada nesse capítulo leva em consideração o processo de obtenção de normas e atuação do agente.

O capítulo será dividido em duas seções: a representação e a deliberação. Na primeira, é apresentado uma arquitetura de agente capaz de representar normas. Na segunda, seção é apresentado o processo de deliberação.

4.1 ARQUITETURA DO AGENTE

Para construir a arquitetura apresentada a seguir foi estudado o modelo BDI (RAO; GEORGEFF, 1995; BRATMAN, 1999). Com a definição do BDI obtemos a arquitetura simplificada apresentada na figura 5, na qual o ambiente do SMA fornece as normas e os recursos. O agente, por sua vez, tem como função:

- perceber e representar as normas e os recursos;
- conhecer os próprios desejos;
- deliberar sobre as normas, desejos e recursos;
- buscar a maximização da recompensa;
- atuar de acordo com os objetivos escolhidos.

No modelo proposto, o agente pode perceber dois tipos de normas (obrigação e proibição) As permissões não são percebidas, pois o modelo considera que tudo é permitido, menos o que é proibido. Sendo assim, as permissões não existem de forma explícita. O é a representação das normas do ambiente, como definido na seção 2.1.1, é apresentada na tupla

$$\langle D, C, T, A, E, S, R \rangle$$

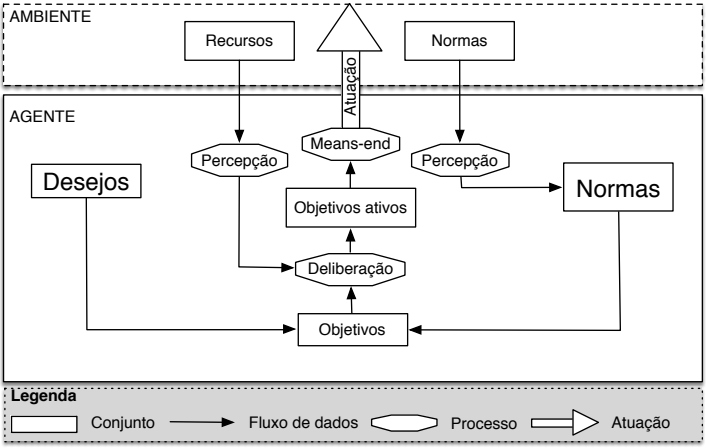

Figura 5 – Arquitetura do agente

na qual $D \in \{O, F\}$ é a modalidade deôntica da norma, obrigação (O), proibição (F); C é o objetivo controlado pela norma; T é o agente ou papel alvo da norma; A é a condição de ativação da norma; E é a condição de expiração da norma; S é a sanção imposta caso o agente descumprir a norma; R é a recompensa recebida caso a norma seja cumprida. Por exemplo, a norma para o veículo parar no sinal vermelho é uma obrigação de parar o veículo cujo alvo é o motorista. A norma é ativada quando o sinal está vermelho. O motorista é sancionado com uma multa de 100 reais caso não cumprir a norma e recebe um bônus no IPVA de 10 reais se cumprir a norma. Este exemplo é representado pela tupla:

$$\langle O, parar, motorista, sinalvermelho, \neg sinalvermelho, 100, 10 \rangle$$

Os desejos serão tratados nesse modelo como as vontades do agente. Os desejos, diferentemente das normas, têm sua origem no próprio agente, representando assim um caráter individual. Os desejos são compostos por um estado que o agente almeja (objetivo - C), uma condição de expiração (E), uma necessidade(N) e uma intensidade (I). A condição de expiração representa a validade do desejo. A necessidade

e a intensidade são valores escalares que representam, respectivamente, o quanto é essencial e o quanto é desejado atingir o objetivo. Nesse trabalho, representamos o desejo através da tupla:

$$\langle C, E, N, I \rangle$$

Por exemplo, um agente guloso tem um grande desejo (**intensidade** 10) de estar **satisfeito**, contudo o agente possui um alto percentual de gordura. Logo, ele possui uma baixa necessidade de se alimentar no momento (**necessidade** 2). Este desejo só não fará mais sentido quando o agente estiver saciado. Esse desejo é representado pela tupla

$$\langle alimentado, saciado, 2, 10 \rangle$$

Os recursos tratados, neste trabalho, são os elementos que o agente dispõem para cumprir os objetivos do agente. Esses elementos são individuais, logo os agentes não compartilham recursos. Os recursos são gastos pelo agente durante o cumprimento dos objetivos. Por outro lado, o incremento dos recursos é dado pela percepção de novos recursos. A representação dos recursos é dada por um identificador (r) e uma quantidade disponível (Q). A definição formal para os recursos utilizadas neste trabalho é uma tupla:

$$\langle r, Q \rangle$$

Por exemplo, um agente que possui 5 litros (Q) de combustível (r). Esse recurso geraria uma tupla

$$\langle combustivel, 5 \rangle$$

Para cada norma e desejo é criado um objetivo relacionado ao estado que o agente almeja alcançar. Nesse ponto, é importante esclarecer que uma norma de proibição é equivalente a uma norma de obrigação de não atingir um estado específico (HANSSON, 1971). Na fórmula 4.1 é representada essa equivalência. Onde $P(\cdot)$ é uma proibição, $O(\cdot)$ é uma obrigação e ϕ é um estado. Nesse trabalho, por utilizarmos o uma arquitetura BDI os estados serão representados por objetivos.

$$P(\phi) = O(not\ \phi) \tag{4.1}$$

Os objetivos criados são avaliados pelo processo de deliberação baseado nos recursos que consomem, buscando uma maximização da recompensa do agente. O fruto da deliberação é um conjunto chamado de objetivos ativos, composto por objetivos não conflitantes e dentro das limitações de recurso que guiará o comportamento do agente. Os objetivos ativos são traduzidos pelo processo de *means-end* nas ações que o agente irá tomar no ambiente.

Outro elemento presente na arquitetura é o processo de percepção. Esse processo tem como finalidade identificar os recursos e normas presentes no ambiente e internalizá-los na mente do agente. Tal processo não foi aprofundado neste trabalho, portanto a percepção é apenas um mapeamento direto do que está no ambiente.

A arquitetura proposta do ponto de vista da capacidade de representação da informação é semelhante às arquiteturas N-BDI encontradas nos trabalhos relacionados. O modelo apresentado assim como os modelos N-BDI observados são capazes de representar as normas, as crenças, os desejos e intenções. O diferencial dessa arquitetura é a capacidade de representar os recursos, o que permite deliberar sobre as normas e desejos através dos objetivos gerados sem extrapolar os recursos disponíveis.

4.2 PROCESSO DE DELIBERAÇÃO

Nesta seção é abordado o processo de deliberação necessário para compor um raciocínio normativo. O primeiro passo para conduzir a elaboração do processo de deliberação foi estudar conceitos psicológicos, para definir quais elementos compõem a deliberação. Essa pesquisa nos levou ao biopsicólogo Thayer, que trabalha com o efeito do ânimo no comportamento humano (THAYER, 1989), nela é tratada a relação entre energias e tensões do indivíduo (THAYER, 1996).

No modelo proposto, o estado de ânimo é visto como um espaço n-dimensional, no qual cada uma das dimensões representa um elemento energético. Essas dimensões são limitadas pela quantidade de recursos. Por exemplo, um agente que possui 3 horas de bateria, possui um espaço unidimensional de tamanho 3, que será utilizado para maximizar a recompensa. A tensão, por sua vez, é a variável na qual o indivíduo representa o stress de um objetivo, caracterizado pelo dano que ele pode causar por não ser cumprido.

O processo de deliberação proposto, representado na figura 6, tem como objetivo deliberar sobre desejos, normas e recursos. Este processo tem como entrada os desejos, normas e os recursos. Os desejos são de ordem interna do agente, as normas são de ordem social e os recursos são os elementos fornecidos pelo ambiente. Essa entrada é traduzida em um conjunto de objetivos. Os objetivos são analisados por um processo de otimização do benefício obtido pelo agente, levando em consideração os recursos demandados. Como resultado, é obtido um conjunto de objetivos e normas não conflitantes que serão cumpridos (conjunto M). Esse conjunto é recalculado toda vez que é necessário uma revisão de objetivos e normas. Os eventos que disparam esse processo de deliberação são três: expiração, cumprimento e percepção.

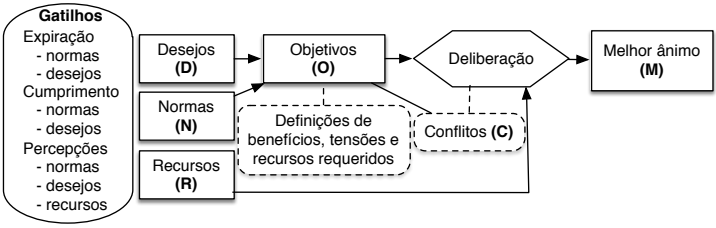

Figura 6 – Processo de deliberação do raciocínio normativo baseado em ânimo

O primeiro evento que tem potencial de disparar esse processo é a **expiração** de um prazo de uma norma ou desejo. Quando o desejo ou norma atinge seu prazo, ele não é mais viável para o estado

atual do agente. Nessa situação, o processo de deliberação é disparado, revisando os objetivos e formando um novo conjunto de objetivos e normas a serem cumpridos.

O segundo evento que dispara o processo de deliberação é o **cumprimento** de uma norma ou desejo. Quando uma norma ou desejo é completado, esse deixa o conjunto M e dispara o processo de deliberação.

O terceiro evento que dispara o processo de deliberação é a **percepção** de normas, desejos ou recursos pelo agente. Nessa situação, o processo é disparado, formando um novo conjunto de objetivos a serem cumpridos.

O processo de deliberação pode ser divido em duas fases. Na primeira fase, o agente **considera as normas, desejos e recursos**. Na segunda, baseado nas características dos objetivos, é **selecionado o conjunto de objetivos** que melhor beneficie o agente.

4.2.1 Considerando normas, desejos e recursos

Nesse momento do processo de deliberação, para cada norma e desejo são criados objetivos. Esses objetivos formam o conjunto O, que contém todos os objetivos potenciais do agente. Quando gerado um objetivo, o agente necessita identificar qual o **benefício**, a **tensão** e quais os **recursos necessários** para cumpri-lo. Dessa maneira, futuramente será possível deliberar sobre os objetivos baseado em seu benefício e consumo de recursos.

O **benefício**(bft) obtido pelo cumprimento de um objetivo corresponde ao que o agente ganha por cumprir um objetivo. O benefício é uma função que retorna um valor entre 0 e 1 que corresponde ao ganho do agente. O tipo desta função é definido em (4.2).

$$bft : O \rightarrow [0..1] \tag{4.2}$$

O benefício é baseado na recompensa quando o objetivo é originado de uma norma, na intensidade quando originado de um desejo e quando tem origem em ambos é baseado na soma da recompensa

e da intensidade. A forma genérica do benefício é definida na expressão (4.3), na qual, $reward(o)$ retorna a recompensa da norma e $intensity(o)$ retorna a intensidade do desejo.

$$bft(o) = \begin{cases} reward(o) & \text{se } o \text{ é ger. por uma norma} \\ intensity(o) & \text{se } o \text{ é ger. por um desejo} \\ reward(o) + intensity(o) & \text{se } o \text{ é ger. pelos dois} \end{cases} \tag{4.3}$$

Nessa proposta, a **perda** que o agente sofre por não cumprir um objetivo é representada pela tensão (THAYER, 1989). A tensão é representada por uma função que retorna um valor real correspondente à tensão do objetivo, como apresentado em (4.4).

$$tns : O \rightarrow \mathbb{R}^{+} \tag{4.4}$$

A **tensão** pode ser composta de três maneiras. Caso a origem do objetivo seja uma norma, o custo da **sanção** corresponderá à tensão. Caso a origem seja um desejo, a tensão corresponde à **necessidade** do agente satisfazer esse desejo. Por fim, caso o objetivo tenha origem em ambos, a tensão é a soma dos dois valores. De forma genérica, pode-se determinar a tensão como:

$$tns(o) = \begin{cases} sanction(o) & \text{se } o \text{ é ger. por uma norma} \\ necessity(o) & \text{se } o \text{ é ger. por um desejo} \\ sanction(o) + necessity(o) & \text{se } o \text{ é ger. pelos dois} \end{cases} \tag{4.5}$$

Conforme apresentado anteriormente, o recurso pode ser considerado uma energia. Como um agente pode depender de diferentes tipos de recursos, esses formam um espaço $|R| - dimensional$, no qual cada dimensão corresponde a um tipo de recurso r, tal que $r \in R$. Sendo R o conjunto de todos os tipos de recurso. Dessa maneira, a função rr retorna um valor real correspondente à quantidade de recursos requeridos para atingir um objetivo, como apresentado em (4.6). De uma forma genérica, a função de recurso requerido tem uma assinatura $rr(o, r)$ e retorna o quanto recurso r é requerido para o objetivo

o.

$$rr : O \times R \rightarrow \mathbb{R}^{+} \tag{4.6}$$

Por exemplo, uma norma que obriga o agente a atingir o objetivo o, tem uma sanção 100 e uma recompensa 150; contudo, para ser cumprida, consome 5 unidades de combustível. Obtemos uma $rr(o, combustivel) = 5$, uma $tns(o) = 100$ e um $bft(o_1) = 150$.

4.2.2 Selecionando objetivos

Na segunda fase, o agente delibera sobre a aceitação dos objetivos contidos no conjunto O. Para tanto, o agente deve tratar os conflitos de objetivos e de recursos que podem aparecer durante a sua existência.

Para tratar dos conflitos, a grande parte dos trabalhos usa critérios de prioridade. Logo, uma norma ou objetivos ditos mais prioritários sempre serão executados mesmo que sub-utilize a capacidade do agente. A proposta desse trabalho é maximizar o benefício do agente, escolhendo objetivos não conflitantes que melhor contemple o benefício, observando os recursos disponíveis.

A escolha desse conjunto de objetivos não conflitantes pode ser mapeada para um problema da mochila com múltipla escolha (KELLERER; PFERSCHY; PISINGER, 2004). Nesse problema, os itens da mochila são divididos em classes, das quais, no máximo, um item de cada classe poderá ser adicionado na mochila. Por exemplo, os itens tocha, lanterna, vela, canivete e barraca estão classificados em três classes: iluminação (tocha, lanterna e vela), corte(canivete) e moradia (barraca). Através do problema da mochila com múltipla escolha, será impossível que a mochila contenha a tocha e a lanterna e a vela simultaneamente, liberando espaço para itens das outras classes.

No Huginn, cada classe c é um conjunto de objetivos conflitantes e cada objetivo pertence a uma ou mais classes. Um objetivo que não conflita com nenhum outro pertence a uma classe apenas com ele mesmo. Com essas classes e a restrição da expressão (4.7) é garantido que, no máximo, um elemento de cada classe pode ser escolhido. Nessa

restrição, m_o é a variável de decisão binária do objetivo o. Quando o valor de m_o é 1, o objetivo o é selecionado e, quando o valor é 0, o objetivo não é selecionado. O conjunto de classes C é um subconjunto do conjunto potência de O representado na expressão (4.8).

$$\sum_{o \in c} m_o \leq 1, c \in C \tag{4.7}$$

$$C \subseteq 2^O \tag{4.8}$$

Por exemplo, os objetivos p, $\neg p$ e q são divididos em duas classes, representados na expressão (4.9). Os objetivos p e $\neg p$ estão na mesma classe, porque eles estão em conflito. Por outro lado, o objetivo q está em outra classe, pois esta não conflita com os anteriores.

$$C = \{\{\neg p, p\}, \{q\}\} \tag{4.9}$$

Para que o problema da mochila com múltipla escolha seja utilizado a fim de decidir a aceitação e resolução de conflitos de recursos, é necessário modificar a restrição do problema. No problema da mochila, a restrição se resume a não ultrapassar a sua capacidade. No caso da aceitação e resolução de conflitos, existe uma série de limitações criadas pelo conjunto de recursos. Dessa maneira, cada recurso gera uma nova restrição ao problema. Para cada um dos recursos será gerado uma restrição como a equação (4.10), na qual, $ar(r)$ retorna quanto recurso r o agente tem.

$$\sum_{c \in C} \sum_{o \in c} (m_o rr(o, r)) \leq ar(r), \ \ r \in R \tag{4.10}$$

A definição completa do problema gerado é representada pela expressão (4.11). Nessa formulação do problema, é possível observar que a função a ser maximizada é dada pelo soma do *benefício* mais a *tensão*; porque, se o agente satisfaz o desejo ou a norma, ele recebe o benefício e evita a tensão. Por exemplo, quando o objetivo o é completado o benefício do agente é 0.5 e quando o objetivo o não é completado, a perda é a tensão é 0.4. Logo, se o agente cumpre o objetivo, ele ganha 0.5 e, *não perde* 0.4. Dessa maneira, o ânimo gerado pelo objetivo o é

0.9.

$$Maximize \sum_{c \in C} \sum_{o \in c} (m_o(bft(o) + tns(o))) \tag{4.11}$$

Subject to

$$\sum_{c \in C} \sum_{o \in c} (m_o rr(o,r)) \leq ar(r), r \in R$$

$$\sum_{o \in c} m_o \leq 1, c \in C$$

Como resultado dessa fase, é obtido um conjunto de objetivos ativos (conjunto M) que serão cumpridos pelo agente até que um evento dispare novamente o processo de deliberação. Esse conjunto M definido pela expressão (4.12).

$$M = \{o | m_o = 1\} \tag{4.12}$$

4.2.3 Exemplo

Para ilustrar o modelo de raciocínio utilizaremos o exemplo a seguir. Considerando um agente com 10 litros de combustível em seu tanque ($ar = 10$). Esse agente percebe e aceita um conjunto de normas, compostos por:

- não atravessar o rio ($\neg r$) durante o alagamento (a), com uma sanção de 2 mil reais e uma recompensa de 3 mil reais. $N_1 = \langle O, \neg r, agente1, a, \neg a, 2, 3 \rangle$;
- atravessar o rio (r), durante um resgate (rgt), com uma sanção de 2 mil reais e uma recompensa de 7 mil reais. $N_2 = \langle O, r, agente1, rgt, \neg rgt, 2, 7 \rangle$;
- proteger os civis (p) durante o alagamento (a), com uma sanção de mil reais e uma recompensa de 2. $N_3 = \langle O, p, agente1, a, \neg a, 1, 2 \rangle$.

Além do conjunto de normas, o agente possui o desejo, que nunca expira ($E = false$), de achar uma posição segura (q) com uma intensidade 2 e uma necessidade 3. $D_1 = \langle q, false, 2, 3 \rangle$.

No primeiro momento, considerando que as condições de ativação alagamento (a) e resgate (rgt) são verdadeiras, o agente percebe um conjunto de normas e desejos contendo todas as normas e desejos $\{N_1, N_2, N_3, D_1\}$ e os interpreta como objetivos do conjunto O. Nesse momento, são calculados o **benefício, a tensão e os recursos requeridos** dos novos objetivos. Para definir os recursos requeridos, o agente verifica a quantidade de cada recurso necessária para cumprir os objetivos.

- não atravessar o rio ($\neg r$) custa ao agente 3 litros de combustível: $rr(\neg r, combustivel) = 3$;
- atravessar o rio (r) custa ao agente 10 litros de combustível: $rr(r, combustivel) = 10$;
- proteger os civis (p) custa ao agente 8 litros de combustível: $rr(p, combustivel) = 8$;
- achar uma posição segura (q) custa ao agente 7 litros de combustível: $rr(p, combustivel) = 7$.

A tensão é formada pela formulação proposta na equação (4.4). Para esse conjunto é obtido:

- não atravessar o rio ($\neg r$) tem sanção de 2 mil: $tns(\neg r) = 0.2$;
- atravessar o rio (r) tem sanção de 2 mil:: $tns(r) = 0.2$;
- proteger os civis (p) tem sanção 1 mil: $tns(p) = 0.1$;
- achar uma posição segura (q) necessidade 3: $tns(q) = 0.3$.

O benefício é formado pela formulação proposta na equação (4.3). Para esse conjunto é obtido:

- não atravessar o rio ($\neg r$) tem recompensa 3 mil: $bft(\neg r) = 0.3$;
- atravessar o rio (r) tem recompensa 7 mil: $bft(r) = 0.7$;

- proteger os civis (p) tem recompensa 2 mil: $bft(q) = 0.2$;
- achar uma posição segura (q) tem intensidade 3 mil: $bft(p) = 0.3$.

Com esses dados, é formado o conjunto O representado na tabela 2.

Or.	Obj.	Ânimo	R. requerido
N_1	$\neg r$	$bft(\neg r) + tns(\neg r) = 0.2 + 0.3 = 0.5$	3
N_2	r	$bft(r) + tns(r) = 0.2 + 0.7 = 0.9$	10
N_3	p	$bft(p) + tns(p) = 0.1 + 0.2 = 0.3$	8
D_1	q	$bft(q) + tns(q) = 0.3 + 0.3 = 0.6$	7

Tabela 2 – Estado inicial do conjunto O

O conjunto $O = \{\neg r, r, p, q\}$ é dividido em três classes de conflito, como representado na expressão(4.13).

$$C = \{\{\neg r, r\}, \{p\}, \{q\}\} \tag{4.13}$$

Ao final do processo de otimização o estado das variáveis de decisão são: $m_{\neg r} = 1$, $m_r = 0$, $m_p = 0$, $m_q = 1$. Assim, gerando um conjunto $M = \{\neg r, q\}$, o que significa que o agente não vai atravessar o rio e vai procurar um local seguro. Destaca-se que apesar de o objetivo r possuir o maior benefício, ele não foi selecionado, pois o conjunto M gerado é capaz de, com o mesmo recurso, conquistar um benefício maior.

Após alguns ciclos de execução, o agente percebe uma nova norma (N_4), na qual o agente tem que auxiliar outro agente (s) com recompensa 6 mil, sanção 2 mil e custo energético de 2 litros. Essa nova norma dispara o processo de deliberação de objetivos, normas e recursos. Como alguns **recursos já foram consumidos** pelos objetivos $\neg r$ e q, eles têm o valor do **recurso requerido** recalculado. O objetivo $\neg r$ consumiu 1 litro de combustível e q consumiu 2. Logo, o recurso disponível para o agente passa a ser $ar(combustível) = 10 - 3 = 7$. Essa situação vai gerar um novo conjunto O:

Após executar o processo de deliberação, obtemos um novo conjunto $M = \{q, s\}$, ou seja, o agente vai procurar um local seguro e au-

Or.	Obj.	ânimo	Recurso requerido
N_1	$\neg r$	0.5	2
N_2	r	0.9	10
N_3	p	0.3	8
D_1	q	0.6	5
N_4	s	0.8	2

Tabela 3 – Estado do conjunto O após a adição de N_4

xiliar outro agente. Nesse exemplo, observamos que apesar de o agente já ter consumido recursos para cumprir $\neg r$, ele achou mais vantajoso abandoná-lo e assumir s, buscando um benefício maior.

4.2.4 Discussão

Como a tabela 4 apresenta, o processo de deliberação proposto possui poucos aspectos similares como os trabalhos relacionados. Os trabalhos relacionados não utilizam os recursos como elemento de avaliação no processo de deliberação, elemento importante em uma situação na qual os recursos são finitos. O processo de deliberação da proposta não leva só em consideração os recursos, como considera a sanção e a recompensa de uma norma. Outra contribuição é o método de deliberação, capaz de encontrar o conjunto que maximiza o benefício do agente.

O modelo proposto apresenta uma limitação, na qual não é capaz de tratar objetivos obtidos a partir de outros agentes. Isso ocorre porque o agente não é capaz de mensurar qual o benefício de um objetivo obtido através de requisições de outros agentes. Para tal, trabalhos como Carabelea, Boissier e Castelfranchi (2005) e Kollingbaum e Norman (2006) utilizam o conceito de poder social (CASTELFRANCHI, 2002). Através desse conceito, é possível o agente determinar o quanto é importante a relação com outro agente e, assim, definir o benefício dos objetivos solicitados por ele. Esse problema pode ser um encaminhamento de trabalho futuro, no qual o agente delibera não somente

Modelo	Pref.	Sanção	Deadline	Depend.	Recursos
Dignum	X				
Boella	X				
BOID	X				
Carabelea				X	
López		X			
Noa	X		X	X	
N. KPG	X				
Gaertner	X				
Criado		X			
Alechina	X		X		
Huginn		X			X

Tabela 4 – Modelos X critérios de avaliação

sobre os objetivos pessoais e normas, mas também sobre os objetivos recebidos de outros agentes.

Uma característica bem evidenciada do modelo proposto é o fato de não utilizar sistema de preferência, a qual é adotada pela maioria dos modelos. O fato de não utilizar um sistema de preferência faz com que o modelo proposto não consiga priorizar um objetivo em relação a outro. Nessa proposta, é possível estabelecer uma relação de prioridade parcial, atribuindo um valor alto para o benefício de um objetivo, o que lhe dá uma probabilidade maior de ser cumprido. Contudo, não há garantia de que o objetivo com maior benefício será cumprido. O fato de não utilizar um sistema de preferência permite obter o melhor benefício do agente, pois os objetivos deixam de ser avaliados individualmente para serem avaliados pelo benefício de um conjunto de objetivos.

Dos trabalhos relacionados, o mais relevante deles é o Alechina, Dastani e Logan (2012), tendo como principal destaque o tratamento dos deadlines dos objetivos. No modelo proposto nesta tese, não é possível tratar o deadline de um objetivo. O que é possível representar, através do modelo proposto, é o tempo na forma de um recurso,

o que permite formar um conjunto de objetivos que será cumprido em um tempo máximo, contudo sem garantia de cumprimento de deadline. Assim como o trabalho de Alechina, Dastani e Logan (2012), o modelo proposto tem problemas em representar o tempo, pois as linguagens utilizadas possuem características não determinísticas, que não permitem precisar o tempo de execução de uma ação. O modelo proposto é adequado para representar recursos finitos bem definidos, como brocas e filtros. Por exemplo, um objetivo tem como estado final 1 litro de água pura. Para purificar 1L de água o agente consumirá 1 filtro de carvão. Dessa maneira, as energias são bem definidas e possíveis de implementar.

Elementos como tempo e carga de bateria, podem ser modelados, mas é necessário estimar um valor seguro[2] para o cumprimento do objetivo, além de utilizar uma política de reciclagem de recursos excedentes. Por exemplo, o objetivo x consome 30% da carga de uma bateria, mas, eventualmente, o objetivo é cumprido gastando somente 20% da carga, os 10% da carga podem ser liberados para o cumprimento de outro objetivo, quando x for cumprido.

A principal desvantagem apresentada por esse processo é a incapacidade de detectar os conflitos. Existe uma vertente nova representada por Gaertner e Toni (2008) e Joseph et al. (2010) a qual trabalha com o conceito de coerência que pode ser um caminho para solucionar esse tipo de problema.

4.3 RESUMO DO CAPÍTULO

Nesse capítulo, foram apresentados uma arquitetura de agente e um processo de deliberação sobre normas e objetivos baseado em ânimo. A primeira seção destacou a arquitetura do agente. A segunda destacou o processo de deliberação.

A arquitetura proposta do ponto de vista da capacidade de representação da informação é semelhante às arquiteturas N-BDI encontradas nos trabalhos relacionados que representam normas, crenças,

[2] Valor superestimado de recurso consumido para o cumprimento do objetivo.

desejos e intenções. O diferencial dessa arquitetura é a capacidade de representar os recursos, o que permite deliberar sobre as normas e desejos através dos objetivos gerados sem extrapolar os recursos disponíveis.

5 IMPLEMENTAÇÃO DO HUGINN

> "A experiência começa de maneira indireta na teoria e se consagra na prática."
>
> Juahrez Alves

Este capítulo apresenta uma implementação do Huginn através da extensão da plataforma Jason (BORDINI; HÜBNER; WOOLDRIDGE, 2007). A escolha pelo Jason foi motivada pela sua consolidação na área, sendo referência constante na área acadêmica para construção de agentes BDI (Belief - Desire - Intention).

A descrição da proposta de implementação ocorrerá em duas partes. As alterações na arquitetura do agente para que o processo de deliberação do Huginn seja possível (subseção 5.1) e a representação das informações requeridas pelo Huginn (subseção 5.2). Ao final desse capítulo há uma avaliação da existência de gargalos na implementação (subseção 5.3).

5.1 EXTENSÃO DA ARQUITETURA DE AGENTE DO JASON

A implementação produzida estende a arquitetura de agente da plataforma Jason (BORDINI; HÜBNER; WOOLDRIDGE, 2007). O funcionamento do Huginn é baseado em gatilhos que disparam o processo de deliberação normativa. No Jason, toda adição e remoção de crença e objetivo gera um evento. Através desses eventos, é possível implementar os gatilhos do processo de deliberação do Huginn. A função $selectEvent$ (S_E), na arquitetura do Jason, é a função onde todos os eventos gerados são selecionados. O Huginn reescreve a função $selectEvent$ identificando os gatilhos e deliberando sobre a seleção, a remoção, a suspensão e a retomada dos objetivos Huginn. Para tanto, a arquitetura do Jason foi estendida e representada na figura 7. O losango preto representa a alteração da função $selectEvent$ e o retângulo preto $Goals$, a estrutura de dados que representa o conjunto O.

Na função $selectEvent$ proposta, os eventos recebido como en-

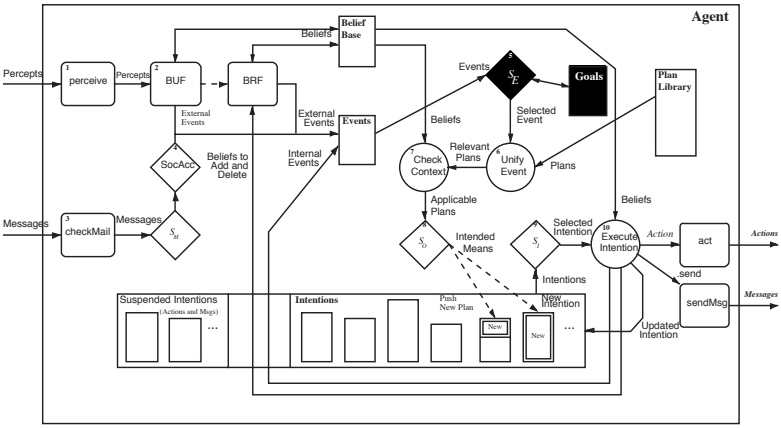

Figura 7 – Arquitetura de agente na plataforma do Huginn

trada são avaliados e selecionados conforme o critério do Huginn (não ultrapassar os recursos disponíveis e obter o maior ganho). Para que isso ocorra, os eventos relacionados aos objetivos são armazenados em uma estrutura paralela representando o conjunto de objetivos O. Este conjunto é representado na figura 7 pelo retângulo Goals em destaque. Para cada objetivo são armazenadas as seguintes informações: descrição do objetivo; origem (desejo, obrigação ou proibição); condição de expiração; lista de recursos necessários para cumprir o objetivo; benefício; tensão; evento referente ao objetivo; comprometimento do agente (Verdadeiro para objetivo ativo, falso para objetivo inativo).

Com base nos conjuntos de objetivos e recursos, a função *select event* segue o algoritmo 2. Esse algoritmo está dividido em três partes. A primeira (linhas 2 a 7) procura na lista de eventos (F_i) por novos objetivos gerados por normas ou desejos. Quando um novo objetivo é encontrado, ele é adicionado ao conjunto O. A segunda parte (linhas 8 a 12) desse algoritmo é a busca e remoção de objetivos expirados no conjunto O. Na terceira (linha 13 a 18) caso seja necessário uma nova deliberação, o processo de deliberação ($deliberation(O, C, R)$) produz um novo conjunto M e para cada objetivo $m \in M$ é retomado e os

objetivos não selecionados ($o \in \{O - M\}$) são suspensos.

Algorithm 2: Função selectEvent

Input: Fila de Eventos F_i
Output: Fila de Eventos filtradas pelos objetivos ativos F_m

```
newDeliberation=false;
F_m = F_i;
for each (f ∈ F_i) do
    if f_o ∉ O then
        O := O ∪ {f_o};
        newDeliberation=true;
    if f_r é adição de recursos then
        newDeliberation=true;
for each (o ∈ O) do
    if condição de expiração de o é verdadeira then
        O := O \ {o};
        newDeliberation=true;
if newDeliberation then
    M := deliberation(O,C,R) ;
    for each (f ∈ F_m) do
        if f_o ∉ M then
            F_m = F_m \ {f_o};
    for each (o ∈ M) do
        if o_f ∉ F_m then
            F_m := F_m ∪ {o_f};
return F_m
```

O processo de deliberação normativa ($deliberation(O,C,R)$) traduz os objetivos (O), conflitos (C) e recursos (R) em um problema da mochila multidimensional com múltipla escolha, como descrito no capítulo 4. O problema de otimização descrito é uma subclasse da

programação inteira mista (MIP - mixed integer programming) conhecida como programação inteira binária. Os problemas do tipo MIP - binário são problemas de programação linear, nos quais as variáveis de decisão são binárias. Para encontrar a solução desse problema, foi utilizado o solver **GLPK**. O GLPK[1] (GNU Linear Programming Kit) é um conjunto de rotinas escritas na linguagem de programação C e organizadas na forma de uma biblioteca. A intenção dessa biblioteca é resolver problemas de programação linear (LP) e programação inteira mista (MIP). Ao utilizar o GLPK, garantimos que o processo encontre sempre a solução ótima, neste caso, o conjunto de objetivos que gera o melhor ânimo.

Existem muitos algoritmos que resolvem esse problema de otimização (KELLERER; PFERSCHY; PISINGER, 2004), com diferentes complexidades e precisões. O algoritmo adotado pode variar de acordo com os requisitos da aplicação a ser desenvolvida. Portanto, criou-se uma arquitetura que permite ao programador substituir o algoritmo de deliberação (neste caso o GLPK) por um que se adeque a sua aplicação.

5.2 EXPRESSANDO RECURSOS, TENSÕES E BENEFÍCIOS

Os três eixos que dão suporte ao processo de deliberação do Huginn são os desejos, as normas e os recursos. Nesta abordagem será considerado que cada agente percebe as normas e os recursos do ambiente. Os desejos, por sua vez, são de natureza interna do agente.

Os recursos são considerados de uso pessoal do agente e, portanto, os agentes não compartilham os mesmos recursos. Para representar os recursos disponíveis, o agente crenças na forma da expressão (5.1), na qual, r é o tipo do recurso e q é a quantidade. Os recursos são observados pela na arquitetura no momento da deliberação, para determinar quais objetivos serão assumidos.

$$huginn\_\_\_resource(r, q) \tag{5.1}$$

[1] http://www.gnu.org/software/glpk/

As normas também são representadas internamente na forma de crenças (5.2). Essa norma representa, na ordem, o objetivo implicado pela norma, a condição de expiração, a sanção, a recompensa e o tipo da norma (*p* para proibição e *o* para obrigação). A condição de ativação não está explícita, pois consideramos que a norma só pode ser percebida quando estiver ativa.

$$norm(goal, expiration, saction, reward, normType) \tag{5.2}$$

Os desejos definidos no Huginn são de natureza interna do agente e, portanto, representados por crenças que identificam o objetivo desejado, a condição de expiração, a necessidade e a intensidade do desejo. A assinatura dessa crença é dada por (5.3). Essa crença possui quatro termos: *goal* representando o objetivo, *expiration* representando a condição de expiração, *necessity* é o valor da necessidade e *intensity* é o valor da intensidade.

$$desire(goal, expiration, necessity, intensity) \tag{5.3}$$

Além dos três eixos do processo de deliberação, o Huginn especifica que os objetivos possuem três valores associados: a tensão, o benefício e os recursos necessários. Conforme descrito no capítulo 4, a tensão está relacionada à sanção da norma e a necessidade do desejo e o benefício está relacionado à recompensa da norma e a intensidade do desejo.

No Huginn, a definição de recurso requerido para atender um objetivo é dada por uma função. O recurso requerido para atingir um objetivo depende do estado atual do agente. Por exemplo, se o agente tem o objetivo de ir para a posição 7 e o mesmo se encontra na posição 1, o consumo de recurso será maior do que se ele estiver na posição 5. Na implementação proposta foi utilizado o motor de regras do Jason para que o desenvolvedor pudesse definir a dinâmica do consumo dos recursos. Essa regra possui três termos em sua cabeça: *goal*, representando o objetivo; *resourcetype*, representando o tipo do recurso e a regra deve definir em *quantity* a quantidade de recurso

resourcetype ainda necessário para atingir o objetivo *goal*. A cabeça da regra é representada em (5.4).

$$huginn\_\_resource\_required(goal, resourcetype, quantity) \quad (5.4)$$

No momento da deliberação, essas regras serão utilizadas para definir quanto recurso será necessário para cumprir o objetivo. Caso a função não seja definida, a deliberação considera o gasto do recurso é igual a 0.

Por exemplo, o agente se move sobre um tabuleiro e a cada casa visitada consome 2L de gasolina. Esse agente pode se mover nas 8 direções. A regra (5.5) representa a dinâmica do recurso gasolina para o objetivo $goto(X1, Y1)$

$$\begin{aligned} &huginn\_\_resource\_required(goto(X1, Y1), battery, B) \quad (5.5)\\ &\quad :- position\ (X0, Y0)\&\\ &\quad B = math.max(math.abs(X1 - X0), math.abs(Y1 - Y0)) * 2. \end{aligned}$$

O Huginn traduz as normas e desejos conhecidos pelo agente em objetivos. Para permitir que o desenvolvedor possa facilmente personalizar o agente de acordo com as necessidades do projeto, optou-se por deixar explícito a tradução das normas e desejos em objetivos.

O plano do trecho de código 5.1 apresenta um plano genérico para tradução de normas do tipo obrigação em objetivos. Esse plano é apresentado na linguagem Jason e representa que toda percepção de norma irá criar um objetivo `GOAL` com as seguintes anotações:

- o identificador Huginn (`huginn_goal`) sinaliza a função S_E que esse objetivo é tratado pelo Huginn;
- a condição de expiração (`cexp`) contendo a expiração da norma (`EXPI RATION`);
- a tensão (`tension`) contendo o valor da sanção (`SANCTION`);
- o benefício (`benefit`) contendo o valor da recompensa (`REWARD`);

- a origem (`typesource`) contendo o tipo da norma, nesse caso *o*.

```
1    +norm(GOAL, EXPIRATION, SANCTION, REWARD, o): true
2     <-
3        !GOAL[ huginn_goal, cexp(EXPIRATION),
4               tension(SANCTION), benefit (REWARD),
5               typesource(o)].
```

Trecho de código 5.1 – Plano genérico para tradução de normas em objetivos

O plano do trecho de código 5.2 exemplifica um caso específico de como as normas do tipo obrigação são traduzidas em objetivos. Nesse plano, o agente tem a obrigação de obter pedras até que não haja mais pedras disponíveis, ou o mesmo possua 5 pedras. Caso não a cumpra, ele sofre uma sanção 0,5. Por outro lado, cumprindo-a, o agente ganha uma recompensa de 0,4. Como é previsto no Huginn, a norma é interpretada como um objetivo com as seguintes propriedades: condição de expiração de não ter mais pedras, tensão de 0,5 e benefício de 0,4.

```
1   +norm(gather(stone), ~stones, 0.5, 0.4, o): true
2     <-
3        !gather(stone)[ huginn_goal, cexp(not stones),
4                        tension(0.5), benefit (0.4),
5                        typesource(o)].
```

Trecho de código 5.2 – Exemplo de percepção de norma

O plano do trecho de código 5.3 apresenta um plano genérico para tradução de normas do tipo proibição em objetivos. Esse plano é apresentado na linguagem Jason e representa que toda percepção de norma irá criar um objetivo `GOAL` com as seguintes anotações:

- o identificador Huginn (`huginn_prohibition`) sinaliza a função S_E que esse objetivo gerado por uma proibição e deve ser tratado pelo Huginn;

- a condição de expiração (`cexp`) contendo a expiração da norma (`EXPI RATION`);
- a tensão (`tension`) contendo o valor da sanção (`SANCTION`);
- o benefício (`benefit`) contendo o valor da recompensa (`REWARD`);
- a origem (`typesource`) contendo o tipo da norma, nesse caso *p*.

```
1   +norm(GOAL, EXPIRATION, SANCTION, REWARD, p): true
2    <-
3       !GOAL[ huginn_prohibition, cexp(EXPIRATION),
4              tension(SANCTION), benefit (REWARD),
5              typesource(p)].
```

Trecho de código 5.3 – Plano genérico para tradução de normas em objetivos

O plano do trecho de código 5.4 exemplifica um caso específico de como as normas do tipo proibição são traduzidas em objetivos. Nesse plano, o agente está proibido de coletar pedras até que a estação base esteja operacional. Caso não a cumpra, ele sofre uma sanção 0,9. Por outro lado, cumprindo-a, o agente não ganha recompensa. Como é previsto no Huginn, a norma é interpretada como um objetivo com as seguintes propriedades: condição de expiração de a estação base estar operacional, tensão de 0,9 e benefício de 0,0.

```
1   +norm(gather(stone), station_status(ready), 0.9, 0.0, p): true
2     <-
3       !gather(stone)[ huginn_prohibition,
4                       cexp(station_status(ready)),
5                       tension(0.9), benefit (0.0),
6                       typesource(p)].
```

Trecho de código 5.4 – Exemplo de percepção de norma

Os desejos são traduzidos de forma similar às normas. O plano do trecho de código 5.5 apresenta um plano genérico para tradução de

desejos em objetivos. Esse plano define que, para toda percepção de desejo, será calculada a quantidade de recurso requerido para atingir o objetivo, representado nas linhas 3 e 4, criando um objetivo `GOAL` com as seguintes anotações:

- o identificador Huginn (`huginn_goal`) sinaliza a função S_E que esse objetivo é tratado pelo Huginn;
- a condição de expiração (`cexp`) contendo a expiração do desejo (`EXPI RATION`);
- a tensão (`tension`) contendo o valor da sanção (`NECESSITY`);
- o benefício (`benefit`) contendo o valor da recompensa (`INTENSITY`);
- a origem (`typesource`) informando que é um desejo (`d`);

```
1  +desire(GOAL, EXPIRATION, NECESSITY, INTENSITY): true
2     <-
3        !GOAL[ huginn_goal, cexp(EXPIRATION),
4               tension(NECESSITY), benefit (INTENSITY),
5               typesource(d)].
```

Trecho de código 5.5 – Plano genérico para tradução de desejos em objetivos

O plano do trecho de código 5.6 é um caso específico de como os desejos são traduzidos em objetivos. Neste trecho o agente tem um desejo de cozinhar carne até que esteja satisfeito. A intensidade desse desejo é 0,3 e a necessidade é 0,8. Como é previsto no Huginn, o desejo é interpretado como um objetivo com as seguintes propriedades: condição de expiração igual a satisfeito, tensão de 0,8 e benefício de 0,3. Assim como no caso das normas, o desenvolvedor pode criar um conjunto de regras para definir benefícios e tensões.

```
1  +desire(cook(meat), satisfied, 0.8, 0.3): true
2     <-
3        !cook(meat)[cexp(satisfied), tension(0.8)
4                    benefit (0.3), typesource(d)].
```

Trecho de código 5.6 – Exemplo de percepção de desejo

Outra característica do modelo Huginn é o tratamento de objetivos que não podem ser atingidos ao mesmo tempo. O modelo não especifica como esses conflitos entre objetivos são detectados. Considerando como premissa que existem trabalhos que tratam da detecção de conflitos (JOSEPH et al., 2010; THAGARD, 2002), esta proposta de implementação não irá determinar um mecanismo específico para detectar conflitos. A proposta formaliza o conflito como uma crença que relaciona dois objetivos. Um exemplo, comum na vida de um doutorando é o conflito entre o objetivo de ir para praia ($goto(beach)$) e trabalhar na tese ($work(thesis)$), representado no trecho de código (5.6).

$$huginn\_\_\_conflict(goto(beach), work(thesis)). \tag{5.6}$$

5.3 VERIFICAÇÃO DE GARGALOS NA IMPLEMENTAÇÃO DO HUGINN

Ao final desse capítulo ficam questionamentos sobre a eficiência do modelo implementado. Para elucidar esses questionamentos, foi utilizado uma ferramenta de profilling (hprof). Essa ferramenta permite identificar possíveis garlgalos na execução de uma aplicação, contabilizando o tempo de CPU de cada método invocado. O Hprof foi invocado para rastrear o callback de todos os métodos até uma profundidade de quatro invocações. O rastreamento ocorreu em todo o ciclo do processador, caracterizando uma análise precisa sobre a execução do Huginn.

Na figura 8 é apresentado a árvore de invocação dos métodos, na qual, destacam-se os métodos *Huggin.select Event*, *Huggin.conflictConstraint*, *Lp.solve* e *Jason.asSemantics.Unifier.unifyTerms*. Em um

cenário com 100 normas onde o processo de deliberação, *Huggin.select Event*, é acionado frequentemente, observamos que o processamento da deliberação ocupou 95,93% do tempo de CPU de toda a aplicação. O método responsável pela resolução do problema da mochila, *Lp.solve*, mostrou-se eficiente para esse tamanho de problema não representando um gargalo para o sistema. Por outro lado, o método que constrói as restrições por conflitos, *Huggin.conflictConstraint*, mostrou-se o principal gargalo na execução do Huginn. O responsável pelo gargalo é o método utilizado para verificar as crenças que representam os conflitos, *Jason.asSemantics.Unifier.unifyTerms*. A análise feita nessa seção indica que para aumentar a escalabilidade do Huginn será necessário encontrar uma maneira mais eficiente de construir as restrições por conflito.

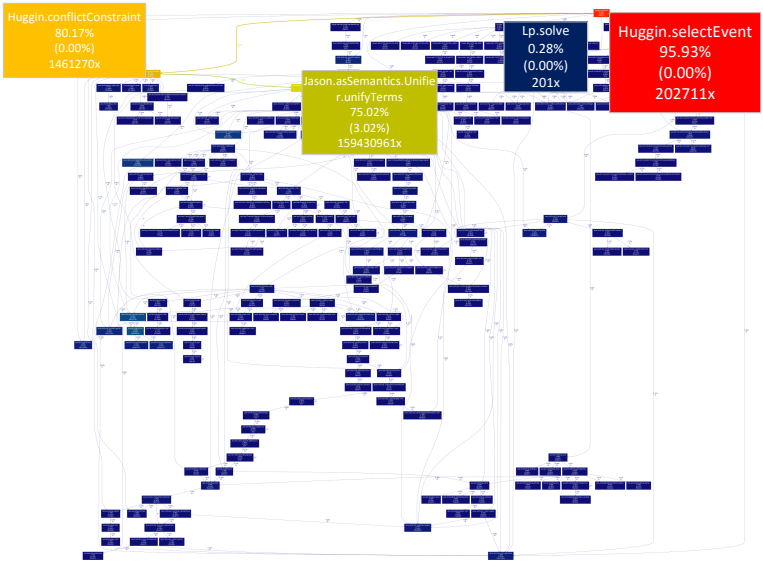

Figura 8 – Avaliação do tempo de execução de cada método

5.4 RESUMO DO CAPÍTULO

Nesse capítulo, foi apresentada uma implementação do Huginn. A primeira seção destacou a representação dos recursos, tensões e benefícios. A segunda destacou como foi estendida a arquitetura do Jason e como o foi implementado Huginn. A implementação proposta contemplou todos os conceitos propostos pelo Huginn. Por fim, a terceira seção apresentou que ainda é necessário investigar um método para tornar mais eficiente a construção das restrições por conflitos.

6 PLANEJAMENTO DE TRAJETÓRIA PARA VEÍCULO AÉREO NÃO TRIPULADO

> "De que vale o eterno criar, se a criação em nada acabar?"
>
> Mephistopheles para Fausto - Goethe

Nos últimos anos, a quantidade de aplicações para Veículos Aéreos Não-Tripulados (VANTs) tem crescido significativamente. A teoria de agentes se encaixa nesse contexto, pois confere ao VANT autonomia na realização de suas missões. O presente capítulo apresenta dois cenários de aplicação para planejamento de rota de VANTs, utilizando o Huginn.

O objetivo do primeiro cenário é apresentar uma implementação do planejamento de trajetória de um VANT, usando o Huginn, e compará-lo a um agente com a mesma função desenvolvido em Jason (SANTOS; HÜBNER; BECKER, 2015). Como visto anteriormente, o Huginn adiciona processos ao ciclo de raciocínio do agente Jason, que permitem deliberar automaticamente sobre normas, desejos e objetivos. Esse cenário mostra que a codificação desse cenário é facilitada pelo uso do Huginn.

O segundo cenário tem o objetivo de demonstrar a facilidade de adicionar um novo comportamento para o agente. O Huginn tem como característica um processo de raciocínio que permite modularizar os objetivos derivados dos desejos e normas. Para demonstrar essa característica, será comparado o modelo do agente Huginn do primeiro cenário com o do segundo.

6.1 INFRAESTRUTURA

Os cenários de planejamento de rota foram implementados na plataforma embarcada do VANT Provant[1]. O Provant é um projeto desenvolvido no departamento de automação e sistemas na Universi-

[1] http://provant.das.ufsc.br

dade Federal de Santa Catarina. O objetivo do Provant é desenvolver uma aeronave tilt-rotor open-source de pequena escala capaz de executar voos de forma autônoma, navegando por waypoints.

O Provant tem duas plataformas computacionais embarcadas. A placa Discovery é usada no nível mais baixo, atuando com os sensores e atuadores. A placa Beaglebone é usada no nível mais alto, implementando o controle do VANT e o planejamento de rota. A comunicação entre as plataformas é realizada por meio de uma porta serial.

No contexto do projeto Provant, o trabalho de Santos, Hübner e Becker (2015) apresenta o modelo de planejamento de rota representado na figura 9 e dividido em três níveis: planejamento, controle e veículo robótico. O nível do planejamento é responsável por produzir a rota. O nível do controle é responsável por definir como o agente irá do ponto A para o ponto B. Nesse nível, ocorrem a fragmentação de rota e o controle cinemático e dinâmico. O nível do veículo utiliza os valores produzidos no nível de controle para ajustar os atuadores e informar os sinais dos sensores ao nível de controle.

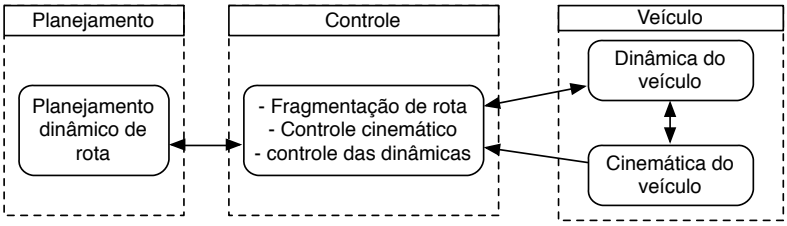

Figura 9 – Modelo de agente (SANTOS; HÜBNER; BECKER, 2015)

O modelo possui um agente Jason planejando a rota e comunicando-se com a camada de controle para informar o próximo ponto a ser visitado e a receber as informações dos sensores. A implementação de Santos, Hübner e Becker (2015) fornece três percepções e duas ações para o nível de planejamento. A percepção (6.1) representa a posição corrente do agente nos três eixos. A percepção (6.2) informa a carga da

bateria e a percepção (6.3) representa a taxa de consumo da bateria[2]. A ação (6.4) informa o ponto de referência (ponto a ser visitado) para o nível de controle. A ação (6.5) informa para o nível de controle a velocidade de referência do agente.

$$position(X, Y, Z) \tag{6.1}$$
$$charge(B) \tag{6.2}$$
$$rate(T) \tag{6.3}$$
$$goto(X, Y, Z) \tag{6.4}$$
$$velocity(V) \tag{6.5}$$

O trabalho de Santos, Hübner e Becker (2015) tinha como objetivo construir e verificar a viabilidade de embarcar um agente Jason para controlar um VANT. Como resultado, esse trabalho apresentou uma arquitetura viável na qual o Jason fornece elementos que facilitam a programação dos comportamentos da aeronave quando comparado a uma implementação baseada em programação procedural.

A implementação do planejamento de rota proposto neste capítulo substitui o agente de planejamento do modelo de Santos, Hübner e Becker (2015) por um agente Huginn. Essa composição possibilita ao agente deliberar sobre normas (Huginn) e interagir com a camada de controle do VANT através das percepções e ações do modelo Santos, Hübner e Becker (2015).

6.2 PRIMEIRO CENÁRIO

O primeiro cenário, extraído de Santos, Hübner e Becker (2015), é uma aplicação para visitar os nodos de uma rede de sensores sem fio com um VANT. Os sensores estão distribuídos em uma região de monitoramento. A aeronave necessita visitar os nodos, partindo e regressando à estação base. A figura 10 exemplifica esse cenário, onde o

[2] Quantidade de bateria necessária para percorrer um metro.

círculo vermelho corresponde à estação base e os círculos verdes são os sensores. Esse cenário assume que:

- a posição de cada sensor é conhecida pelo VANT;
- a aeronave parte do solo;
- a aeronave mantém uma altitude de cruzeiro;
- a escolha do sensor considera a carga da bateria e o consumo;
- a escolha do sensor considera o ponto de não retorno[3].

Figura 10 – Exemplo do primeiro cenário

[3] Ponto de não retorno é o ponto a partir do qual a aeronave não pode mais voltar para a base

6.2.1 Modelagem

Um agente Huginn é uma extensão de um agente Jason, portanto é possível que o agente possua objetivos que não façam parte do processo de deliberação Huginn. Essa é uma característica desejável, pois permite ao desenvolvedor criar objetivos que o agente sempre tentará cumprir independentemente da deliberação. Posto isso, um objetivo Huginn é gerado por um desejo ou norma e passa pelo processo de deliberação. Um objetivo nativo do Jason não tem uma fonte geradora específica e não passa pelo processo de deliberação.

A modelagem do sistema foi baseada no método Prometheus AEOlus (UEZ, 2013). Para melhor representar as características do Huginn, o método foi estendido. A extensão do Diagrama de Objetivos do Sistema permitiu a análise dos objetivos através de uma árvore de decomposição AND/OR. A figura 11 representa como o agente Huginn foi modelado para planejar a rota do VANT. Esse diagrama possui uma implicação sequencial e paralela, de modo a descrever os objetivos do sistema durante a sua execução. As elipses amarelas são objetivos não considerados pelo Huginn e as elipses azuis representam os objetivos derivados de normas e desejos e portanto sujeitos ao processo decisório do huginn.

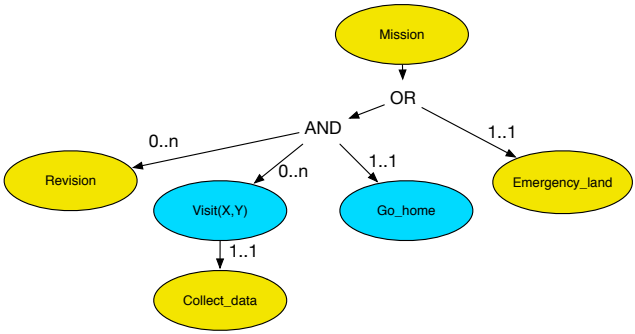

Figura 11 – Diagrama de Objetivos do Sistema.

O objetivo *mission* tem como propósito inicializar o agente, dis-

parando os demais objetivos do agente. O objetivo *revision* almeja atualizar os valores de benefício e tensão dos objetivos Huginn, para que esses se mantenham fiéis às condições atuais do VANT. O objetivo *visit(X, Y)* almeja levar o VANT até a posição X, Y, e ao atingir a posição, é criado um objetivo *collect_data* para coletar os dados do sensor. Em paralelo ao *visit(X, Y)*, o objetivo huginn *go_home* almeja posicionar o VANT junto à estação base. Por fim, em caso de desconformidade no funcionamento do VANT, o *emergency_land* almeja um pouso na posição atual do VANT.

Nessa modelagem, os objetivos: *mission*, *revision*, *collect_data* e *emergency_land* são objetivos não tratados pelo Huginn. A *revision* é um objetivo operacional que deve ser executado para atualizar os valores de benefício e tensão dos desejos. O objetivo *collect_data* é um sub-objetivo de *visit(X, Y)*, sendo assim, ao suspender *visit(X, Y)*, o *collect_data* também é suspenso. Por fim, quando há um objetivo *emergency_land*, ele tem a preferência absoluta para o seu cumprimento, não necessitando passar por um processo de deliberação.

Na modelagem proposta, *visit(X, Y)* e *go_home* são objetivos Huginn, pois são gerados pelos desejos de coletar os dados dos sensores e voltar à base. Ambos necessitam passar por um processo de deliberação para identificar qual composição desses tem um melhor ganho para o agente. Por esses motivos, eles são modelados como objetivos Huginn.

A dinâmica dos benefícios segue as equações (6.6) e (6.7). A dinâmica do benefício do objetivo *visit(X, Y)* é dada pela função inversa da distância até o nodo e a do objetivo *ir para casa* é dada pela relação entre a energia disponível e a distância da base.

$$bft(visit(X,Y)) = \begin{cases} \dfrac{1}{\Delta D} & \text{se } \Delta D >= 1 \\ 1 & \text{se } \Delta D < 1 \end{cases} \tag{6.6}$$

$$bft(go\_home) = 1 - cos(\frac{\pi}{2} \times \frac{\Delta D}{B \times TRM}) \tag{6.7}$$

ΔD é a distância entre a posição atual e a destino, TRM é a taxa máxima de consumo e B é a carga da bateria.

O diagrama de restrição da figura 12 representa como os objetivos se relacionam com os recursos e quais são as restrições entre eles. Nesse diagrama, observa-se que o objetivo $visit(X,Y)$ é conflitante com ele mesmo e com o *go_home*. Assim apenas um objetivo Huginn pode ser adotado por vez. Os dois objetivos Huginn necessitam do recurso bateria e a dinâmica de consumo desses é representada nas equações (6.8) e (6.9).

$$rr(visit(X,Y), battery) = (\Delta D + \Delta R) \times TRM \tag{6.8}$$

$$rr(go\_home, battery) = \Delta R_H \times TRM \tag{6.9}$$

ΔR é a distância do ponto destino até a estação base e ΔR_H é a distância da posição atual até a estação base.

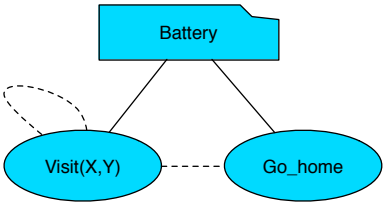

Figura 12 – Diagrama de restrições dos objetivos

O diagrama do agente da figura 13 apresenta os componentes do agente. No diagrama, estão descritos as percepções e ações da infraestrutura apresentada na seção 6.1, as crenças do agente (sensores e altitude cruzeiro) e o recurso bateria. A crença *sensor(X, Y)* é uma representação do desejo de coletar os dados do sensor na posição X, Y e a crença *altitude(A)* é uma representação da altitude cruzeiro que o VANT deve manter.

6.2.2 Implementação

A modelagem do primeiro cenário pelo Huginn define que:

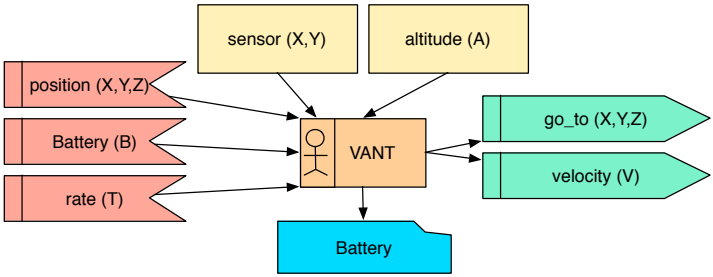

Figura 13 – Diagrama do agente

- cada ponto não visitado é representado por um objetivo derivado de um desejo;
- que o agente tem o objetivo de retornar à base;
- esses objetivos podem existir simultaneamente;
- existe um objetivo especial para atualizar os valores de benefício dos objetivos Huginn.

Posto isto, o Huginn é encarregado de selecionar os objetivos que maximizam o ânimo do agente, respeitando a carga da bateria. O planejamento de rota implementado pode ser dividido em 5 componentes: crenças, recursos, dinâmica do recurso, objetivos, planos e conflitos.

O modelo possui duas crenças implementadas nos predicados (6.10) e (6.11). A crença (6.10) implementa um ponto que o agente deseja visitar, no qual X e Y são valores numéricos que indicam a posição do ponto, levando a estação base como origem. A crença (6.11) implementa a altitude à que o agente voará.

$$sensor(X, Y) \tag{6.10}$$

$$altitude(A) \tag{6.11}$$

Os trechos de código 6.1 e 6.2 implementam, respectivamente, a dinâmica do recurso bateria para o objetivo $visit(X, Y)$ e go_home. Conforme modelado na seção anterior.

```
1  huginn__resource_required (visit(X,Y), battery, B):-
2      position(X0,Y0,_) & distanceXYZ(X0, Y0, 0, X, Y, 0, D) &
3      distanceXYZ(X, Y, 0, 0, 0, 0, DR) & B=(D+DR)*4.
```

Trecho de código 6.1 – Regra de recurso requerido do objetivo $visit(X,Y)$

No trecho de código 6.1, $position(X0,Y0,\_)$ é a crença da posição atual do VANT. Em $distanceXYZ(X0,Y0,0,X,Y,0,D)$ a distância da posição atual (X0,Y0) até o sensor destino (X,Y) está contida na variável D. Em $distanceXYZ(X,Y,0,0,0,0,DR)$ a distância do sensor destino até a estação base está contida na variável DR. Por fim, a variável B é unificada com a soma das distâncias multiplicado pela taxa de consumo máximo (TRM), que nesse caso é 4.

```
1  huginn__resource_required (go_home, battery, B):-
2      position(X0,Y0,_) & distanceXYZ(X0, Y0, 0, 0, 0, 0, DR) &
3      B = DR * 4.
```

Trecho de código 6.2 – Regra de recurso requerido do objetivo go_home

No trecho de código 6.2, o agente não necessita visitar um sensor, logo o recurso requerido é dado pela distância da posição atual até a estação base, multiplicado pela taxa de consumo máxima ($TRM = 4$).

O objetivo *revision* revisa os outros objetivos Huginn, atualizando o valor do benefício, conforme apresentado na modelagem. A revisão ocorre quando um objetivo de visitar um sensor é atingido. O trecho de código 6.3 implementa a revisão dos objetivos Huggin.

```
1  near(X,Y,Z) :-  position(X0,Y0,Z0) & /* regra da cnd. de exp.*/
2                  distanceXYZ(X0, Y0, Z0, X, Y, Z, D) & (D < 0.10).
3
4  @revision[atomic] /*Plano atomico*/
5  +!revision
6    <-
7     for (sensor(ID,X,Y,R)) { /*para cada sensor*/
8       ?sensor_intensity (X,Y,NS); /*necessidade de visitar sensor*/
9       !!visit(X,Y)[
10           huginn_goal, /*cria ou atualiza visit(X,Y)*/
11           benefit(NS),tension(0),
12           cexp(altitude(A) & near(X,Y,A)),
13           typesource(o),
14           source (self)
15      ];
16    }
17    ?go_home_necessity (N); /*necessidade de voltar a base*/
18    !!go_home[huginn_goal, /*cria ou atualiza go_home*/
19         benefit(0),
20         tension(N),
21         cexp(near(0,0,0)),
22         typesource(o),
23         source (self)].
```

Trecho de código 6.3 – Plano para atingir o objetivo *revision*

O plano do trecho de código 6.3, entre as linhas 7 e 15, resgata da base de crenças do agente o benefício obtido por visitar os sensores (linha 8) e cria/atualiza o objetivo $visit(X,Y)$ com os elementos de um objetivo Huginn (linhas 9 à 14). O benefício é igual a intensidade, a tensão é igual a 0 e a condição de expiração é o VANT estar a menos de 10 cm do sensor. Entre as linhas 17 e 23, é resgatada da base de crenças a necessidade de voltar para a base (linha 17) e criado/atualizado o objetivo *go_home* (linhas 18 à 23). O benefício é igual a 0, a tensão é igual à necessidade e a condição de expiração é o VANT estar a menos de 10 cm da estação base. Este plano totaliza 5 ações: percorrer os sensores (linha 7), resgatar a intensidade do sensor (linha 8), criar/atualizar $visit(X,Y)$ (linha 9), resgatar a necessidade

de voltar para a base (linha 17) e criar/atualizar *go_home* (linha 18).

O objetivo *visit(X, Y)* é um objetivo Huginn que visita um sensor na coordenada X, Y. O trecho de código 6.4 é o conjunto de 3 planos que atingem esse objetivo.

```
1  +!visit (X,Y)
2    <-
3       ?altitude(A); /* resgata a altitude */
4       go_to(X,Y,A); /* acao envia a referencia para o controle */
5       .wait(false). /* sincronizacao com o controle */
6
7  ^!visit (X,Y)[state(finished)]
8    <-
9      !collect_data; /* coleta dados */
10     -sensor(_,X,Y,_); /* remove sensor da base de crencas */
11     !!revision. /* cria um objetivo revision */
12
13 ^!visit (X,Y)[state(resumed)]
14     <-
15       ?altitude(A); /* resgata a altitude */
16       go_to(X,Y,A). /* acao envia a referencia para o controle */
```

Trecho de código 6.4 – Planos para atingir o objetivo $visit(X,Y)$

O primeiro plano (linhas 1 a 5) é executado quando o objetivo $visit(X\ ,Y)$ é criado. Nesse plano, é resgatada da base de crenças a altitude cruzeiro (linha 3), executado a ação $go\_to(X,Y,Z)$ para atualizar a referência do controle com a posição do sensor e a altitude cruzeiro (linha 4) e, por fim, uma ação para aguardar que o VANT atinja a condição de expiração do objetivo (linha 5).

O segundo plano (linhas 7 à 11) é executado quando a condição de expiração do objetivo $visit(X,Y)$ é válida. Nesse plano, são coletados os dados (linha 9), removido o sensor da base de crenças (linha 10) e é criado um objetivo *revision* (linha 11).

O terceiro plano (linhas 13 à 16) é executado quando o objetivo $visit(X,Y)$ é retomado. Nesse plano, é resgatada da base de crenças a altitude cruzeiro (linha 15) e executado a ação $go\_to(X,Y,Z)$ para

atualizar a referência do controle com a posição do sensor e a altitude cruzeiro (linha 4). Esse plano se faz necessário, pois, caso o objetivo tenha sido suspenso, é necessário informar novamente a referência ao controle.

O objetivo *go_home* é um objetivo Huginn que leva o VANT de volta à estação base. O trecho de código 6.5 é o conjunto de 3 planos que atingem esse objetivo.

```
1  +!go_home
2    <-
3      go_to(0,0,0); /* acao envia a referencia para o controle */
4      .wait(false). /* sincronizacao com o controle */
5
6  ^!go_home[state(finished)]
7    <-
8      .stopMAS. /* termina a execucao do agente */
9
10 ^!go_home[state(resumed)]
11   <-
12     go_to(0,0,0). /* acao envia a referencia para o controle */
```

Trecho de código 6.5 – Planos para atingir o objetivo $visit(X,Y)$

O primeiro plano (linhas 1 à 4) é executado quando o objetivo *go_ho me* é criado. Nesse plano, é executada a ação $go\_to(X,Y,Z)$ para atualizar a referência do controle com a posição da estação base (linha 3) e uma ação para aguardar que o VANT atinja a condição de expiração do objetivo (linha 4).

O segundo plano (linhas 6 à 8) é executado quando a condição de expiração do objetivo *go_home* é válida. Nesse plano, o agente termina sua execução (linha 8).

O terceiro plano (linhas 10 à 12) é executado quando o objetivo *go_home* é retomado. Nesse plano, é executada a ação $go\_to(X,Y,Z)$ para atualizar a referência do controle com a posição da estação base(linha 12). Como no caso do $visit(X,Y)$, esse plano se faz necessário, pois, caso o objetivo tenha sido suspenso, é necessário informar novamente

a referência ao controle.

O modelo possui dois conflitos implementadas nos predicados (6.12) e (6.13). O conflito (6.12) implementa o conflito entre os objetivos visitar. Dois objetivos $visit(X,Y)$ distintos não podem ocorrer simultaneamente conforme foi apresentado anteriormente no modelo. O conflito (6.13) implementa o conflito entre os objetivos $visit(X,Y)$ e go_home.

$$huginn\_\_\_conflict(visit(\_,\_),visit(\_,\_)) \tag{6.12}$$

$$huginn\_\_\_conflict(visit(\_,\_),go\_home) \tag{6.13}$$

Todos os objetivos Huginn estão nas mesmas classes de conflitos no problema da mochila multidimensional com múltipla escolha. Assim, a resolução do problema sempre seleciona o objetivo com o maior benefício que é viável com a carga atual da bateria. Essa propriedade permite substituir o GLPK por um algoritmo de complexidade $O(|G|)$, onde G é o conjunto de objetivos.

6.2.3 Execução

Os testes foram executados na plataforma de alto nível do Provant (beaglebone), contudo não foi utilizado o corpo da aeronave. O mesmo foi substituído por um modelo computacional do comportamento da aeronave desenvolvido pelo projeto Provant. Dessa forma, o nível mais alto da plataforma interage com o modelo computacional e não com o nível mais baixo (placa discovery). A técnica de troca do corpo da aeronave por um modelo computacional é chamado simulação Hardware in the loop (HIL). Essa técnica é aplicada no desenvolvimento, no teste e na validação de sistemas de tempo real embarcados complexos. O HIL proporciona uma plataforma que permite a adição de complexidade aos testes com um baixo custo e a segurança da plataforma de testes (LOUALI et al., 2011).

No teste, o agente se encontra na estação base e deseja visitar cinco pontos: P1 $(2,3)$, P2 $(3,7)$, P3 $(7,8)$, P4 $(10,2)$ e P5 $(14,1)$. A

figura 14 ilustra esses pontos. A altitude é determinada em 2 metros e é alterada apenas no pouso e decolagem.

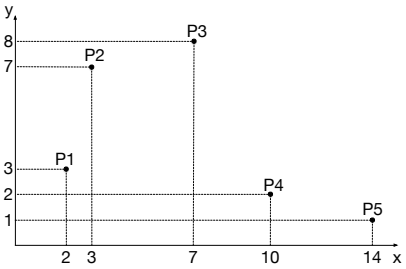

Figura 14 – Cenário 1

As tabelas 5, 6, 7, 8 e 9 apresentam o estado atual do agente, os valores dos elementos do Huginn para cada objetivo e o objetivo selecionado em cada deliberação ocorrida no exemplo. Na tabela 5 o agente se encontra na base, com a carga total na bateria. A deliberação seleciona o objetivo $visit(3,2)$, pois ele possui o maior benefício.

	X	Y	Z	TRM	Bateria	B. Consumida
P. Atual	0	0	0	4	100	0
Objetivo	X	Y	Z	Benefício	Tensão	R. Necessário
visit(3,2)	3	2	2	0,2425	0	32,9848
visit(7,3)	7	3	2	0,1270	0	62,9920
visit(8,7)	8	7	2	0,0924	0	86,5332
visit(2,10)	2	10	2	0,0962	0	83,1384
visit(1,14)	1	14	2	0,0705	0	113,4195
go_home	0	0	0	0	0	0

Tabela 5 – Estado Inicial

Após coletar os dados do $sensor(3,2)$, é criado o objetivo $revision$ para atualizar os valores dos objetivos Huginn. Na tabela 6, estão os valores atualizados dos objetivos. Nessa tabela, é possível observar que o objetivo $visit(3,2)$ deixou de existir, pois foi atingido e o objetivo

go_home tem sua tensão aumentada. Dado esse conjunto de objetivos, a deliberação seleciona o objetivo $visit(7,3)$, pois ele possui o maior benefício.

	X	Y	Z	TRM	Bateria	B. Consumida
P. Atual	3	2	2	4	91,755	8,245
Objetivo	X	Y	Z	Benefício	Tensão	R. Necessário
visit(7,3)	7	3	2	0,2182	0	49,8263
visit(8,7)	8	7	2	0,1360	0	72,6604
visit(2,10)	2	10	2	0,1203	0	74,7957
visit(1,14)	1	14	2	0,0811	0	106,0250
go_home	0	0	0	0	0,0001	14,4222

Tabela 6 – Após visitar o primeiro sensor

Após coletar os dados do $sensor(7,3)$, o processo se repete, o objetivo *revision* atualiza os valores dos objetivos Huginn. Na tabela 7, estão os valores atualizados dos objetivos. A deliberação seleciona o objetivo $visit(8,7)$, pois ele possui o maior benefício e ainda possui recurso suficiente para atender o objetivo. Nessa tabela, observa-se que o consumo de atingir os objetivos $visit(X,Y)$ é menor que o recurso necessário estimado. Esta característica é fruto do caráter conservador da modelagem, na qual é utilizada uma alta taxa de consumo máxima.

	X	Y	Z	TRM	Bateria	B. Consumida
P. Atual	7	3	2	4	85,6325	14,3675
Objetivo	X	Y	Z	Benefício	Tensão	R. Necessário
visit(8,7)	8	7	2	0,2182	0	61,5969
visit(2,10)	2	10	2	0,1132	0	76,8962
visit(1,14)	1	14	2	0,0788	0	107,46409
go_home	0	0	0	0	0,0006	30,4630

Tabela 7 – Após visitar o segundo sensor

Ao fim da coleta dos dados do $sensor(8,7)$, o objetivo *revision*

atualiza os valores dos objetivos Huginn. Na tabela 8, estão os valores atualizados dos objetivos. A deliberação seleciona o objetivo $visit(2,10)$, pois ele possui o maior benefício e ainda possui recurso suficiente para atender o objetivo.

	X	Y	Z	TRM	Bateria	B. Consumida
P. Atual	8	7	2	4	70,235	29,765
Objetivo	X	Y	Z	Benefício	Tensão	R. Necessário
visit(2,10)	2	10	2	0,1428	0	69,5692
visit(1,14)	1	14	2	0,0990	0	97,1078
go_home	0	0	0	0	0,0017	42,5205

Tabela 8 – Após visitar o terceiro sensor

Ao fim da coleta dos dados do $sensor(2,10)$, o objetivo $revision$ atualiza os valores dos objetivos Huginn. Na tabela 9, estão os valores atualizados dos objetivos. A deliberação seleciona o objetivo go_home, pois o objetivo $visit(1,14)$, apesar de ter um benefício maior, não é atingível com a carga atual da bateria. Por fim, o agente chega à base e é desativado.

	X	Y	Z	TRM	Bateria	B. Consumida
P. Atual	2	10	2	4	52,845	47,155
Objetivo	X	Y	Z	Benefício	Tensão	R. Necessário
visit(1,14)	1	14	2	0,2182	0	75,0400
go_home	0	0	0	0	0,0028	40,7921

Tabela 9 – Após visitar o quarto sensor

O resultado da simulação é uma rota onde apenas quatro pontos foram visitados. Os pontos visitados, em ordem, foram: P1, P2, P3 e P4. A figura 15 ilustra a rota adotada. O ponto P5 não foi visitado, pois o VANT não tinha bateria suficiente para visitá-lo e retornar à base. Nesse teste, o agente foi capaz de planejar a rota durante o tempo de

voo, demonstrando que é possível utilizar o Huginn para planejar rotas de forma autônoma no VANT do Provant.

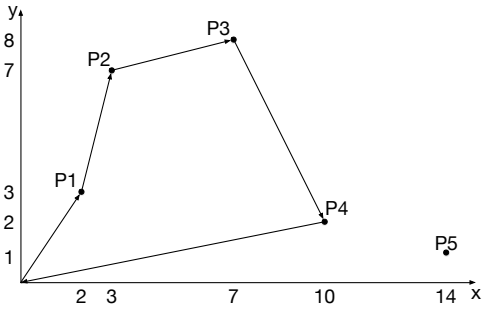

Figura 15 – Rota percorrida

6.2.3.1 Avaliação

Em Santos, Hübner e Becker (2015), foi utilizado o Jason para modelar o comportamento do planejamento de rota em um cenário igual ao apresentado nessa seção. A modelagem do sistema é representada por uma extensão Diagrama de Objetivos do Sistema do Prometheus AEOlus (UEZ, 2013). A figura 16 representa como o agente proposto por Santos, Hübner e Becker (2015) foi modelado para planejar a rota do VANT.

No Jason, a deliberação sobre qual sensor será visitado é explicitamente codificada, juntamente com a gerência da carga da bateria. Na figura 16, destacam-se os objetivos *select sensor, check position e check battery* que o agente desenvolvido em Santos, Hübner e Becker (2015) acrescenta para deliberar sobre qual sensor será visitado ou se o agente retornará à base.

Na implementação proposta nessa tese, todos os objetivos são instanciados simultaneamente, permitindo que o Huginn gerencie qual e quando objetivos serão adotados. Posto isto, o desenvolvedor não necessita de um conjunto de objetivos para gerenciar a bateria e os objetivos *visit(X,Y)* e *go_home* e, tão pouco, precisa se preocupar

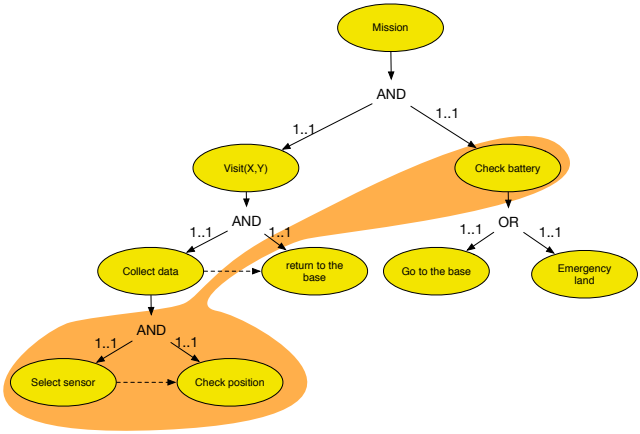

Figura 16 – Objetivos que produzem o raciocínio

com a ordem que os objetivos serão atendidos. A desvantagem da implementação proposta fica por conta do custo extra no processo de deliberação, mesmo que minimizado pela troca do solver.

6.3 SEGUNDO CENÁRIO

O segundo cenário adiciona dois tipos de normas ao primeiro cenário. A primeira é a obrigação de voar em uma altura específica dentro de um determinado espaço aéreo. A segunda é a proibição de visitar um sensor específico. O agente, além de planejar a rota, tem que considerar a alteração da altitude e as proibições. A figura 17 exemplifica o cenário, onde o círculo vermelho corresponde à estação base, os círculos verdes são os sensores, as áreas em azul são os espaços aéreos e o X vermelho é a proibição de visitar um sensor.

6.3.1 Modelagem

Assim como o primeiro cenário, foi utilizada uma extensão do Prometheus AEOlus (UEZ, 2013) para modelá-lo. Algumas partes da modelagem do primeiro cenário estão reproduzidas nessa seção por

Figura 17 – Exemplo do segundo cenário

questões de clareza. A figura 18 representa o diagrama de objetivos do sistema. Os objetivos em vermelho são objetivos obtido por proibições e, relembrando, que os objetivos em azul são objetivos Huginn. O objetivo *mission* tem como propósito inicializar o agente, disparando os demais objetivos do agente. O objetivo *revision* almeja atualizar os valores do benefício e tensão dos objetivos Huginn, para que esses se mantenham fiéis às condições atuais do VANT. O objetivo *visit(X, Y)* em azul almeja levar o VANT até a posição X, Y e, ao atingir a posição, é criado um objetivo *collect_data* para coletar os dados do sensor. O objetivo *visit(X, Y)* em vermelho almeja não levar o VANT até a posição X, Y. O objetivo *altitude_adjust(A)* almeja ajustar a altitude conforme o espaço aéreo demanda. O objetivo Huginn *go_home* almeja posicionar o VANT junto à estação base. Por fim, em caso de desconformidade no funcionamento do VANT, o *emergency_land* almeja um

pouso na posição atual do VANT.

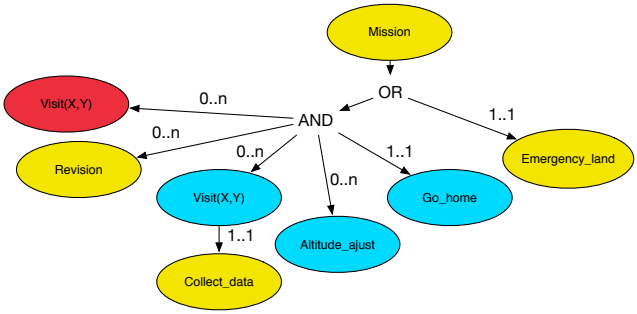

Figura 18 – Diagrama de Objetivos do Sistema.

Nessa modelagem, os objetivos *mission, revision, collect_data, emergency land, visit(X,Y)* e *go_home* são idênticos aos objetivos do primeiro cenário (seção 6.2). O objetivo *altitude_adjust(A)* almeja ajustar a altitude do VANT conforme o espaço aéreo solicita através de uma obrigação.

A dinâmica dos benefícios dos objetivos existentes no primeiro não sofreram alteração. A dinâmica do benefício do objetivo *visit(X,Y)* é dada pela função inversa da distância até o nodo (equação (6.14)) e a do objetivo *go_home* é dada pela relação entre a energia disponível e a distância da base (equação (6.15)).

$$bft(visit(X,Y)) = \begin{cases} \dfrac{1}{\Delta D} & \text{se } \Delta D >= 1 \\ 1 & \text{se } \Delta D < 1 \end{cases} \tag{6.14}$$

$$bft(go\_home) = 1 - cos(\frac{\pi}{2} \times \frac{\Delta D}{B \times TRM}) \tag{6.15}$$

ΔD é a distância entre a posição atual e a destino, TRM é a taxa máxima de consumo e B é a carga da bateria.

A dinâmica do benefício e da tensão do objetivo *altitude_adjust (A)* é dada pela sanção e recompensa da norma. Neste cenário, as normas têm alta importância, logo o benefício e tensão possuem os valores máximos (igual a 1).

O diagrama de restrição da figura 19 representa como os objetivos se relacionam com os recursos e quais são as restrições entre eles. Nesse diagrama, observa-se que os objetivos $visit(X,Y)$, go_home e $altitude\_ad\ just(A)$ são conflitantes entre si. Os três objetivos Huginn necessitam do recurso bateria e a dinâmica de consumo desses é representada nas equações (6.16), (6.17) e (6.18). Por fim, existe um conflito entre o objetivo $visit(X,Y)$ e a proibição de visitar X,Y (representado pelo objetivo *visit(X, Y)* em vermelho).

$$rr(visit(X,Y), battery) = (\Delta D + \Delta R) \times TRM \tag{6.16}$$

$$rr(altitude\_adjust(A), battery) = \Delta D \times TRM \tag{6.17}$$

$$rr(go\_home, battery) = \Delta R_H \times TRM \tag{6.18}$$

ΔR é a distância do ponto destino até a estação base e ΔR_H é a distância da posição atual até a estação base.

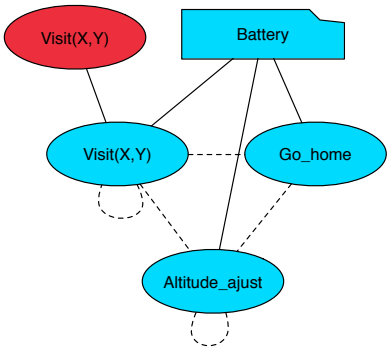

Figura 19 – Diagrama de restrições dos objetivos

O diagrama do agente da figura 20 apresenta os componentes do agente. No diagrama, estão descritas as percepções e ações da infraestrutura apresentada na seção 6.1 mais as percepções das obrigações (*obligation (altitude (A))*) e proibições (*prohibition (visit (X, Y))*), as crenças do agente (sensores, altitude, coordenada destino, proibições)

e o recurso bateria. A crença *sensor(X, Y)* é uma representação do desejo de coletar os dados do sensor na posição X, Y, a crença *altitude(A)* é uma representação da altitude que o VANT deve manter e a crença *target_goto(X, Y)* é uma representação do ponto destino.

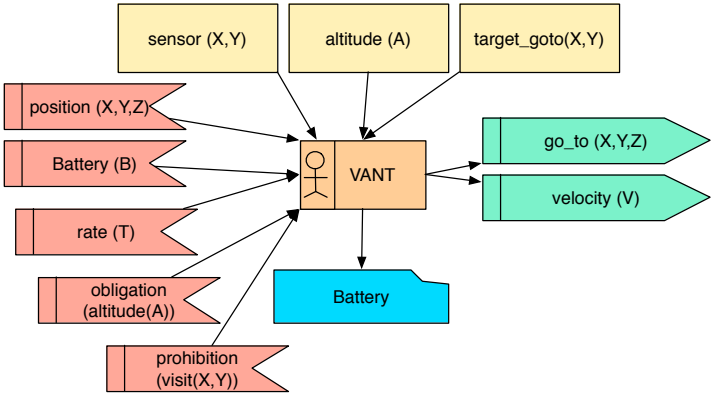

Figura 20 – Diagrama do agente

6.3.2 Implementação

A modelagem do segundo cenário define que:

- cada ponto não visitado é representado por um objetivo derivado de um desejo;
- o agente tem o objetivo de retornar à base;
- a aeronave tem obrigação de voar a uma altitude específica conforme determinado pelo espaço aéreo;
- a aeronave tem proibições de visitar sensores;
- a escolha do sensor considera a carga da bateria, as obrigações, as proibições e o consumo;
- a escolha do sensor considera o ponto de não retorno.

Posto isto, o Huginn é encarregado de selecionar as normas e os desejos que maximizam o ânimo do agente, respeitando a carga da bateria. Essa implementação pode ser dividida em 6 componentes: crenças, recursos, dinâmica do recurso, objetivos, planos e conflitos.

O modelo possui quatro crenças implementadas nos predicados (6.19), (6.20) e (6.21). A crença (6.19) implementa um ponto que o agente deseja visitar, no qual X e Y são valores numéricos que indicam a posição o ponto, levando à estação base como origem. A crença (6.20) implementa a altitude à que o agente voará. A crença (6.21) implementa um ponto que o agente selecionou para visitar.

$$sensor(X,Y) \tag{6.19}$$

$$altitude(A) \tag{6.20}$$

$$target\_goto(X,Y) \tag{6.21}$$

A dinâmica do uso da bateria pelos objetivos $visit(X,Y)$ e *go_home* continuam inalteradas em relação ao cenário da seção 6.2. Nesse cenário, acrescentou-se a dinâmica do objetivo *altitude_adjust(A)*. O trecho de código 6.6 implementa a dinâmica do recurso bateria para o objetivo $altitude\_adjust(A)$.

```
1  huginn__resource_required (altitude_adjust(A), battery, C):-
2      position(X0,Y0,Z0) & target_goto (X,Y) &
3      ALPHA = (Y-Y0)/(X-X0) &
4      B = Y0 - ALPHA*X0 &
5      XR= X0+ 0.25*(X-X0)/math.abs(X-X0) &
6      YR= ALPHA*XR+B &
7      distanceXYZ(X0, Y0, Z0, XR, YR, A, D) &
8      C=D*4.
```

Trecho de código 6.6 – Regra de recurso requerido do objetivo $altitide\_ajust(A)$

No trecho de código 6.6, $position(X0,Y0,Z0)$ é a crença da posição atual do VANT. *target_goto (X,Y)* é a crença com o ponto destino. As linhas 3 até 6 calculam um ponto intermediário na rota traçada anteriormente. As linhas 7 e 8 calculam a distância do ponto atual até o ponto intermediário na nova altitude e multiplica pela taxa

de consumo máximo, que nesse caso é 4, para obter o consumo máximo esperado da bateria.

O objetivo *revision* revisa os outros objetivos Huginn gerados pelos desejos (*visit(X,Y)* e *go_home*), não sofre alteração em relação ao apresentado no exemplo da seção 6.2. A revisão ocorre quando o objetivo: visitar um sensor, alterar altitude, ou ir para a base é atingido. O trecho de código 6.3 é o plano que implementa a revisão dos objetivos Huggin.

O objetivo *visit(X,Y)* é um objetivo Huginn que visita um sensor na coordenada X, Y. O trecho de código 6.7 é o conjunto de 3 planos que atingem esse objetivo. Em relação aos planos do cenário anterior, foram acrescentadas apenas as linhas 4 e 17.

```
1  +!visit (X,Y)
2    <-
3       ?altitude(A); /* resgata a altitude */
4       -+target_goto(X,Y); /* Atualiza ponto destino */
5       go_to(X,Y,A); /* acao envia a referencia para o controle */
6       .wait(false). /* sincronizacao com o controle */
7
8  ^!visit (X,Y)[state(finished)]
9    <-
10      !collect_data; /* coleta dados */
11      -sensor(_,X,Y,_); /* remove sensor da base de crencas */
12      !!revision. /* cria um objetivo revision */
13
14 ^!visit (X,Y)[state(resumed)]
15     <-
16       ?altitude(A); /* resgata a altitude */
17       -+target_goto(X,Y); /* Atualiza ponto destino */
18       go_to(X,Y,A). /* acao envia a referencia para o controle */
```

Trecho de código 6.7 – Planos para atingir o objetivo *visit*(X,Y)

O primeiro plano (linhas 1 a 6) é executado quando o objetivo *visit*(X, Y) é criado. Nesse plano, é resgatada da base de crenças a altitude cruzeiro (linha 3), é atualizada a crença *target_goto(X,Y)* com ponto destino, executada a ação *go_to*(X,Y,Z) para atualizar a referência do controle com a posição do sensor e a altitude cruzeiro (linha 5) e, por fim, é realizada uma ação para aguardar que o VANT atinja a condição de expiração do objetivo (linha 6).

O segundo plano (linhas 8 a 12) é executado quando a condição de expiração do objetivo $visit(X,Y)$ é válida. Nesse plano, são coletados os dados (linha 10), é removido o sensor da base de crenças (linha 11) e é criado um objetivo *revision* (linha 12).

O terceiro plano (linhas 14 a 18) é executado quando o objetivo $visit(X,Y)$ é retomado. Nesse plano, é resgatada da base de crenças a altitude cruzeiro (linha 16), é atualizada a crença target_goto(X,Y) com ponto destino (linha17) e é executada a ação $go\_to(X,Y,Z)$ para atualizar a referência do controle com a posição do sensor e a altitude cruzeiro (linha 18). Esse plano se faz necessário, pois caso, o objetivo tenha sido suspenso, é necessário informar novamente a referência ao controle.

O objetivo *go_home* é um objetivo Huginn que leva o VANT de volta à estação base. Esse objetivo não sofreu alteração em relação ao cenário apresentado na seção 6.2. O trecho de código 6.5 é o conjunto de 3 planos atingem esse objetivo.

O objetivo *altitude_adjust(A)* é um objetivo Huginn que altera a altura do VANT para a altitude A. O trecho de código 6.8 é o conjunto de 2 planos atingem esse objetivo.

```
1  +!altitude_adjust(A)
2    <-
3      ?target_goto(X,Y); /* resgata o ponto destino */
4      ?position(X0,Y0,Z0); /* resgata a posicao atual */
5      ALPHA = (Y-Y0)/(X-X0); /* calc. o ang. de inclinacao */
6      B = Y0 - ALPHA*X0; /* calc. o termo independente */
7      XR= X0+ 0.25*(X-X0)/math.abs(X-X0); /* calc. X pto. interm. */
8      YR= ALPHA*XR+B;        /* calc. Y do pto. interm. */
9      go_to(XR,YR,A); /* acao envia a referencia para o controle */
10     .wait(false). /* sincronizacao com o controle */
11
12 ^!altitude(A)[state(finished)]
13   <-
14     !!revision. /* cria um objetivo revision */
```

Trecho de código 6.8 – Planos para atingir o objetivo $altitude\_adjust(A)$

O plano do trecho de código 6.8 resgata da base de crenças do agente o destino atual do agente (linha 3) e a posição atual (linha 4). Entre as linhas 5 e 8, é calculado um ponto dentro da rota estabelecida. A partir do ponto calculado, o agente se encontrará na altitude A. A execução a ação *go_to(XR,YR,A)* atualiza a referência do controle com a posição do ponto intermediário e a nova altitude (linha 9).

Quando o VANT percebe uma nova norma do tipo *prohibition(visit (X,Y))*, é necessário que ela seja traduzida a um objetivo Huginn. O trecho de código 6.9 é o plano que faz essa tradução. O plano cria/atualiza um objetivo do tipo *huginn_prohibition* e suas anotações do Huginn (linhas 3 até 7).

```
1  +prohibition(visit (X,Y))
2   <-
3     !!prohibition(visit (X,Y))
4              [ huginn_prohibition,
5                benefit(1), tension(1),
6                cexp( open(X,Y)),
7                typesource(p),source(org) ].
```

Trecho de código 6.9 – Plano para traduzir a norma *prohibition(visit (X,Y))*

Quando o VANT percebe uma nova norma do tipo *obligation (altitude (A))*, é necessário que ela seja traduzida a um objetivo Huginn. O trecho de código 6.10 é o plano que faz essa tradução. O plano cria um objetivo *altitude_adjust(A)* com as anotações do Huginn (linhas 3 até 8).

```
1  +obligation(altitude(A))
2   <-
3     !!altitude_adjust(A)
4              [ huginn_goal,
5                benefit(1),tension(1),
6                cexp( position(_,_,Z) &
7                      distanceXYZ(0, 0, Z, 0, 0, A, D) & (D < 0.11)),
8                typesource(o), source (tower) ].
```

Trecho de código 6.10 – Plano para traduzir a norma *obligation(altitude(A))*

O modelo possui 5 conflitos implementadas nos predicados (6.22), (6.23), (6.24), (6.25), e (6.26). O conflito (6.22) implementa o conflito entre os objetivos visitar. Dois objetivos $visit(X,Y)$ distintos não podem ocorrer simultaneamente conforme foi apresentado anteriormente no modelo. O conflito (6.23) implementa o conflito entre os objetivos $visit(X,Y)$ e go_home. O conflito (6.24) implementa o conflito entre dois objetivos $altitude\_adjust(A)$. O conflito (6.25) implementa o conflito entre os objetivos $altitude\_adjust(A)$ e go_home. O conflito (6.26) implementa o conflito entre os objetivos $altitude\_adjust(A)$ e go_home.

$$huginn\_\_\_conflict(visit(\_,\_),visit(\_,\_)) \tag{6.22}$$

$$huginn\_\_\_conflict(visit(\_,\_),go\_home) \tag{6.23}$$

$$huginn\_\_\_conflict(altitude\_adjust(\_),altitude\_adjust(\_)) \tag{6.24}$$

$$huginn\_\_\_conflict(altitude\_adjust(\_),go\_home) \tag{6.25}$$

$$huginn\_\_\_conflict(altitude\_adjust(\_),visit(\_,\_)) \tag{6.26}$$

Assim como no cenário apresentado na seção 6.2, o GLPK foi substituído por uma heurística para se adequar aos recursos limitados da Beaglebone. Para esse cenário, foi utilizada uma heurística baseada em um algoritmo guloso proposto em Kellerer, Pferschy e Pisinger (2004), que possui um tempo de resposta melhor, mas uma qualidade de solução inferior ao GLPK.

O algoritmo 3 apresenta a função seleção de objetivos que implementa a deliberação normativa. Esta função tem como entrada o conjunto objetivos O. O algoritmo ordena o conjunto O por ordem decrescente de ganho (benefício+tensão). Na etapa seguinte, percorre o conjunto ordenado adicionando ao conjunto M aqueles objetivos que não ultrapassam a quantidade disponível de recursos e que não tenham conflito com os objetivos já presentes nesse conjunto (linhas 3 a 7 do algoritmo 3). O recurso necessário ($rr(r,o)$) para atingir cada objetivo é reservado quando este é adicionado a M, para tanto, a quantidade necessária é deduzida do recurso disponível ($ar(r)$).

Algorithm 3: Heurística para deliberação normativa

Input: Conjunto de objetivos O
Output: Conjunto de objetivos selecionados M
$sort(O)$;
$M := \emptyset$;
for *each* $(o \in O)$ **do**
  if $\forall r \in R \;\; rr(r,o) \leq ar(r) \wedge$
  $\forall m \in M \neg conflict(o,m)$ **then**
    $M := M \cup \{o\}$;
    for *each* $(r \in R)$ **do**
      $ra(r) = ra(r) - rr(r,o)$
return M

6.3.3 Execução

O teste, assim como no primeiro cenário, foi executado na plataforma de alto nível do Provant (beaglebone) com uma simulação Hardware in the loop (HIL). No teste executado, o agente inicialmente se encontra na estação base e deseja visitar cinco pontos: P1 $(2,3)$, P2 $(3,7)$, P3 $(7,8)$, P4 $(10,2)$ and P5 $(14,1)$. O ambiente possui cinco espaços aéreos com ponto central igual aos pontos P1 a P5 e raio igual a 2 metros. Ainda no primeiro momento, o agente é proibido de visitar o ponto P3. A figura 21 representa o ambiente.

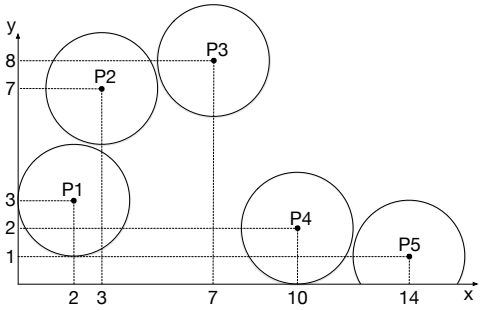

Figura 21 – Cenário 2

As tabelas 10, 11, 12 e 13 apresentam o estado atual do agente, os valores dos elementos do Huginn para cada objetivo e o objetivo selecionado em cada deliberação ocorrida no exemplo. Na tabela 10, o agente se encontra na base com a carga total na bateria. A deliberação seleciona o objetivo $visit(2,3)$, pois ele possui o maior benefício.

		X	Y	Z	TRM	Bateria	B. Cons.
	P. Atual	0	0	0	4	100	0
Tipo	Objetivo	X	Y	Z	Benefício	Tensão	R. Nec.
o	visit(2,3)	2	3	2	0,2425	0	32,9848
o	visit(3,7)	3	7	2	0,1270	0	62,9920
o	visit(7,8)	7	8	2	0,0924	0	86,5332
o	visit(10,2)	10	2	2	0,0962	0	83,1384
o	visit(14,1)	14	1	2	0,0705	0	113,4195
o	go_home	0	0	0	0	0	0
p	visit(7,8)	7	8	0	1	1	0

Tabela 10 – Estado Inicial

Ao entrar no espaço aéreo (2,3), o agente percebe uma obrigação para corrigir a altitude para 4 metros, gerando um objetivo *altitude_adjust(4)*. Na tabela 11, estão os valores atualizados dos objetivos. Nessa tabela, é possível observar que o objetivo $visit(2,3)$ ainda existe, pois não foi atingido e o objetivo *altitude_adjust(4)* foi criado. Dado esse conjunto de objetivos, o processo de deliberação seleciona o objetivo *altitude_adjust(4)*, pois ele produz maior benefício.

Após cumprir o objetivo *altitude_adjust(4)*, o agente delibera sobre qual o próximo objetivo a ser cumprido. Na tabela 12, estão os valores atualizados dos objetivos. Nessa tabela, é possível observar que o objetivo *altitude_adjust(4)* não existe mais, pois foi atingido e o valor de Z, para todos os objetivos, foi alterado para a altitude cruzeiro do espaço aéreo. Dado esse conjunto de objetivos, o processo de deliberação seleciona o objetivo *visit(2,3)*, pois ele produz maior benefício.

Após cumprir o objetivo *visit(2,3)*, o agente delibera sobre qual o próximo objetivo a ser cumprido. Na tabela 13, estão os valores atu-

		X	Y	Z	TRM	Bateria	B. Cons.
	P. Atual	1,5	1,1	2	4,0000	97	3
Tipo	Objetivo	X	Y	Z	Benefício	Tensão	R. Nec.
o	visit(2,3)	2	3	2	0,2425	0	32,9848
o	visit(3,7)	3	7	2	0,1270	0	62,9920
o	visit(7,8)	7	8	2	0,0924	0	86,5332
o	visit(10,2)	10	2	2	0,0962	0	83,1384
o	visit(14,1)	14	1	2	0,0705	0	113,4195
o	go_home	0	0	0	0	0	0
p	visit(7,8)	7	8	0	1	1	0
o	altitude_ adjust(4)	1,75	1,25	4	1,0000	1	16,0424

Tabela 11 – Ao entrar no espaço aéreo

		X	Y	Z	TRM	Bateria	B. Cons.
	P. Atual	1,75	1,25	4	4,0000	92,7800	7,2200
Tipo	Objetivo	X	Y	Z	Benefício	Tensão	R. Nec.
o	visit(2,3)	2	3	4	0,5657	0	21,4933
o	visit(3,7)	3	7	4	0,1699	0	54,0003
o	visit(7,8)	7	8	4	0,1169	0	76,7258
o	visit(10,2)	10	2	4	0,1207	0	73,9282
o	visit(14,1)	14	1	4	0,0816	0	105,1529
p	visit(7,8)	7	8	0	1	1	0
o	go_home	0	0	4	0,0000	0	8,6023

Tabela 12 – Ao ajustar a altitude

alizados dos objetivos. Nessa tabela, é possível observar que o objetivo *visit(2,3)* não existe mais, pois foi atingido. Dado esse conjunto de objetivos, o processo de deliberação seleciona o objetivo *visit(3,7)*, pois ele produz maior benefício.

Nesse documento, não apresentaremos todos os passos da simulação, pois ela é grande e repetitiva. A simulação resultou uma rota que visitou todos os pontos. A figura 22 representa a rota adotada. O VANT visitou os pontos P1 e P2 e retornou para base, porque é proibido visitar o ponto P3 e ele não tinha bateria suficiente para visitar os pontos P4 e P5. Após decolar novamente, a proibição de visitar P3

		X	Y	Z	TRM	Bateria	B. Cons.
	P. Atual	2	3	4	4,0000	87,4075	12,5925
Tipo	Objetivo	X	Y	Z	Benefício	Tensão	R. Nec.
o	visit(3,7)	3	7	4	0,2425	0	46,9555
o	visit(7,8)	7	8	4	0,1414	0	70,8049
o	visit(10,2)	10	2	4	0,1240	0	73,0412
o	visit(14,1)	14	1	4	0,0822	0	104,8048
p	visit(7,8)	7	8	0	1	1	0
o	go_home	0	0	4	0,0000	0,0001	14,4222

Tabela 13 – Ao visitar o primeiro sensor

expirou. Assim, o VANT visitou o ponto P3 e P4 e retornou à base para recarregar. O último ponto visitado foi P5. O VANT ajustou a altitude toda vez que entrou em um novo espaço aéreo. O ajuste de altitude é representado pelas setas vermelhas da figura 22.

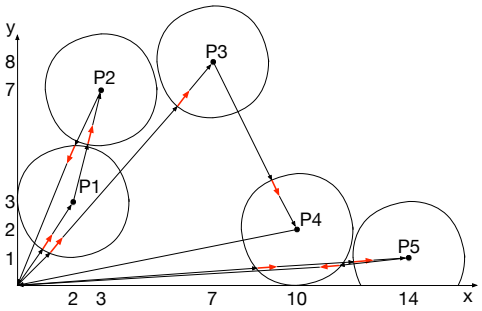

Figura 22 – Rota adotada

6.3.4 Avaliação

No teste, o agente foi capaz de planejar a rota durante o voo, ajustando a altitude e respeitando as proibições. Com esse teste, demonstramos que é possível usar o Huginn para planejar a rota do VANT em um cenário mais complexo.

A complexidade do cenário da seção 6.2 foi aumentada e, mesmo assim, foi possível reutilizar o código do agente do primeiro cenário

integralmente, apenas adicionando os novos objetivos. O conceito do Huginn permite criar novos objetivos, tratando desejos e normas de forma independente. Os conflitos entre esses objetivos são resolvidos pelo processo de deliberação. O agente apenas necessita, que junto com as normas e desejos, sejam especificados os elementos do Huginn. Por outro lado, o processo de deliberação aumenta o tempo de processamento do ciclo de raciocínio do agente.

Como observado no cenário, o Huginn é declarativo. O programador necessita declarar os recursos disponíveis (crença *huginn___resource*), os recursos necessários para cumprir os objetivos (regras *huginn___resour ce_required*) e sinalizar os objetivos que são considerados na deliberação do Huginn (anotações *huginn_goal* e *huginn_prohibition*). O Huginn também fica encarregado do processo de deliberação, diferentemente do Jason puro ou mesmo de outras linguagens procedurais, não onerando o programador.

6.4 RESUMO DO CAPÍTULO

Esse capítulo apresentou uma aplicação de planejamento de rota para VANT, utilizando o Huginn. Os cenários apresentados demonstraram que é possível utilizar o Huginn para programar o comportamento de um VANT. Ainda, os cenários apresentaram que a programação é facilitada pela modularização dos objetivos.

Os cenários utilizados corroboram os resultados do capítulo **??**, no qual se conclui que o processo de deliberação é viável. Na aplicação escolhida destacam-se:

- a capacidade de utilizar o huginn para o planejamento de rota;
- a facilidade de incluir novos comportamentos no agente.

7 CONCLUSÃO

Nesse trabalho, foi apresentado um modelo de raciocínio normativo e sua implementação. A principal contribuição dessa tese, conforme identificado no capítulo 3, é a capacidade de o agente compreender e deliberar sobre desejos e normas considerando os recursos disponíveis. Para desenvolver o modelo de raciocínio, foi criado um processo de deliberação capaz de:

- raciocinar sobre a relevância dos desejos e normas;
- resolver conflitos entre desejos e normas;
- resolver conflitos por recurso.

No capítulo 4 propôs-se resolver esses problemas:

- um modelo baseado em ânimo, no qual é possível definir a tensão (feedback negativo), benefício (feedback positivo) e recursos para raciocinar sobre o cumprimento e relevância dos desejos e normas;
- reduzir o problema de conflitos entre desejos e normas ao problema da mochila com múltipla escolha;
- reduzir o problema de conflito de recursos ao problema da mochila multidimensional.

A avaliação do modelo de raciocínio proposto foi apresentada nos capítulos 5 e 6. Nesses capítulos, o modelo implementado foi avaliado quanto a sua expressividade e aplicação em um veículo aéreo não tripulado.

Na aplicação apresentada no capítulo 6, observou-se a capacidade de utilizar o Huginn para modelar o planejamento de rota de um VANT. O capítulo 6, através dos cenários apresenta como pode ser modelado um agente Huginn.

7.1 TRABALHOS FUTUROS

O trabalho apresentado, em seu estado atual, apresenta diversos caminhos para a continuação. Os principais são:

- o tratamento de conflitos indiretos;
- o conceito de ânimo de um grupo de agentes.

O tratamento de conflitos indiretos é um tópico interessante a ser explorado. Esse tratamento permitirá ao agente ter um comportamento consistente, evitando o gasto de recursos com objetivos não relacionados. Para solucionar esse problema, é necessário que o agente tenha um conhecimento semântico dos objetivos. Esse tratamento enriquecerá a arquitetura do agente, que terá que representar o elemento semântico dos objetivos, e o processo de deliberação possivelmente ganhará um novo conjunto de restrições, impactando no tempo de resposta. Um exemplo desse tipo de conflito é um agente que possui 3 normas:

- andar de bicicleta;
- chegar em um ponto X, a 5 km de distância em 1 hora;
- fazer um desvio de 40 km.

O agente pode assumir duas dessas normas em qualquer arranjo, mas ao assumir as três, elas se inviabilizam.

O conceito de ânimo de grupo permite que uma população de agentes assuma normas e desejos baseados no ânimo do coletivo. Isso implica uma tomada de decisão coletiva, em determinar como a tensão e o benefício interagem com as normas coletivas e de como é feito o balanceamento dos objetivos entre os agentes.

REFERÊNCIAS

ALECHINA, N.; DASTANI, M.; LOGAN, B. Programming norm-aware agents. In: CONITZER, V. et al. (Ed.). *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*. Valencia, Spain: [s.n.], 2012. Citado 8 vezes nas páginas 19, 50, 51, 52, 53, 60, 77 e 78.

ANDRIGHETTO, G. et al. On the immergence of norms: a normative agent architecture. In: *Proceedings of AAAI Symposium, Social and Organizational Aspects of Intelligence*. [S.l.: s.n.], 2007. Citado na página 19.

BOELLA, G.; TORRE, L.; VERHAGEN, H. Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, Kluwer Academic Publishers, v. 12, n. 2-3, p. 71–79, 2006. ISSN 1381-298X. Citado na página 17.

BOELLA, G.; TORRE, L. W. N. van der. Fulfilling or violating obligations in normative multiagent systems. In: *IAT*. [S.l.: s.n.], 2004. p. 483–486. Citado na página 19.

BOELLA, G.; TORRE, L. W. N. van der. Regulative and constitutive norms in normative multiagent systems. In: *KR*. [S.l.: s.n.], 2004. p. 255–266. Citado na página 23.

BORDINI, R. H.; HÜBNER, J. F.; WOOLDRIDGE, M. *Programming Multi-Agent Systems in AgentSpeak using Jason*. [S.l.]: Wiley-Interscience, 2007. ISBN 0470029005. Citado 2 vezes nas páginas 28 e 81.

BRATMAN, M. E. *Intention, Plans, and Practical Reason*. [S.l.]: Cambridge University Press, 1999. Paperback. ISBN 1575861925. Citado 3 vezes nas páginas 18, 27 e 64.

BROERSEN, J. et al. The boid architecture: conflicts between beliefs, obligations, intentions and desires. In: *Proceedings of the fifth international conference on Autonomous agents*. New York, NY, USA: ACM, 2001. (AGENTS '01), p. 9–16. ISBN 1-58113-326-X. Citado 2 vezes nas páginas 33 e 35.

BROERSEN, J.; TORRE, L. van der. Reasoning about norms, obligations, time and agents. In: *PRIMA*. [S.l.]: Springer, 2007. (Lecture Notes in Computer Science). Citado na página 19.

BULFINCH, T. *O Livro De Ouro Da Mitologia: Histórias De Deuses E Heróis*. [S.l.]: Editora Ediouro, 1998. ISBN 8500006714. Citado na página 63.

CARABELEA, C.; BOISSIER, O.; CASTELFRANCHI, C. Using Social Power to Enable Agents to Reason About Being Part of a Group. In: GLEIZES, M.-P.; OMICINI, A.; ZAMBONELLI, F. (Ed.). [S.l.]: Springer Berlin / Heidelberg, 2005, (Lecture Notes in Computer Science, v. 3451). p. 898. ISBN 978-3-540-27330-1. Citado 6 vezes nas páginas 19, 48, 49, 50, 60 e 76.

CASTELFRANCHI, C. Commitments: From individual intentions to groups and organizations. In: LESSER, V. R.; GASSER, L. (Ed.). *ICMAS*. [S.l.]: The MIT Press, 1995. p. 41–48. ISBN 0-262-62102-9. Citado na página 26.

CASTELFRANCHI, C. Prescribed mental attitudes in goal-adoption and norm-adoption. *Artificial Intelligence and Law*, Springer Netherlands, v. 7, p. 37–50, 1999. ISSN 0924-8463. 10.1023/A:1008363413485. Citado na página 44.

Prescribed Mental Attitudes in Goal-Adoption and Norm-Adoption, v. 7, n. 1. 37-50 p. Citado 2 vezes nas páginas 53 e 59.

CASTELFRANCHI, C. A micro and macro definition of power. In: *In ProtoSociology - An International Journal of Interdisciplinary Research*. [S.l.: s.n.], 2002. p. 208–268. Citado 3 vezes nas páginas 32, 48 e 76.

CONTE, R.; CASTELFRANCHI, C.; DIGNUM, F. Autonomous norm acceptance. In: MüLLER, J.; RAO, A.; SINGH, M. (Ed.). *Intelligent Agents V: Agents Theories, Architectures, and Languages*. [S.l.]: Springer Berlin / Heidelberg, 1999, (Lecture Notes in Computer Science, v. 1555). p. 99–112. ISBN 978-3-540-65713-2. Citado na página 44.

CRIADO, N.; ARGENTE, E.; BOTTI, V. Rational strategies for norm compliance in the n-BDI proposal. In: VOS, M. D. et al. (Ed.). *Coordination, Organizations, Institutions, and Norms in Agent Systems VI*. [S.l.]: Springer Berlin / Heidelberg, 2011, (Lecture Notes in Computer Science, v. 6541). p. 1–20. ISBN 978-3-642-21267-3. Citado 6 vezes nas páginas 19, 42, 43, 44, 45 e 47.

DASTANI, M. 2apl: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, Springer, v. 16, n. 3, p. 214–248, 2008. Citado 3 vezes nas páginas 28, 50 e 52.

DEMAZEAU, Y.; MÜLLER, J. *Decentralized A.I. 2: proceedings of the Second European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Saint Quentin en Yvelines, France, August 13-16, 1990*. [S.l.]: North-Holland, 1991. (Decentralized A.I. 2: Proceedings of the Second European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Saint Quentin en Yvelines, France, August 13-16, 1990). ISBN 9780444890511. Citado na página 26.

DIGNUM, F. et al. Towards socially sophisticated BDI agents. In: *In Proceedings of the ICMAS 2000*. [S.l.]: IEEE Computer Society, 2000. p. 111–118. Citado na página 33.

D'INVERNO, M.; LUCK, M. (Ed.). *Understanding agent systems*. New York: Springer, 2001. (Springer series on agent technology.). 2001031126 Mark d'Inverno. Michael Luck (eds.) Includes bibliographical references and index. Citado na página 39.

FERBER, J. *Les Systèmes multi-agents: vers une intelligence collective*. [S.l.]: InterEditions, 1995. (I.I.A. Informatique intelligence artificielle). ISBN 9782729605728. Citado na página 26.

FERBER, J.; GUTKNECHT, O.; MICHEL, F. From agents to organizations: An organizational view of multi-agent systems. In: GIORGINI, P.; MÜLLER, J.; ODELL, J. (Ed.). *Agent-Oriented Software Engineering IV*. [S.l.]: Springer Berlin Heidelberg, 2004, (Lecture Notes in Computer Science, v. 2935). p. 214–230. ISBN 978-3-540-20826-6. Citado na página 17.

FIPA - Foundation for Intelligent Physical Agents. 2013. Citado na página 25.

FOUCAULT, M. *What is Enlightenment*. [S.l.]: THE NEW YORK PRESS, 2003. 43–57 p. Citado na página 17.

GAERTNER, D.; TONI, F. Casapi: a system for credulous and sceptical argumentation. In: *Proc. Workshop on Argumentation for Non-monotonic Reasoning. (2007) 80–95*. [S.l.: s.n.], 2007. p. 80–95. Citado na página 38.

GAERTNER, D.; TONI, F. Preferences and assumption-based argumentation for conflict-free normative agents. In: *Proceedings of the 4th international conference on Argumentation in multi-agent systems*. Berlin, Heidelberg: Springer-Verlag, 2008. (ArgMAS'07), p. 94–113. ISBN 3-540-78914-6, 978-3-540-78914-7. Citado 3 vezes nas páginas 19, 38 e 78.

HANSSON, B. An analysis of some deontic logics. In: HILPINEN, R. (Ed.). *Deontic Logic: Introductory and Systematic Readings*. [S.l.]: Springer Netherlands, 1971, (Synthese Library, v. 33). p. 121–147. ISBN 978-90-277-1302-5. Citado na página 66.

JONES, A.; SERGOT, M. *A Formal Characterisation of Institutionalised Power*. 1996. Citado na página 48.

JOSEPH, S. et al. Deductive coherence and norm adoption. *Logic Journal of the IGPL*, v. 18, n. 1, p. 118–156, 2010. Citado 6 vezes nas páginas 19, 57, 58, 59, 78 e 90.

KELLERER, H.; PFERSCHY, U.; PISINGER, D. *Knapsack Problems*. [S.l.]: Springer, 2004. ISBN 3540402861. Citado 3 vezes nas páginas 71, 84 e 119.

KOLLINGBAUM, M. J.; NORMAN, T. J. Informed deliberation during norm-governed practical reasoning. In: *Proceedings of the 2005 international conference on Agents, Norms and Institutions for Regulated Multi-Agent Systems*. Berlin, Heidelberg: Springer-Verlag, 2006. (AAMAS'05), p. 183–197. ISBN 3-540-35173-6, 978-3-540-35173-3. Citado 4 vezes nas páginas 19, 49, 60 e 76.

KOWALSKI, R. A.; TONI, F. Abstract argumentation. *Artif. Intell. Law*, v. 4, n. 3-4, p. 275–296, 1996. Citado na página 38.

LÓPEZ, F. L. y; MARQUEZ, A. A. An architecture for autonomous normative agents. In: *ENC*. [S.l.]: IEEE Computer Society, 2004. p. 96–103. ISBN 0-7695-2160-6. Citado 4 vezes nas páginas 39, 40, 42 e 60.

LOUALI, R. et al. Real-time characterization of microsoft flight simulator 2004 for integration into hardware in the loop architecture. In: IEEE. *Control & Automation (MED), 2011 19th Mediterranean Conference on*. [S.l.], 2011. p. 1241–1246. Citado na página 105.

LóPEZ, F. L. Y. *Social Power and Norms: Impact on Agent Behaviour*. [S.l.], 2003. Citado 2 vezes nas páginas 44 e 45.

MENEGUZZI, F.; LUCK, M. Norm-based behaviour modification in BDI agents. In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. [S.l.]: International Foundation for Autonomous Agents and Multiagent Systems, 2009. p. 177–184. Citado 5 vezes nas páginas 19, 54, 55, 57 e 59.

MILL, J.; SHER, G. *Utilitarianism*. [S.l.]: Hackett Pub., 2001. (Classics Series). ISBN 9780872206052. Citado na página 46.

MOREIRA, M. *Teorias de aprendizagem*. [S.l.]: Editora Pedagógica e Universitária, 1999. ISBN 9788512321400. Citado na página 32.

PACHECO, N. C. *USING NORMS TO CONTROL OPEN MULTI-AGENT SYSTEMS*. Tese (Tesis Doctoral en Informática) — Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 2012. Citado 2 vezes nas páginas 24 e 60.

RAO, A. S.; GEORGEFF, M. P. BDI agents: From theory to practice. In: *IN PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS (ICMAS-95*. [S.l.: s.n.], 1995. p. 312–319. Citado 2 vezes nas páginas 28 e 64.

RAWLS, J. *Two Concepts of Rules*. [S.l.]: Ardent Media, Incorporated, 1955. (Bobbs-Merrill reprint series in philosophy). ISBN 9780829026016. Citado na página 23.

RIEMSDIJK, M. B. van et al. Dynamics of declarative goals in agent programming. In: *Proceedings of the Second international conference on Declarative Agent Languages and Technologies*. Berlin, Heidelberg: Springer-Verlag, 2005. (DALT'04), p. 1–18. ISBN 3-540-26172-9, 978-3-540-26172-8. Citado na página 51.

SADRI, F.; STATHIS, K.; TONI, F. Normative kgp agents. *Computational & Mathematical Organization Theory*, v. 12, n. 2-3, p. 101–126, 2006. Citado 2 vezes nas páginas 19 e 36.

SANTOS, F. R.; HÜBNER, J. F.; BECKER, L. B. Concepção e análise de um modelo de agente bdi voltado para o planejamento de rota em um vant. In: *Anais do IX Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações – IX WESAAC*. [S.l.: s.n.], 2015. p. 66–77. ISBN 2177-2096. Citado 4 vezes nas páginas 93, 94, 95 e 109.

SEARLE, J. *Speech Acts: An Essay in the Philosophy of Language*. [S.l.]: Cambridge University Press, 1969. (Cam: [Verschiedene Aufl.]). ISBN 9780521096263. Citado na página 23.

SEARLE, J. R. *The Construction of Social Reality*. [S.l.]: Free Press, 1997. ISBN 0684831791. Citado na página 23.

SHOHAM, Y. Agent-oriented programming. *Artif. Intell.*, Elsevier Science Publishers Ltd., Essex, UK, v. 60, n. 1, p. 51–92, mar. 1993. ISSN 0004-3702. Citado na página 26.

SICHMAN, J. S. et al. A social reasoning mechanism based on dependence networks. In: . [S.l.]: John Wiley and Sons, 1994. p. 188–192. Citado 2 vezes nas páginas 48 e 50.

THAGARD, P. *Coherence in thought and action*. [S.l.]: M I T PRESS (MA), 2002. (A Bradford book). ISBN 9780262700924. Citado 3 vezes nas páginas 58, 59 e 90.

THAYER, R. E. *The biopsychology of mood and arousal*. 1st. ed. [S.l.]: Oxford University Press, 1989. ISBN 0195068270. Citado 5 vezes nas páginas 11, 28, 29, 67 e 70.

THAYER, R. E. *The origin of everyday moods: Managing energy, tension and stress*. 1st. ed. [S.l.]: Oxford University Press, 1996. ISBN 0756756901. Citado 2 vezes nas páginas 28 e 67.

TONI, F. Assumption-based argumentation for selection and composition of services. In: *In: Proceedings of the 8th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA VIII)*. [S.l.: s.n.], 2007. Citado na página 38.

TORRE, L. W. N. van der; TAN, Y.-H. Diagnosis and decision making in normative reasoning. *Artif. Intell. Law*, v. 7, n. 1, p. 51–67, 1999. Citado na página 18.

UEZ, D. M. *Método para o Desenvolvimento de Software Orientado a Agentes Considerando o Ambiente e a Organização*. Dissertação (Mestrado) — UFSC - Universidade Federal de Santa Catarina, 2013. Citado 3 vezes nas páginas 97, 109 e 110.

WHISNER, W. A new theory of emotion: Moral reasoning and emotion. *Philosophia*, Springer, v. 54, n. 1, p. 3–30, 2003. Citado na página 17.

WOOLDRIDGE, M. *An Introduction to MultiAgent Systems*. 1st. ed. [S.l.]: John Wiley & Sons, 2002. Paperback. ISBN 047149691X. Citado na página 39.

WOOLDRIDGE, M.; JENNINGS, N. R. Agent theories, architectures, and languages: a survey. In: *Proceedings of the workshop on agent theories, architectures, and languages on Intelligent agents*. New York,

NY, USA: Springer-Verlag New York, Inc., 1995. (ECAI-94), p. 1–39. ISBN 3-540-58855-8. Citado 3 vezes nas páginas 25, 26 e 27.