Huginn for Jason: Uma arquitetura para agentes normativos

Tiago L. Schmitz [1]
Jomi F. Hübner [1]

Data de submissão: 20.01.2015
Data de aceitação: 05.06.2015

Resumo: Uma classe de sistemas multiagente pode ser vista como uma sociedade na qual o sistema normativo orienta os agentes para atingirem os objetivos do sistema. Quando um agente percebe um sistema de normas, ele precisa raciocinar sobre o impacto dessas normas em seus objetivos 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. O modelo Huginn trata explicitamente do raciocínio normativo com recursos finitos propondo um processo de deliberação que traduz as normas, desejos e recursos para um problema de otimização conhecido como problema da mochila multidimensional com múltipla escolha. Este artigo apresenta a teoria e implementação do modelo Huginn.

Abstract: There is one multi agent class that can be viewed as a society where the normative system guides the agents to achieve the system goals. When an agent perceives a normative system, it needs to reason about the norms impact on its desires. Considering that the agent's resources are limited, it is necessary to reason about the resources available and if they are enough to achieve a desire or norm. The Huginn is a model to reason about norms, desires and limited resources. The deliberation process of this model translate the norms, desires and resources into an optimization problem (multiple-choice multidimensional knapsack problem). This paper introduces the Huginn model: theory and implementation.

1 Introdução

Em sistemas multiagente abertos, nos quais os agentes podem entrar e sair do mesmo a qualquer momento, é importante coordenar as ações dos agentes para poder atingir os ob-

[1]Departamento de Engenharia e Automação, UFSC, Caixa Postal 476
Florianópolis, SC, CEP 88040-900
{tiagolschmitz@gmail.com, jomi.hubner@ufsc.br}

jetivos da aplicação. O sistema normativo pode ser utilizado para esta finalidade, regulando o comportamento de um agente através de permissões, proibições e obrigações.

A capacidade de compreender e interpretar os sistemas normativos é uma característica desejável para os agentes presentes nesse tipo de sistema. Um agente normativo precisa resolver conflitos entre desejos (de caráter individual) e normas (de caráter coletivo). Às vezes, estes agentes têm o desejo de alcançar um objetivo específico, mas uma norma os proíbe de atingir esse objetivo. Por exemplo, um agente deseja chegar ao topo da colina e, ao mesmo tempo, uma norma o proíbe de chegar ao topo. Este agente precisa ponderar o que é melhor para ele e para o grupo.

No caso específico onde os agentes normativos têm recursos limitados, eles necessitam raciocinar também sobre os recursos disponíveis. Às vezes, um agente pode concordar com a norma mas não tem recursos suficientes para alcançar o objetivo implicado pela norma ou a recompensa da norma não justifica o recurso necessário.

Dentre os modelos propostos para agentes normativos, o modelo Huginn destaca-se por tratar explicitamente da utilização de recursos finitos no raciocínio de desejos e normas. O Huginn é um modelo teórico que traduz o processo de deliberação para um problema de otimização conhecido como problema da mochila multidimensional com múltiplas escolhas. Este modelo é descrito na seção 3. Este artigo apresenta o modelo conceitual e a implementação do Huginn. Para tal os objetivos deste artigo são:

- Apresentar o modelo Huginn identificando os elementos que compõem o raciocínio normativo. Na seção 3 é apresentado brevemente a inspiração deste modelo e como o raciocínio é composto.

- Identificar os elementos do Huginn e atribuir a estes uma representação dentro do framework JaCaMo[2] [1]. Na seção 4.1 é apresentado uma série de artefatos, crenças e notações de objetivos para representar os elementos constituintes do modelo Huginn.

- Alterar a arquitetura do agente Jason[3] [2] para que este possa identificar os elementos do Huginn e assim executar o processo de deliberação. Na seção 4.2 são apresentados os elementos adicionados à arquitetura de agente do Jason e os algoritmos envolvidos.

Este é um trabalho em andamento e os resultados apresentados neste artigo dizem respeito a concepção prática do modelo Huginn. Futuramente, estão previstas avaliações comparativas entre diferentes modelos normativos.

[2]http://jacamo.sourceforge.net/
[3]http://jason.sourceforge.net/

2 Conceitos preliminares

Antes de apresentar o modelo Huginn é necessário apresentar o modelo de normas e desejos empregados. As subseções 2.1 e 2.2 descrevem os elementos que compõem as normas e desejos.

2.1 Normas

Normas são mecanismos de coordenação com o objetivo de reforçar comportamentos adequados e inibir comportamentos inadequados dentro de uma organização. Na sociologia existem trabalhos [3, 4, 5] que descrevem diferentes tipos de normas em um mesmo sistema normativo. Especificamente nesse trabalho, serão tratadas as normas deônticas, as quais são expressas em termos de obrigações, permissões e proibições. As definições de cada uma dessas modalidades segue um padrão definido. Obrigações são objetivos globais[4] que devem ser atingidos por um agente. Proibições definem estados globais que, quando atingidos, estão em desacordo com os objetivos globais do sistema. 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 a norma será instanciada. A condição de expiração define o período de validade da norma.

Outro elemento da norma é o mecanismo de reforço. Nesse mecanismo, a norma ganha dois componentes a sanção e a recompensa. A sanção corresponde a punição que o agente recebe por não cumprir a norma e a recompensa é o reforço positivo recebido pelo agente por tê-la cumprido. A definição formal para as normas deônticas utilizadas nesse trabalho é uma tupla:

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

na qual $D \in \{O, F, P\}$ é a modalidade deôntica da norma, obrigação (O), proibição (F) e permissão (P); 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 o 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:

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

[4] objetivos globais são objetivos do sistema multiagentes. Objetivos locais são objetivos individuais do agente.

2.2 Desejos

Os desejos serão tratados nesse trabalho como as vontades do agente. Os desejos, diferentemente das normas, tem sua origem no próprio agente, representando assim um caráter pessoal. 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 alimentado. Essa condição geraria uma tupla:

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

2.3 Recursos

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

$$\langle r, Q \rangle$$

Por exemplo, um agente que possui 5 litros (Q) de combustível (r). Essa condição geraria uma tupla:

$$\langle combustivel, 5 \rangle$$

3 Modelo de raciocínio normativo baseado em ânimo (Modelo Huginn)

Para considerar os recursos do agente, o modelo de raciocínio proposto é baseado no conceito psicológico de estado anímico [6]. Thayer considera que o estado anímico é composto por três elementos, a energia, a tensão e o benefício [7]. A energia corresponde ao

estado físico do indivíduo (cansado ou disposto). A tensão corresponde ao estado psíquico do indivíduo (calmo ou estressado). O benefício corresponde ao ganho do indivíduo por atingir um objetivo.

No Huginn a tensão é o feedback negativo de uma norma ou desejo, o benefício é o feedback positivo de uma norma ou desejo e a energia é a quantidade de recursos que o agente dispõe. A energia é vista como um espaço d-dimensional, no qual cada dimensão representa um recurso. Por exemplo, um agente possui 3 horas de bateria logo ele possui um espaço unidimensional de tamanho 3 que será utilizado para minimizar a tensão e maximizar o benefício.

No modelo proposto nesse artigo o raciocínio é dado por um ciclo que tem como entrada os desejos, as normas e os recursos. Nessa entrada, os desejos são de ordem pessoal do agente, as normas são de ordem social e os recursos são os elementos fornecidos pelo ambiente. Essa entrada é analisada por um processo de maximização do benefício e minimização da tensão, levando em consideração os recursos disponíveis. Como resultado é obtido um conjunto de objetivos e normas não conflitantes que serão cumpridos (conjunto M). Esse conjunto é recalculado toda vez que ocorre uma revisão de objetivos e normas. Os eventos que disparam essa revisão são três: expiração, cumprimento e percepção.

O primeiro evento que tem potencial de disparar o processo de revisão é a **expiração** do prazo de uma norma ou de um desejo. Quando o desejo ou norma expira, este não é mais viável para o estado atual do agente. Nessa situação, a revisão é disparada, formando um novo conjunto de desejos e normas a serem cumpridos (conjunto M).

O segundo evento que dispara o processo de revisão é o **cumprimento** de uma norma ou de um desejo. Quando um desejo ou norma é cumprido esse deixa de fazer parte do conjunto M.

O terceiro evento que dispara o processo de revisão é a **percepção** de uma nova norma ou desejo pelo agente. Nessa situação, a revisão é disparada, formando um novo conjunto M.

A figura 1 ilustra o processo de revisão proposto. O hexágono representa o processo de revisão normativa, os retângulos representam conjuntos, e a caixa tracejada representa a definição de energias, tensão e benefício. A figura 1 apresenta cinco conjuntos (R, D, N, O, M), onde R é composto por todos os tipos de recursos disponíveis para o agente, D é o conjunto de todos os desejos, N é composto por todas as normas ativas, O é o conjunto de objetivos implicadas pelos desejos e normas em D e N, por fim, M é o conjunto de objetivos que maximiza os ganhos do agente.

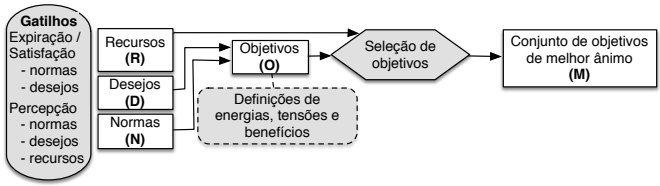

Figura 1. Fluxograma do processo de deliberação

3.1 Revisão de objetivos e normas

A revisão pode ser divida em duas fases. Na primeira fase o agente considera as normas, desejos e recursos. Para cada norma e desejo é criado um objetivo. Esses objetivos e o conjunto de objetivos vigentes (M) formam o conjunto O que contém todos os objetivos potenciais do agente.

Nesta mesma fase são definidos o benefício, os recursos necessários e a tensão para o cumprimento de cada elemento de O. Para determinar o **benefício** de um objetivo normativo é utilizado a recompensa da norma. Já para os objetivos obtidos pelos desejos, o benefício corresponde a **intensidade** do desejo. Os recursos requeridos correspondem as quantidades de **recursos** que serão utilizadas para atingir o objetivo. Por fim, para determinar a **tensão** de um objetivo normativo é utilizado a **sanção** da norma e no caso de um objetivo obtido pelo desejo a **tensão** corresponde a **necessidade** do desejo. De forma genérica, pode-se descrever o recurso, a tensão e o benefício como as funções (1)(2)(3), respectivamente.

$$resource_{req}(o, r) = \text{ recursos } r \text{ requeridos para cumprir } o \tag{1}$$

$$tension(o) = \begin{cases} sanction(o) & \text{se } o \text{ deriva de norma} \\ necessity(o) & \text{se } o \text{ deriva de desejo} \end{cases} \tag{2}$$

$$benefit(o) = \begin{cases} reward(o) & \text{se } o \text{ deriva de norma} \\ intensity(o) & \text{se } o \text{ deriva de desejo} \end{cases} \tag{3}$$

$resource_{req}(o, r)$ é a função que determina a quantidade de recurso r necessário para cumprir objetivo o; $tension(o)$ é a função que determina a tensão do objetivo o; $benefit(o)$ é a função que determina qual o benefício do objetivo o; $sanction(o)$, $necessity(o)$, $reward(o)$ e $intensity(o)$ são funções que retornam, respectivamente, o valor da sanção, da necessidade, da recompensa e da intensidade do objetivo o. Por exemplo, uma norma n_1 que obriga o agente a atingir o objetivo o_1 tem uma sanção 100 e uma recompensa 150, contudo para ser cumprida consome 5 unidades de combustível. Obtemos uma $resource_{req}(o_1, combustivel) = 5$ e uma $tension(o_1) = 100$ e um $benefit(o_1) = 150$.

3.2 Deliberação sobre normas e objetivos

Nessa fase da revisão o agente delibera sobre a aceitação dos objetivos contidos em O. Durante a vida do agente é possível que ele encontre conflitos entre esses elementos. Para tratar esses conflitos a maioria dos trabalhos relacionados usa critérios de prioridade. Logo, uma norma ou objetivos dito mais prioritário sempre será executado mesmo que sub-utilize a capacidade do agente. Para que isso ocorra é necessário tratar os conflitos entre objetivos e os conflitos de recursos.

O problema do conflito entre objetivos pode ser mapeado para um problema da mochila com múltipla escolha [8]. Nesse problema, os itens são divididos em classes e 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, a lanterna e a vela simultaneamente, liberando espaço para itens das outras classes. Na resolução de conflito entre os objetivos potenciais (O), cada classe c corresponderá a um conflito de objetivos. Por exemplo, o objetivo p e o objetivo $\neg p$ pertencem a uma mesma classe $c = \{p, \neg p\}$, o que garante que apenas um dos objetivos conflitantes será escolhido.

Durante o ciclo de vida do agente é possível que o mesmo possua objetivos que necessitem do mesmo recurso, contudo este recurso é insuficiente para suprir a demanda dos objetivos. Caracterizando um conflito entre os objetivos pelo recurso. Este conflito pode ser mapeado em um problema da mochila multidimensional [8], onde cada dimensão da mochila corresponde a um recurso disponível e os objetivos assumidos não podem ultrapassar a quantidade de recursos disponíveis.

O ganho de um objetivo é calculado somando-se o benefício e a tensão. Isso se deve ao fato de que, ao cumprir um objetivo, o agente recebe o benefício relativo a esse objetivo e evita a tensão de não cumpri-lo. Dessa maneira a função (4) é a função a ser maximizada neste problema.

$$Maximize \sum_{c \in C} \sum_{o \in c} m_o(benefit(o) + tension(o))) \tag{4}$$

Na função (4), C é o conjunto de classes de conflitos entre objetivos; c é o conjunto de objetivos de um conflito c; m_o é uma variável de decisão com valores pertencentes ao conjunto $\{0, 1\}$, sendo 1 objetivo aceito e 0 objetivo recusado.

As restrições desse problema de otimização serão dependentes do número de recursos

disponíveis. Para cada um dos recursos será gerada uma restrição como a equação (5).

$$\sum_{c \in C} \sum_{o \in c} (m_o \times resource_{req}(o, r)) \leq resourceAvailable(r), r \in R \tag{5}$$

Essa equação representa que a soma dos recursos requeridos de todos objetivos aceitos tem que ser menor ou igual a quantidade de recurso disponível ($resourceAvailable(r)$). Por fim, para garantir que apenas um dos objetivos de cada classe seja escolhido a restrição (6) é adicionada ao sistema:

$$\sum_{o \in c} (m_o) \leq 1, c \in C \tag{6}$$

Por exemplo, um caso no qual há apenas um recurso (gas), têm-se uma modelagem do problema:

$$Maximize \sum_{c \in C} \sum_{o \in c} (m_o \times (benefit(o) + tension(o))) \tag{7}$$

Sujeito a

$$\sum_{c \in C} \sum_{o \in c} (m_o \times resource_{req}(o, gas)) \leq resourceAvailable(gas) \tag{8}$$

$$\sum_{o \in c} m_o \leq 1, c \in C \tag{9}$$

3.3 Exemplo

Para ilustrar o modelo de raciocínio utilizaremos o exemplo a seguir. Considerando um agente com 10 litros de gasolina e um conjunto de normas e desejos compostos por:

- $N_1 = \langle O, \neg r, agente1, a, \neg a, 2, 3 \rangle$
- $N_2 = \langle O, r, agente1, a, \neg a, 2, 7 \rangle$
- $N_3 = \langle O, p, agente1, a, \neg a, 1, 2 \rangle$
- $D_1 = \langle q, \neg a, 2, 3 \rangle$

No primeiro momento, o agente percebe esse conjunto de normas e desejos e os interpreta como objetivos do conjunto O, no qual será calculado o seu **benefício, energia e tensão**. Para definir a energia o agente verifica a quantidade necessária de cada recurso para cumprir os objetivos. Nesse caso, o recurso requerido para cumprir para cada objetivo é: 3

litros para $\neg r$; 10 litros para r; 8 litros para p; 7 litros para q, obtendo dessa forma os valores descritos na tabela 1.

Or.	Obj.	Tensão	Benefício	Recurso requerido
N_1	$\neg r$	$tension(\neg r) = 2$	$benefit(\neg r) = 3$	$resource_{req}(\neg r, gas.) = 3$
N_2	r	$tension(r) = 2$	$benefit(r) = 7$	$resource_{req}(r, gas.) = 10$
N_3	p	$tension(p) = 1$	$benefit(p) = 2$	$resource_{req}(p, gas.) = 8$
D_1	q	$tension(q) = 2$	$benefit(q) = 3$	$resource_{req}(q, gas.) = 7$

Tabela 1. Estado inicial do conjunto O

O conjunto $O = \{\neg r, r, p, q\}$ é utilizado como entrada no algoritmo de otimização, no qual obtemos 3 classes de conflitos: $C_1 = \{\neg r, r\}$, $C_2 = \{p\}$ e $C_3 = \{q\}$

Após a execução do algoritmo é gerado um conjunto $M = \{\neg r, q\}$ que consumirá 10 litros de gasolina. Apesar do objetivo r possuir a maior benefício ele não foi selecionado, pois o conjunto M é capaz de com o mesmo recurso conquistar um ganho maior.

Após alguns ciclos de execução o agente percebe uma nova norma N_4 com recompensa 6, tensão 2, energia 2 e objetivo s. Essa nova norma dispara a revisão de objetivos e normas. Como alguns **recursos já foram consumidos** por $\neg r$ e q, os **recursos requeridos são recalculado**. $\neg r$ consumiu 1 litro e q consumiu 2. Logo, os recursos requeridos passam a ser, respectivamente, 2 e 5 e o recurso disponível passa a ser 7 litros. Essa situação vai gerar um novo conjunto O representado na tabela 2.

Or.	Obj.	Tensão	benefício	Recurso requerido
N_1	$\neg r$	$tension(\neg r) = 2$	$benefit(\neg r) = 3$	$resource_{req}(\neg r, gas.) = 2$
N_2	r	$tension(r) = 2$	$benefit(r) = 7$	$resource_{req}(r, gas.) = 10$
N_3	p	$tension(p) = 1$	$benefit(p) = 2$	$resource_{req}(p, gas.) = 8$
D_1	q	$tension(q) = 2$	$benefit(q) = 3$	$resource_{req}(q, gas.) = 5$
N_4	s	$tension(s) = 2$	$benefit(s) = 6$	$resource_{req}(s, gas.) = 2$

Tabela 2. Estado do conjunto O após a adição de N_4

Após executar o algoritmo de otimização, obtemos um novo conjunto $M = \{q, s\}$. 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 ganho maior.

4 Implementação da arquitetura de agente baseada no Huginn

Este artigo apresenta uma proposta de implementação do Huginn através da extensão da plataforma Jason [2]. 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). O fato de fazer parte do framework JaCaMo foi outro fator que motivou a escolha. Inicialmente, a implementação proposta fará uso apenas do JaCa (Jason e CArtAgO [9])[5].

A descrição da proposta de implementação ocorrerá em duas partes. A representação das informações requeridas pelo Huginn (subseção 4.1) e as alterações na arquitetura do agente para que o processo de deliberação do Huginn seja possível (subseção 4.2).

4.1 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 possui dois artefatos com os quais apenas o agente é capaz de interagir. Este artefatos representam os recursos do agente e as normas que o agente deve obedecer. Os desejos por sua vez são de natureza interna do agente e, como tal, são representados por crenças.

Os recursos são considerados de uso pessoal do agente e, portanto, os agentes não compartilham os mesmos recursos. Para representar os recursos disponíveis, cada agente possui um artefato composto por uma propriedade observável para cada recurso. Esta propriedade é representada em (10).

$$huginn\_\_resource(resourceType, quantity) \tag{10}$$

O artefato de normas por sua vez possui apenas propriedades observáveis referentes às obrigações e proibições do agente. A propriedade observável identifica 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 o artefato apresenta a norma como uma propriedade observável quando a mesma está ativa, sendo assim, quando o agente percebe a norma ela está sempre ativa. A assinatura dessas propriedades é representada por (11). Esta propriedade possui cinco termos: $goal$ representando o objetivo, $expiration$ representando a condição de expiração, $saction$ é o valor da sanção, $reward$ é o valor da recompensa e $normType$ é o tipo da norma.

$$norm(goal, expiration, saction, reward, normType) \tag{11}$$

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 desta crença é dada por (12). A assinatura

[5]Neste primeiro momento, não se utilizou o Moise porque ele segue uma abordagem onde tudo é proibido exceto o que é explicitamente permitido, já o Huginn considera que tudo é permitido menos o que é proibido.

dessas propriedades é representada por (11). Esta propriedade 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.

$$desire(goal, expiration, necessity, intensity) \tag{12}$$

Além dos três eixos do processo de deliberação o Huginn especifica que os objetivos possuem três tipos de propriedades: a tensão, o benefício e os recursos necessários. Conforme descrito na seção 3, 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 definida pelo projetista do agente. 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 o agente 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. Esta 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 (13).

$$huginn\_\_resource\_required(goal, resourcetype, quantity) \tag{13}$$

No momento da deliberação estas 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 (14) representa a dinâmica do recurso gasolina para o objetivo $goto(X1, Y1)$

$$\begin{aligned} huginn\_\_resource\_required(goto(X1, Y1), battery, B) :- \\ position\ (X0, Y0) \\ \&B = math.max(math.abs(X1 - X0), math.abs(Y1 - Y0)) * 2. \end{aligned} \tag{14}$$

Este artigo, em sua proposta de implementação, 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 1 apresenta um plano genérico para tradução de normas em objetivos. Este plano é apresentado na linguagem Jason e representa que toda percepção de norma, irá criar um objetivo GOAL com as seguintes anotações:

- a condição de expiração (`cexp`) contendo a expiração da norma (`EXPIRATION`);
- 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 (`NORMTYPE`).

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

Plano 1. Plano genérico para tradução de normas em objetivos

O plano 2 exemplifica um caso específico de como as normas são traduzidas em objetivos. Neste plano o agente tem a obrigação de obter pedras até que não tenha 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(~stones), tension(0.5),
4                       benefit (0.4), typesource(o)].
```

Plano 2. Exemplo de percepção de norma

Os desejos são traduzidos de forma similar às normas. O plano 3 apresenta um plano genérico para tradução de desejos em objetivos. Este plano representa 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:

- a condição de expiração (`cexp`) contendo a expiração do desejo (`EXPIRATION`);

- 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), tension(NECESSITY),
4              benefit (INTENSITY), typesource(d)].
```

Plano 3. Plano genérico para tradução de desejos em objetivos

O plano 4 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 deste 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)].
```

Plano 4. 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 estes conflitos entre objetivos são detectados. Considerando como premissa que existem trabalhos que tratam da detecção de conflitos [10, 11], 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 (15).

$$huginn\_\_conflict(goto(beach), work(thesis)). \tag{15}$$

A figura 2 resume os itens abordados nesta seção, na qual tratamos:

- a representação dos três eixos do Huginn: os desejos, as normas e os recursos;
- a tradução das normas e desejos em objetivos, passando pela identificação da tensão, do benefício e dos recursos requeridos;
- a representação de conflitos.

Estes itens serão utilizados na arquitetura apresentada na seção 4.2 para compor efetivamente o raciocínio normativo.

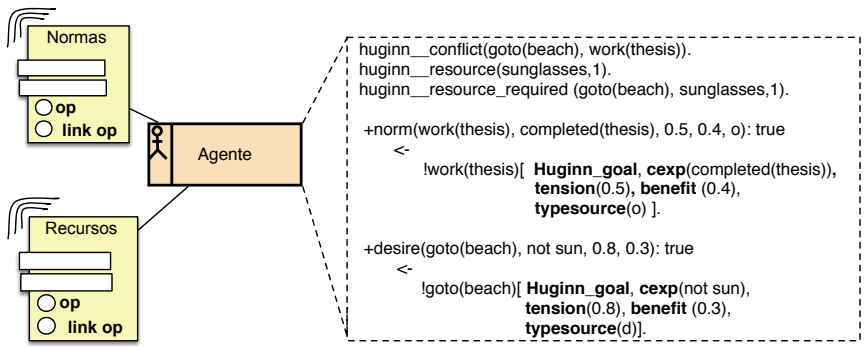

Figura 2. Exemplo de codificação de um agente normativo

4.2 Extensão da arquitetura de agente do Jason

A implementação proposta neste artigo estende a arquitetura de agente da plataforma Jason [2]. A arquitetura estendida do Jason é representada na figura 3. O funcionamento do Huginn é baseado em eventos que quando ocorridos disparam o processo de deliberação normativa. Os eventos podem ser capturados na função $selectEvent$ (S_E) da arquitetura do Jason, destacado no losango 5 da figura 3. Estes eventos podem conter a adição e remoção de crenças e objetivos.

Na função $selectEvent$ proposta, os eventos recebido como entrada são filtrados de forma a 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 3 pelo retângulo 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

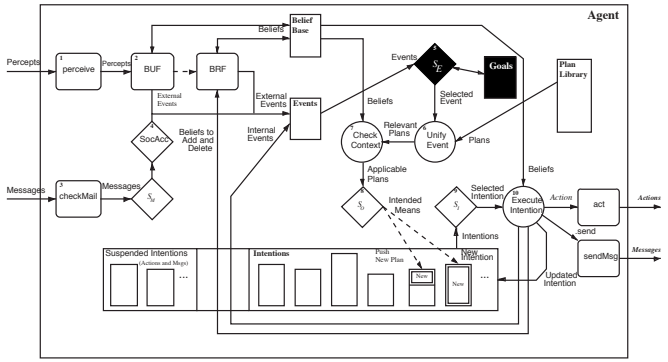

Figura 3. Arquitetura de agente na plataforma do Huginn

no conjunto de objetivos, a função $selectEvent$ segue o algoritmo 1, onde f_o é o objetivo representado pelo evento f, o_f é o evento que representa o objetivo o e f_r é o evento que representa a adição ou subtração do recurso r.

Para descrever em detalhes o processo de seleção proposto o mesmo foi dividido em duas partes: eventos que o disparam e a seleção dos objetivos.

Dois tipos de eventos disparam o processo de deliberação do Huginn: a percepção, o cumprimento e a expiração. Para identificá-los a fila de eventos recebida como parâmetro da função *selectEvent* é percorrida. Caso alguma das situações a seguir ocorra o processo de deliberação será disparado.

- Percepção de um novo desejo ou norma. Para tal, existirá um evento contendo um objetivo que provém de um desejo ou norma. Um novo objetivo é percebido quando o objetivo presente no evento não está presente no conjunto O. Neste caso o novo objetivo é adicionado ao conjunto O e será executado o processo de deliberação. Esta situação é representada nas linhas 4 à 6 do algoritmo 1.

- Percepção de adição de recurso. Esta situação corresponde a encontrar um evento de adição de crença que aumente a quantidade de recurso disponível. Para tanto o processo consulta a base de crenças do agente para comparar se o valor adicionado é maior que o existente. Caso o recurso não esteja presente na base de crenças a simples adição da crença dispara o processo de deliberação. Esta situação é representada nas linhas 7 e 8 do algoritmo 1.

- Expiração de um objetivo. Esta situação ocorre quando um evento possui um objetivo com a condição de expiração satisfeita. Para identificar esta condição o processo verifica na base de crenças se a condição é verdadeira. Neste caso o objetivo é removido

Algoritmo 1: Função selectEvent

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

```
1  newDeliberation=false;
2  F_m = F_i;
3  for each (f ∈ F_i) do
4      if f_o ∉ O then
5          O := O ∪ {f_o};
6          newDeliberation=true;
7      if f_r é adição de recursos then
8          newDeliberation=true;
9  for each (o ∈ O) do
10     if condição de expiração de o é verdadeira then
11         O := O\{o};
12         newDeliberation=true;
13 if newDeliberation then
14     M := deliberation(O, C, R) ;
15     for each (f ∈ F_m) do
16         if f_o ∉ M then
17             F_m = F_m\{f_o};
18     for each (o ∈ M) do
19         if o_f ∉ F_m then
20             F_m := F_m ∪ {o_f};
21 return F_m
```

do conjunto O e é executado o processo de deliberação. Esta situação é representada nas linhas 9 à 12 do algoritmo 1.

Quando uma destas três situações ocorre o processo de deliberação normativa (linha 14 do algoritmo 1) é executado, selecionando os objetivos que oferecem o melhor ânimo. Após selecionado os objetivos, os eventos que contêm objetivos que não foram selecionados são removidos da lista de eventos (linhas 15 à 17 do algoritmo 1) e os objetivos selecionados que não possuem evento na lista tem seu evento adicionado à mesma (linhas 18 à 20 do algoritmo 1).

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 múlti-

pla escolha, como descrito ns seção 3. 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 deste problema foi utilizado o solver **GLPK**. O GLPK[6] (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 [8], 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 Trabalhos relacionados

Existem diversos trabalhos sobre programação normativa e plataformas para desenvolvimento de sistemas multiagente normativos e alguns permitem ao agente deliberar sobre o cumprimento de normas. Vários destes trabalhos são teóricos dentre eles podemos citar o **BOID** [12] que é um dos precursores do assunto. O BOID estende o conceito de BDI [13] declarando explicitamente as obrigações e criando um sistema de preferências para selecionar as normas ou desejos mais relevantes. Todavia o mesmo não apresenta nenhuma implementação. Outros modelos de agente normativo que possuem implementação são o **Normative KPG** [14] e o **N-2APL**[15].

O Normative KPG estende o modelo KPG (*knowledge-goal-plan*) adicionando noções sobre obrigações, proibições e papéis. O modelo considera relevante todas as normas que afetam o papel desempenhado por ele, portanto em caso de conflito com objetivos pessoais a norma sempre prevalece. Logo, o agente não é capaz de descumprir uma norma.

O N-2APL é uma linguagem de programação baseada no paradigma BDI que suporta agentes capazes de raciocinar sobre as normas. A N-2APL adiciona à linguagem 2APL [16] 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 arquitetura de implementação apresentada neste trabalho apresenta similaridades com outras arquiteturas N-BDI. O diferencial desta arquitetura é que ela implementa o Hu-

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

ginn que trabalha com recursos limitados para deliberar sobre desejo e normas. Sendo que os recursos limitados não são considerados nos trabalhos relacionados. A tabela 3 resume os elementos tratados em cada modelo.

Modelo	Itens tratados pelo modelo					
	Obrigação	Proibição	sanção	benefício	Recurso	prazo
BOID	X					
N. KPG	X	X				
N-2APL	X	X	X			X
Huginn	X	X	X	X	X	

Tabela 3. Tabela comparativa dos modelos

6 Considerações finais

O modelo proposto nesse trabalho é capaz de raciocinar sobre os objetivos e normas, considerando o estado anímico. Essa característica permite ao agente gerenciar seus recursos de forma a maximizar o ganho que esses podem trazer. Tal característica é bastante desejada na utilização de agentes com recursos finitos.

Os modelos analisados [12, 14, 15] fazem uso de um sistema de preferências para solucionar conflitos. Esse trabalho, por sua vez busca formar um conjunto consistente de normas e objetivos que ofereça a maior ganho ao agente, permitindo ao mesmo adotar objetivos com menos prioridade, mas, que em conjunto, terão uma recompensa maior.

O presente trabalho apresenta uma arquitetura de implementação para o modelo Huginn. Para desenvolver esta arquitetura foi necessário:

- Identificar os elementos do Huginn e atribuir a estes uma representação dentro do ambiente JaCaMo [1]. Neste artigo foi apresentado o uso de artefatos para representar os recursos limitados o agente. Além de crenças e notações de objetivos para representar os elementos constituintes dos desejos, normas.

- Alterar a arquitetura do agente Jason para que este possa identificar os elementos do Huginn e assim executar o processo de deliberação. Neste artigo foi apresentado os processos que são alterados para que a arquitetura de agente do Jason seja capaz de fazer uma deliberação normativa como prevê o Huginn.

Este é um trabalho em desenvolvimento e podemos identificar alguns trabalhos futuros. O primeiro é a conclusão da implementação desta arquitetura para que os futuros usuários possam fazer testes e verificar a capacidade de raciocínio do agente. O segundo é

a avaliação da performance (memória e CPU) da implementação com a finalidade de refinar os algoritmos utilizados na arquitetura. O terceiro é adicionar funções de comprometimento que facilitem ao programador criar objetivos que ganhem importância durante a sua obtenção. O último é raciocinar sobre as normas fornecidas pelo Moise, atendendo assim todo o framework JaCaMo.

7 Referências

[1] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi, "Multi-agent oriented programming with jacamo," *Science of Computer Programming*, vol. 78, no. 6, pp. 747–761, 2013.

[2] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. Wiley-Interscience, 2007.

[3] J. Rawls, *Two Concepts of Rules*, ser. Bobbs-Merrill reprint series in philosophy. Ardent Media, Incorporated, 1955. [Online]. Available: http://books.google.com.br/books?id=9UV3AAAACAAJ

[4] J. Searle, *Speech Acts: An Essay in the Philosophy of Language*, ser. Cam: [Verschiedene Aufl.]. Cambridge University Press, 1969. [Online]. Available: http://books.google.com.br/books?id=t3_WhfknvF0C

[5] J. R. Searle, *The Construction of Social Reality*. Free Press, 1997.

[6] R. E. Thayer, *The biopsychology of mood and arousal*, 1st ed. Oxford University Press, 1989.

[7] ——, *The origin of everyday moods: Managing energy, tension and stress*, 1st ed. Oxford University Press, 1996.

[8] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.

[9] A. Ricci, M. Viroli, and A. Omicini, "Cartago: A framework for prototyping artifact-based environments in mas," in *Proceedings of the 3rd International Conference on Environments for Multi-agent Systems III*, ser. E4MAS'06. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 67–86. [Online]. Available: http://dl.acm.org/citation.cfm?id=1759343.1759348

[10] S. Joseph, C. Sierra, W. M. Schorlemmer, and P. Dellunde, "Deductive coherence and norm adoption," *Logic Journal of the IGPL*, vol. 18, no. 1, pp. 118–156, 2010.

[11] P. Thagard, *Coherence in thought and action*, ser. A Bradford book. M I T PRESS (MA), 2002. [Online]. Available: http://books.google.com.br/books?id=Px0vctI8eGQC

[12] J. Broersen, M. Dastani, J. Hulstijn, Z. Huang, and L. van der Torre, "The BOID architecture: conflicts between beliefs, obligations, intentions and desires," in *Proceedings of the fifth international conference on Autonomous agents*, ser. AGENTS '01. New York, NY, USA: ACM, 2001, pp. 9–16. [Online]. Available: http://doi.acm.org/10.1145/375735.375766

[13] A. S. Rao and M. P. Georgeff, "Bdi agents: From theory to practice," in *In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95*, 1995, pp. 312–319.

[14] F. Sadri, K. Stathis, and F. Toni, "Normative kgp agents," *Computational & Mathematical Organization Theory*, vol. 12, no. 2-3, pp. 101–126, 2006.

[15] N. Alechina, M. Dastani, and B. Logan, "Programming norm-aware agents," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, V. Conitzer, M. Winikoff, L. Padgham, and W. van der Hoek, Eds., Valencia, Spain, June 2012.

[16] M. Dastani, "2APL: a practical agent programming language," *Autonomous Agents and Multi-Agent Systems*, vol. 16, no. 3, pp. 214–248, 2008.