UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Elétrica e de Computação

Eduardo de Moraes Fróes

Construindo um Subsistema Motivacional para o Cognitive Systems Toolkit

Campinas

2017

Eduardo de Moraes Fróes

Construindo um Subsistema Motivacional para o Cognitive Systems Toolkit

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade de Campinas como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica na área da Engenharia de Computação.

Orientador: Prof. Dr. Ricardo Ribeiro Gudwin.

Este exemplar corresponde à versão final da tese defendida pelo aluno Eduardo de Moraes Fróes, e orientada pelo Prof. Dr. Ricardo Ribeiro Gudwin.

Campinas
2017

Agência(s) de fomento e nº(s) de processo(s): CAPES, 1551328

ORCID: https://orcid.org/0000-0003-3973-8697

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

F922c

Froes, Eduardo de Moraes, 1990-
Construindo um subsistema motivacional para o cognitive systems toolkit / Eduardo de Moraes Fróes. – Campinas, SP : [s.n.], 2018.

Orientador: Ricardo Ribeiro Gudwin.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Inteligencia artificial. 2. Ciencia cognitiva. 3. Agentes inteligentes. 4. Computadores - Simulação. I. Gudwin, Ricardo Ribeiro, 1967-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Building a motivational subsystem for the cognitive system toolkit
Palavras-chave em inglês:
Artificial inteligence
Cognitive science
Inteligent agents
Computers - Simulation
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Ricardo Ribeiro Gudwin [Orientador]
José Mario de Martino
Roberto José Maria Covolan
Data de defesa: 29-01-2018
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Eduardo de Moraes Froes **RA:** 159206
Data da defesa: 29/01/2018

Titulo da Dissertação: "Construindo um Subsistema Motivacional para o Cognitive Systems Toolkit".

Dissertation Title: "Building a Motivational Subsystem for the Cognitive System Toolkit".

Prof. Dr. Ricardo Ribeiro Gudwin (Presidente, FEEC/UNICAMP)
Prof. Dr. José Mario de Martino (FEEC/UNICAMP)
Prof. Dr. Roberto José Maria Covolan (IFGW/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Agradecimento

Primeiramente, agradeço a Deus e a Nossa Senhora de Aparecida por ter me dado saúde e inspiração para elaboração deste trabalho. Aos meus pais Eurico e Neuza, por ter me dado amor, carinho e o bem mais valioso da minha vida, a "Educação". Aos meus irmãos Caio e Marcel por serem a minha referência pessoal e profissional. A minha sobrinha Isadora, por ter feito o meu cotidiano mais feliz nos momentos mais difíceis.

Ao meu orientador Ricardo Ribeiro Gudwin, agradeço por ter me dado a oportunidade de fazer o mestrado. Muito obrigado professor, por confiar em mim e por ter me ajudado na elaboração dos artigos e no desenvolvimento deste trabalho. Agradeço aos meus amigos e colegas em geral. Aos meus amigos do laboratório "DCA", Wandemberg, Suelen e André, aos meus amigos da "Ala Negra", Celso, Raul, Zel, Espina, Alan e Lopes, aos meus amigos de Faculdade, Girotto, Guilherme, Henrique, Carlota e Queiroz, e aos meus amigos da "RedPipow", Freitas, Lucas, Alexandre, Paulo Marchi e Jair. Todos eles foram pessoas que me motivaram na busca da realização deste sonho.

Também agradeço a todos os professores que tive durante a minha vida acadêmica. Professores da Faculdade de Engenharia de Sorocaba - FACENS, da Faculdade de Tecnologia do Estado de São Paulo - FATEC/Itu e da Faculdade de Engenharia Elétrica e Computação da Universidade Estadual de Campinas - FEEC/UNICAMP.

Agradeço as agências de fomento a pesquisa, CAPES e FAPESP pela ajuda financeira durante o período do mestrado. Um agradecimento especial à empresa Ericsson Telecomunicações do Brasil S.A., por ter me dado a oportunidade para o desenvolvimento da arquitetura cognitiva MECA e também por ter financiado os meus estudos nesta universidade.

E por fim, gostaria de fazer um agradecimento especial, ao meu amor, a minha noiva Camila. Meu amor muito obrigado, pelo apoio, ajuda, carinho, amor, parceria e muita paciência. Obrigado por estar ao meu lado em todos os momentos, sendo eles bons ou ruins.

"Às vezes as pessoas que ninguém imagina que poderiam fazer algo são as que fazem coisas que ninguém poderia imaginar." - Alan Turing

Resumo

Motivações e emoções estão inseridas intrinsecamente na cognição e no comportamento de diversos tipos de animais, particularmente nos seres humanos. São responsáveis por apoiar a tomada de decisões, estimulando comportamentos diferentes de modo que suas necessidades internas sejam satisfeitas. Este trabalho propõe a concepção e implementação de um subsistema motivacional dotado de capacidades motivacionais e emocionais para o *Cognitive System Toolkit* (CST), um conjunto de ferramentas de software para computação cognitiva desenvolvido pelo nosso grupo de pesquisa, com base em estudos da literatura e diferentes implementações de sistemas motivacionais e emocionais em arquiteturas cognitivas conhecidas. Neste contexto, o Subsistema Motivacional foi submetido a um conjunto de simulações no *World Server 3D Application*. Neste simulador, é possível realizar simulações onde uma criatura artificial "vive" em um ambiente virtual onde pode encontrar comida, obstáculos e objetos de valor. Sobre esses objetos, a criatura pode realizar um conjunto de ações. No caso das comidas, o agente pode comê-las, guardá-las em sua sacola ou enterrá-las para que possam ser utilizadas posteriormente. Já para as joias, o agente pode capturá-las e guardá-las em sua sacola para que elas sejam trocadas por pontos no ponto de troca (*Delivery Spot*). Elas também podem ser enterradas, para evitar que a criatura colida com joias indesejáveis durante o processo de exploração do ambiente. No contexto dos obstáculos, eles são objetos inanimados que têm por objetivo dificultar a percepção e a exploração da criatura quando ela está em busca de objetos desejáveis. Para que se possa avaliar a eficiência e eficácia do modelo, foi desenvolvido um conjunto de experimentos com diferentes controladores inteligentes utilizando o modelo de subsistema motivacional, um sistema reativo e também um conjunto de arquiteturas cognitivas como: JSOAR, CLARION e LIDA. Os experimentos são executados durante o período de dez minutos e têm por objetivo fazer com que a criatura virtual capture joias presentes no ambiente e troque-as por pontos, ao mesmo tempo garantir que o agente sobreviva aos gastos energéticos oriundos da movimentação e exploração do ambiente. Com a execução dos experimentos, pode-se observar que o controlador motivacional foi o melhor controlador, considerando-se os aspectos de eficiência energética, com uma média igual a "535.36" e mediana igual "530.00". Além disso, com o valor da média dos desvios-padrão de "177.88" e variância de "33331.17", também pode-se afirmar que o controlador motivacional foi o controlador que manteve mais estável as reservas de energia da criatura. Do ponto de vista da eficácia energética, o controlador motivacional obteve uma acurácia de "80%", ou seja, dentre dez experimento executados, oito deles conseguiram manter a energia da criatura acima de zero até fim da simulação. Considerando-se a máxima pontuação obtida, novamente o controlador motivacional foi o melhor entre todos os controladores testados, com uma pontuação média igual a "77.3".

Palavras-chaves: Sistema Bio-inspirados; Sistema Motivacional; Comportamento Guiados a Objetivo; Arquiteturas Cognitivas; Cognitive System Toolkit.

Abstract

Motivations and emotions are intrinsically embedded in animal cognition and behavior, particularly in humans. They are responsible for supporting decision making, stimulating different behaviors such that their internal needs are satisfied. This work proposes the design and implementation of a motivational subsystem endowed with motivational and emotional capacities for the *Cognitive System Toolkit* (CST), a software toolkit for cognitive computing being developed by our research group, based on studies from the literature and different implementations of motivational and emotional systems in known cognitive architectures. In this context, the Motivational Subsystem was submitted to a set of simulations in the *World Server 3D Application*. In this simulator, it is possible to perform simulations where an artificial creature "lives" in a virtual environment where it can find food, obstacles and objects of value. Upon these objects, the creature can perform a set of actions. In the case of food, the agent can eat them, store them in their bag or bury them, such that they can be used later. For jewels, the agent can capture them and store them in their bag such that they are exchanged for points at the (*Delivery Spot*). They can also be buried to prevent the creature from colliding with undesirable jewels during the process of exploring the environment. In the context of obstacles, they are inanimate objects where they aim to make the creature's perception and exploitation difficult when it is in search of desirable objects. In order to evaluate the efficiency and effectiveness of the model, we developed a set of experiments with different intelligent controllers using the motivational subsystem model, a reactive system and also a set of cognitive architectures such as: JSOAR, CLARION and LIDA. The experiments are executed during a ten-minute period and are intended to cause the virtual creature to capture jewelry present in the environment, exchange them for points and at the same time make the agent survive, considering the energy expenditure demanded by movement while exploring the environment. With the execution of the experiments, we observed that the motivational controller was the best controller, regarding energy efficiency, with a mean equals to "535.36" and a median equals to "530.00". In addition, with the mean value of the standard deviation of "177.88" and variance of "33331.17", we noticed that the motivational controller was the controller which kept the creature's energy more stable. From the point of view of energy efficiency, the motivational controller obtained "80 %" accuracy of its experiments, that is, among ten performed experiments, eight of them resulted in keeping the creature's energy above zero until the end of the simulation . Considering the maximum score obtained, again the motivational controller was the best among all controllers with an average score of "77.3".

Keywords: Bio-inspired systems; Motivational System; Goal-directed Behaviors; Cognitive Architectures; Cognitive System Toolkit.

Lista de ilustrações

Figura 1 – Visão Geral da Arquitetura Cognitiva CLARION (SUN, 2005). 30
Figura 2 – A arquitetura cognitiva LIDA (MCCALL, 2014). 32
Figura 3 – Representação do WMEs na AC SOAR (LAIRD; CONGDON, 2014; LUCENTINI, 2017). 34
Figura 4 – Exemplo de regras utilizado pelo JSOAR durante a etapa de planejamento. 36
Figura 5 – A arquitetura cognitiva SOAR (LAIRD, 2008). 37
Figura 6 – Arquitetura de Subsunção Padrão x Subsunção Dinâmica 40
Figura 7 – Comportamentos Motivacionais sendo orientados por *drives* (FRóES, 2017). 41
Figura 8 – Drives sendo gerados pelos Codelets Motivacionais (FRóES, 2017). . . 42
Figura 9 – Os Codelets Emocionais aplicando a distorção cognitiva nos drives. . . 44
Figura 10 – Valor de Eval no estado normal ou de urgência. 45
Figura 11 – Os Codelets de Planejamento e Seleção de Planos interagindo com as sub-estruturas da Memória de Trabalho. 47
Figura 12 – Codelet de Avaliação realizando a avaliação primária do agente através da memória sensorial. 49
Figura 13 – Hypothetical Situations Appraisal. 49
Figura 14 – Codelets de Criação de Goals (ou Goal Codelets). 50
Figura 15 – Visão Geral do escopo do Subsistema Motivacional presente no CST. . 53
Figura 16 – Pacotes do projeto do CST. 54
Figura 17 – Diagrama UML das classes presentes no Subsistema Motivacional. . . 55
Figura 18 – A Aplicação World Server 3D (FRóES, 2017). 57
Figura 19 – Arquitetura Cognitiva para o Controle do Agente - Instância-exemplo do Subsistema Motivacional desenvolvido neste trabalho. 60
Figura 20 – Interação entre Codelets Sensoriais e o World Server 3D Proxy. 61
Figura 21 – Codelets Perceptuais gerando perceptos na memória perceptual a partir da memória sensorial. 63
Figura 22 – Interação entre Codelets Sensoriais, Perceptuais e os Motivacionais. . . 66
Figura 23 – Codelet de Apreciação realizando a avaliação primária do agente inteligente a partir das informações das memórias sensoriais. 68
Figura 24 – Regras Fuzzy Implentadas no Modelo de Mamdami presente no Codelet de Avaliação. 70
Figura 25 – Imagens das funções de pertinência utilizadas para determinar a Avaliação Primaria do agente cognitivo. 71

Figura 26 – Codelets de Humor gerando os Humores de Ansiedade e Exaltação, a partir da memória de avaliação do agente inteligente. 72
Figura 27 – A imagem da esquerda, demonstra a função matemática utilizada para computar a distorção cognitiva do drive de Fome e a imagem da direita apresenta a função matemática utilizada para computar a distorção cognitiva do drive de Ambição 75
Figura 28 – Codelets Emocionais computando as distorções cognitivas e interferindo no comportamento motivacional. 76
Figura 29 – A classe *Abstract Object* e suas relações com outros objetos do CST (GUDWIN *et al.*, 2017). 77
Figura 30 – Codelet de Geração de*Goals* utilizando informações sensoriais para gerar e definir um *Goal*. 78
Figura 31 – Estrutura do WMEs na disposição de um grafo dentro do JSOAR. . . . 80
Figura 32 – Exemplo de regras utilizado pelo JSOAR durante a etapa de planejamento. 84
Figura 33 – Os Codelets de Planejamento e de Seleção de Planos interagindo com as memórias de Goals, perceptual e de novo plano. 86
Figura 34 – Troca de mensagens entre os Codelets Perceptuais, Motivacionais, de Planejamento e Seleção de Planos. 91
Figura 35 – Os Codelets Motores 91
Figura 36 – Gráficos de Desempenho Energético e de Pontos Obtidos dos experimentos executados com o controlador desenvolvido com o controlador reativo. 93
Figura 37 – Gráficos de Desempenho Energético e de Pontos Obtidos dos experimentos executados com o controlador desenvolvido com a arquitetura cognitiva JSOAR. 93
Figura 38 – Gráficos de Desempenho Energético e de Pontos Obtidos dos experimentos executados com o controlador desenvolvido com a arquitetura cognitiva CLARION. 94
Figura 39 – Gráficos de Desempenho Energético e de Pontos Obtidos dos experimentos executados com o controlador desenvolvido com a arquitetura cognitiva LIDA. 94
Figura 40 – Gráficos de desempenho Energético e de Pontos Obtidos dos experimentos executados com o controlador desenvolvido com o Subsistema Motivacional do CST. 94
Figura 41 – Níveis de Ativação das seguintes variáveis do agente inteligente durante a execução do sexto experimento do controlador motivacional: Apreciação Primária (SMAN), Humores, Distorções Cognitivas das Emoções e Drives gerais (SMS). 100

Figura 42 – HLMS vs SMS. 102

Lista de tabelas

Tabela 1 – Parâmetros dos Codelets Perceptuais. 63
Tabela 2 – Parâmetros dos Drives gerados pelos Codelets Motivacionais. 66
Tabela 3 – Parâmetros do Sistema Fuzzy utilizado para determinar a Avaliação Primária do Agente Cognitivo. 71
Tabela 4 – Parâmetros dos Codelets de Comportamento Motivacional 90
Tabela 5 – Resultados Obtidos dos experimentos executados com o controlador Reativo. 95
Tabela 6 – Resultados Obtidos dos experimentos executados com o controlador com a arquitetura JSOAR. 96
Tabela 7 – Resultados Obtidos dos experimentos executados com o controlador com a arquitetura CLARION. 96
Tabela 8 – Resultados Obtidos dos experimentos executados com o controlador com a arquitetura LIDA. 97
Tabela 9 – Resultados Obtidos dos experimentos executados com o controlador com o Subsistema Motivacional. 97
Tabela 10 – Rankings de eficiência (a) e eficácia (b) energética, juntamente com o ranking de pontuação obtida (c) dos controladores Motivacional, LIDA, CLARION, JSOAR e Reativo. 99
Tabela 11 – Número de Planos gerados e executados no experimentos do controlador motivacional. 101

Lista de acrônimos e abreviações

AC	Arquitetura Cognitiva
ACT-R	Adaptive Control of Thought—Rational
ACS	Action-Centered Subsystem
ATTR	Atributo
CERA-CRANIUM	Conscious and Emotional Reasoning Architecture - Cognitive Robotics Architecture Neurologically Inspired Underlying Manager
CLARION	Connectionist Learning with Adaptive Rule Induction On-line
CST	Cognitive System Toolkit
HLMS	High-Level Motivational Subsystem
IA	Inteligência Artificial
ID	Identificador
JSOAR	Java - State, Operator And Result
LEABRA	Local, Error-driven and Associative, Biologically Realistic Algorithm
LIDA	Learning Intelligent Distribution Agent
MCS	Meta-Cognitive Subsystem
MS	Motivational System
NACS	Non-Action-Centered Subsystem
SM	Subsistema Motivacional
SMAN	Subsistema Motivacional de Alto Nível
SMEM	Mémoria Semântica
SMS	Subsistema Motivacional de Subsunção
SOAR	State, Operator And Result
SVS	Spatial Visual System
PAM	Perceptual Associative Memory
WS3D	The World Server 3D Application

Sumário

1 Introdução 17
1.1 Motivações 21
1.2 Objetivos 22
1.3 Organização do Texto 22
2 Fundamentação Teórica 24
2.1 Motivações e Emoções Segundo a Psicologia Cognitiva 24
2.2 Arquiteturas Cognitivas Utilizando Modelos Motivacionais e Emocionais . 26
2.3 O Subsistema Motivacional da Arquitetura CLARION 28
2.4 O Subsistema Motivacional da Arquitetura LIDA 31
2.5 Representações Motivacionais na Arquitetura SOAR 33
3 Modelo de Subsistema Motivacional 38
3.1 O Subsistema Motivacional de Subsunção 39
3.2 O Subsistema Motivacional de Alto Nível 46
3.2.1 Codelets de Planejamento e de Seleção de Planos 46
3.2.2 Codelets de Apreciação 48
3.2.3 Codelet de Criação de *Goals* 49
3.2.4 Integrando os Diferentes Codelets 50
3.3 Visão Geral do Subsistema Motivacional 52
3.4 O Desenvolvimento e Implementação do Subsistema Motivacional 54
4 Controlador de Agentes Inteligentes Baseado no Modelo Motivacional. . . 56
4.1 A Criatura Artificial e seu Ambiente Virtual 56
4.2 Experimentos 57
4.3 Detalhes do Controlador Motivacional 59
4.3.1 Codelets Sensoriais 59
4.3.2 Codelets Perceptuais 61
4.3.3 Codelets Motivacionais 63
4.3.4 Codelets de Apreciação 66
4.3.5 Codelets de Humor 70
4.3.6 Codelets Emocionais 74
4.3.7 Codelets de Geração de Goals 75
4.3.8 Codelets de Planejamento e Seleção de Planos 79
4.3.9 Codelets de Comportamento Motivacional 86
4.3.10 Codelets Motores 89
5 Discussão e Análise dos Resultados. 93
5.1 Análise das Estruturas Internas do Controlador Motivacional. 99

6 Conclusão 103
6.1 Trabalhos Futuros 104

Referências 106
7 Anexo 112

1 Introdução

Desde as primeiras conferências sobre a simulação do comportamento adaptativo, contribuições sucessivas desta comunidade foram propostas com o intuito de modelar como motivações e emoções podem ser úteis para o desenvolvimento de sistemas inteligentes (MEYER; WILSON, 1990). No final dos anos 90, surge o termo “Arquitetura Cognitiva”(AC), designando estruturas e processos essenciais de um modelo computacional geral inspirado na cognição e no comportamento animal. Sloman (2002), por exemplo, propõe uma visão "arquitetural"para a mente como um sistema de controle, categorizando conceitos e processos mentais. Sun (2004) discute o que seriam requisitos necessários para tipos gerais de arquiteturas cognitivas. Além disso, Sun (2007a) explora problemas e desafios encontrados no desenvolvimento de arquiteturas cognitivas. Ao mesmo tempo, Langley *et al.* (2009) analisa diversas arquiteturas cognitivas, avaliando os progressos obtidos até então.

Esses estudos indicam que a principal vantagem de caracterizar uma arquitetura como sendo cognitiva, é construir um *Framework* concreto (e testável) para a modelagem de processos cognitivos. Assim, uma AC pode ser vista como um sistema de controle de propósito geral, inspirado em diversas teorias científicas desenvolvidas originalmente para explicar a cognição em animais e no ser humano. De um modo geral, uma AC pode ser decomposta em vários módulos, cada um deles representando distintas capacidades cognitivas, tai como: Percepção, Atenção, Memória, Raciocínio, Aprendizado, Motivação, Emoção e outras estruturas.

O estudo das ACs tornou-se bastante popular nos últimos anos. Ao longo dos últimos 20 anos, diversas ACs foram propostas. As primeiras ACs têm suas raízes em sistemas baseados em conhecimento (sistemas especialistas/sistema de produção), muito populares na década de 70, antes mesmo desses sistemas entrarem em declínio. Tais sistemas passaram por aprimoramentos importantes, antes de ganhar um status de AC. Dentre as arquiteturas mais populares estão: o ACT-R (*Adaptive Control of Thought - Rational* (ANDERSON, 1983b; ANDERSON, 1983a; ANDERSON; BOWER, 1973) e o SOAR (*State, Operator And Result*) (LAIRD *et al.*, 1987; LAIRD *et al.*, 1984). Em 2010, Samsonovich (2010) publicou um amplo estudo que resultou em uma tabela comparativa, apresentando uma revisão abrangente das ACs implementadas e disponíveis na literatura. Entre tais ACs, um certo número delas são de propósito geral e em alguns casos, apresenta seu código fonte disponível na Internet para ser baixado, e consequentemente, prontas para desenvolver simulações. Existem diferentes tipos de ACs, com diferentes aspectos da modelagem cognitiva sendo utilizados em sua implementação. Neste trabalho, concentramo-nos apenas em ACs com algum tipo de estrutura e/ou capacidade motiva-

cional ou emocional.

O estudo das capacidades motivacionais ou emocionais se insere no âmbito mais amplo dos sistemas de comportamento (ou sistemas geradores de comportamento). Estritamente falando, se quiséssemos classificar os diversos comportamentos possíveis (sejam eles realizados por animais ou máquinas), provavelmente poderíamos enquadrá-los dentro das seguintes classes:

- Comportamentos Aleatório *(Random Behaviors)*.
- Comportamentos Reativos *(Reactive Behaviors)*.
- Comportamentos Propositados *(Goal-directed Behaviors)*.

Os comportamentos aleatórios são aqueles que não dependem de nada. São completamente aleatórios, sem influência de qualquer entrada. Os comportamentos reativos são comportamentos que dependem de algum tipo de entrada, ao qual o sistema reage. Geralmente, essa entrada é algum tipo de entrada sensorial, que é transformada internamente em uma saída pelo sistema. A saída de um sistema reativo é basicamente uma função determinística de suas entradas e também possivelmente de algum tipo de variável interna do sistema. Ambos os comportamentos, sendo eles aleatórios e/ou reativos são típicos em máquinas e outros objetos não animados. A compreensão da terceira categoria é um pouco mais complexa. Os comportamentos propositados são típicos dos sistemas vivos, particularmente dos animais. A ideia deste tipo de comportamento não é apenas um comportamento que seja aleatório ou reativo a algo. A ideia de um comportamento propositado, ou seja, orientado a objetivos, está atrelada a existência de uma finalidade para este comportamento, um objetivo que deve ser alcançado e realizado. Existe uma profunda discussão na filosofia sobre esta questão. Esta discussão começa em Aristóteles e em sua noção de causalidade final, passando por discussões subsequentes com relação a teleologia, teleonomia, e finalmente chegando na cibernética: a ciência do controle.

A pesquisa sobre sistemas motivacionais pode ser inserida neste estudo mais amplo sobre o comportamento propositado. No cotidiano das pessoas, existem exemplos muito simples de comportamentos propositados que as vezes passam desapercebidos, como por exemplo, qualquer sistema de controle em malha fechada, como termostatos (presentes em geladeiras, equipamentos de ar condicionado, além de outros equipamentos). Quando um termostato determina os sinais de controle para um sistema de refrigeração, ele não está simplesmente reagindo a uma entrada, mas está mudando o ambiente para que um objetivo (neste caso, uma temperatura de referência) seja atingido. Sendo assim, devido à realimentação (*feedback*) inerente a sistemas de malha fechada, o ambiente converge lentamente para a temperatura desejada. Entretanto, é importante diferenciar os comportamentos propositados dos comportamentos reativos. Um comportamento reativo não

tem um estado futuro previsto para ser alcançado. Ele simplesmente reage às entradas, sem uma determinação do que deve acontecer no futuro. Por outro lado, os sistemas propositados têm um futuro previsto e desejado, ou seja, as saídas do sistema estão, de certo modo, comprometidas com esse futuro esperado.

Mas os sistemas propositados podem ser bem mais complexos do que um simples termostato. O objetivo a ser alcançado pode exigir um planejamento anterior, com um conjunto de etapas intermediárias que devem ser realizadas para se alcançar o estado desejado. O sistema precisa, de certa forma, antecipar o futuro e selecionar um comportamento que maximize e otimize as chances de alcançar esse estado futuro desejado. Além disso, diversos acontecimentos podem ocorrer no ambiente e o sistema precisa encontrar o melhor comportamento possível para contornar esses eventos imprevisíveis, de modo que o estado do ambiente convirja para o desejado. Existem muitos modelos na literatura que tentam explicar o comportamento propositado. O trabalho de Hull (1943) sobre os princípios dos comportamentos é um desses modelos. De acordo com Hull, o comportamento propositado pode ser explicado em termos de *necessidades* intrínsecas aos seres vivos, que determinam o comportamento desses seres vivos em seu ambiente. Essas necessidades são a fonte de motivação para o comportamento propositado, direcionando o comportamento dos seres vivos. Os sistemas que podem ser modelados em termos de um conjunto de necessidades direcionando seu comportamento subjacente são chamados de sistemas motivacionais (ou motivados). Neste sentido, o conceito de emoção constitui uma sofisticação a esta ideia, estando portanto dentro do escopo dos sistemas motivacionais. Em sistemas motivacionais, usualmente diversas necessidades diferentes estão, ao mesmo tempo, direcionando o comportamento do sistema. É necessário existir algum tipo de prioridade entre elas, para que necessidades afetando os mesmos atuadores do sistema possam co-existir. Usualmente, em sistemas motivacionais, essa prioridade entre necessidades é fixa, o que significa que as necessidades mais prioritárias sempre terão mais força na hora de decidir o comportamento do sistema. É aí que a ideia de emoções pode sofisticar esse mecanismo. Em situações onde deseja-se que essa prioridade possa ser modificada, por alguma razão, um mecanismo emocional pode ser usado para modular essas necessidades, re-organizar a prioridade entre elas. Uma situação onde isso seria desejável seria em circunstâncias excepcionais, que demandam uma inversão de prioridades de forma a contemplar a excepcionalidade. Essa excepcionalidade pode ser devida à detecção de um perigo iminente, que coloca em risco a sobrevivência de uma criatura (medo), ou a percepção de uma situação rara, que não acontece sempre, e que portanto demanda uma ação imediata de forma a aproveitar esse acontecimento a benefício da criatura (oportunismo). O tratamento dessa excepcionalidade envolve uma modificação temporária na prioridade entre necessidades, que cessando-se essa condição, retorna a sua situação original. Motivações e emoções são partes fundamentais da vida humana que favorecem a sobrevivência e a convivência social. Essas ideias, são originalmente desenvolvidas para explicar os comportamentos realizados

pelos seres vivos (especialmente seres humanos), mas que podem ser utilizadas também em sistemas artificiais para gerar comportamentos propositados (CAñAMERO, 1997; SUN, 2009).

Toates (1986) e Breazeal *et al.* (1998) definem sistemas de comportamento motivados por inspiração biológica usando a ideia de necessidades internas como fonte de motivação. Essas necessidades internas são mensuradas por "Drives" [1] (um *drive* é uma variável numérica que mede o nível de insatisfação de uma determinada necessidade) e desempenham funções importantes em animais, tais como:

i). Motivá-los a selecionar a melhor decisão para satisfazer suas necessidades fisiológicas e sociais.

ii). Influenciar o estado emotivo passando energia de ativação para os processos emotivos.

iii). Fornecer um contexto de aprendizagem aos animais para aprimorar e criar novas habilidades.

Toates (1986) apresenta uma analogia interessante que pode ser associada a "Drives": um "Drive" energiza um animal da mesma forma que a gasolina faz isso em um carro.

Diversas ACs buscam inspiração em modelos cognitivos de motivações e emoções em seus mecanismos internos. Por exemplo, CLARION, LIDA, CERA-CRANIUM e MicroPsi, todas estas ACs disponibilizam algum tipo de mecanismo motivacional ou emocional dentre seus recursos disponíveis. Esses mecanismos suportam a seleção e execução de ações em função de motivações/emoções. Embora seja possível conceber ACs puramente reativas, uma das maneiras para que comportamentos propositados possa ser gerado envolve o uso de mecanismos motivacionais ou emocionais (embora não seja a única maneira, entretanto). Além disso, como as ACs são um tipo de sistema de controle de propósito geral, inspirado em modelos cognitivos de animais e homens, essas capacidades certamente são desejáveis em uma AC moderna.

Neste trabalho, desenvolvemos um Sistema Motivacional para o CST, um *toolkit* baseado em Java para a construção de ACs, sendo desenvolvido pelo nosso grupo de pesquisa na FEEC-UNICAMP. O CST incorpora diversos recursos comuns presentes em outras ACs, mas até o momento não possuía capacidades motivacionais. Como explicamos na seção 2 a seguir, antes de desenvolver uma implementação computacional, realizamos um estudo sobre como as motivações e emoções estão sendo usadas em diferentes ACs e,

[1] Com relação ao termo "Drive", preferimos manter a sua grafia original, sem tradução, tendo em vista que o mesmo é um termo técnico amplamente utilizado na literatura. Possíveis traduções do termo para a língua portuguesa não expressam adequadamente todo o significado do termo original em inglês. O mesmo acontece com outras palavras presentes neste trabalho, tais como: "Goals" ou "Codelets".

em seguida, selecionamos um conjunto de recursos que consideramos importante implementar no Subsistema Motivacional do CST. Finalmente, para avaliar o seu desempenho, planejamos e executamos experimentos usando o simulador *World Server 3D Application*, um ambiente virtual usado também em outros trabalhos do nosso grupo de pesquisa, para testar e validar os diferentes aspectos do módulo motivacional aqui desenvolvido. Paralelamente a isso, realizamos experiências similares usando outras ACs, como SOAR (LAIRD *et al.*, 1987; LAIRD *et al.*, 1984), CLARION (SUN, 2003) e LIDA (FRANKLIN *et al.*, 2016), para ter uma comparação com nossa implementação. Além disso, nosso subsistema motivacional foi testado e comparado com um controlador completamente reativo utilizando entidades básicas do CST.

1.1 Motivações

Dentre as motivações que alavancaram o desenvolvimento deste trabalho está, em primeiro lugar, a demanda por equipar o CST com algum tipo de subsistema motivacional, tendo em vista que o mesmo não possuía até então nenhum tipo de suporte a capacidades motivacionais/emocionais em seu escopo. Algumas das ACs mais populares e difundidas no meio acadêmico (como LIDA, CLARION, CERA-CRANIUM etc.) implementam algum módulo disponibilizando facilidades motivacionais/emocionais. Portanto, sendo o CST uma ferramenta para a construção de ACs, seria de grande importância que o mesmo disponibilizasse recursos de apoio a motivações e emoções para os desenvolvedores, de forma análoga às demais ACs que possuem módulos motivacionais.

Uma segunda motivação para o desenvolvimento deste trabalho foi a possibilidade do mesmo trazer também alguma contribuição ao estudo de motivações e emoções, dentro do escopo das Ciências Cognitivas. Modelos computacionais de processos motivacionais/emocionais podem trazer uma melhor compreensão da natureza desse fenômeno, e nos ajudar a compreender melhor como o ser humano e animais escolhem suas ações e o que são e para que servem as emoções. Não é por acaso que a "Inteligência Artificial" (IA), é vista como uma das ciências cognitivas, sendo uma importante proponente de teorias visando explicar como funcionam os processos cognitivos, comportamentos e habilidades dos seres vivos, exatamente por tentar reproduzi-los em sistemas de controle de agentes artificiais (incluindo-se máquinas, robôs, carros autônomos, jogos de computador/consoles, etc.). Desta forma, imaginamos que novos modelos computacionais para motivações e emoções poderiam trazer alguma contribuição para as ciências cognitivas.

Nosso intuito, com o desenvolvimento do modelo de subsistema motivacional proposto por este trabalho, era poder implementar e executar experimentos de controladores de agentes inteligentes baseados no modelo a ser desenvolvido, a fim de comprovar e verificar sua eficiência e eficácia, quando comparado com sistemas de controle tradicionais

e também sistemas de controle baseado em outras ACs. Da mesma forma, vislumbrávamos que a implementação de um modelo de subsistema motivacional proporcionaria uma compreensão mais apurada de como as motivações e emoções atuam nas mentes dos seres vivos, principalmente em situações excepcionalmente adversas (risco de sobrevivência) ou favoráveis (oportunidades de recompensas), que é onde as emoções parecem ter maior efeito sob os animais.

1.2 Objetivos

O principal objetivo deste trabalho foi estudar diferentes implementações de sistemas motivacionais e emocionais em ACs e a partir desse estudo desenvolver um subsistema motivacional equipado com capacidades motivacionais e emocionais para o CST, permitindo a construção de agentes cognitivos artificiais com capacidades motivacionais/emocionais na determinação de comportamento propositado. Para alcançar esse objetivo geral, identificou-se os seguintes objetivos específicos:

- Estudar teorias de motivação e emoção na Psicologia Cognitiva;
- Selecionar e revisar ACs com algum tipo de estrutura ou comportamento motivacional e/ou emocional;
- Estudar os recursos e conceitos fundamentais sobre o CST;
- Propor e desenvolver um modelo computacional para comportamentos motivacionais e emocionais a serem disponibilizado no CST;
- Executar experimentos que validem a eficiência e a eficácia de nossa proposta de subsistema motivacional;
- Analisar e comparar os resultados dos experimentos com outras ACs;

1.3 Organização do Texto

Os próximos capítulos estão organizados da seguinte forma. O capítulo 2 descreve os fundamentos teóricos do projeto. A seção 2.1 faz uma revisão das principais teorias psicológicas sobre motivações e emoções. A seção 2.2, analisa um grupo selecionado de ACs que fazem uso de representações motivacionais e/ou emocionais em seu repertório de capacidades. O capítulo 3 apresenta os principais detalhes do modelo motivacional proposto neste trabalho. Já no capitulo 4, descrevemos o controlador motivacional implementado utilizando nosso modelo de subsistema motivacional, junto com a metodologia utilizada para sua validação, especificando o ambiente de simulação, controladores desenvolvidos

e os experimentos que foram executados. Finalmente, no capítulo 5 apresentamos os resultados obtidos a partir da execução dos experimentos e no capítulo 6 desenvolvemos uma conclusão sobre os experimentos realizados e a validação do subsistema motivacional como um todo.

2 Fundamentação Teórica

Neste capítulo, realizamos uma breve revisão da literatura envolvendo os temas de motivações e emoções. Na seção 2.1, apresentamos a revisão da literatura sobre comportamentos motivacionais e emocionais, a partir de uma perspectiva da psicologia cognitiva. Já na seção 2.2, fazemos uma analise de como essas ideias foram utilizadas em algumas ACs selecionadas.

2.1 Motivações e Emoções Segundo a Psicologia Cognitiva

O conceito de comportamento motivacional em ACs tem sua inspiração em estudos sobre motivação humana realizados por Hull (1943) e Maslow (1943) na Psicologia Cognitiva. De acordo com a teoria do comportamento de Hull (HULL, 1943), quando uma ação motora constitui-se de um pré-requisito para otimizar a probabilidade de sobrevivência de um indivíduo ou espécie, dizemos que existe aí uma *necessidade*. Esta necessidade motiva ou impulsiona uma ação motora associada. Sendo assim, Hull define um *drive* como sendo uma variável usada para caracterizar a insatisfação de uma necessidade, ou seja, uma medida do quão distante está essa necessidade de ser satisfeita. Os *drives* são usados para direcionar um futuro desejável, que uma criatura deve alcançar para sobreviver. Em um ser vivo, podem existir diversos *drives*, cada um deles relacionado a uma diferente necessidade, como por exemplo a necessidade de alimentos, água, ar, a necessidade de evitar lesões, manter uma temperatura ideal, a necessidade de descansar, dormir, acasalar etc. Em um agente artificial, os *drives* estão associados aos comportamentos desejáveis a um agente, para a satisfação de seus objetivos. Eles direcionam um futuro desejável para esse agente. Assim, um *drive* pode ser visto como uma especificação de um propósito. Além disso, a cada *drive* é associado um comportamento, que deve ser responsável em diminuir o *drive*. Dessa forma, dizemos que esse comportamento é um comportamento motivacional ou motivado.

Seguindo as ideias de Hull, Sun (2003) propôs uma adaptação dessa teoria de motivação, de modo a permitir sua implementação em uma AC. De acordo com Sun (2009), humanos, animais ou agentes cognitivos precisam seguir alguns critérios importantes em sua atividade diária, de modo que possam sobreviver em determinado ambiente. Esses critérios são: Sustentabilidade, Propósito, Foco e Adaptação. Todo agente deve atender às suas necessidades fisiológicas (Sustentabilidade), como a fome, a sede, evitar perigos físicos e outros. Consequentemente, suas ações devem ser baseadas em propósitos, ao invés de escolhas aleatórias ou puramente reativas (Propósito) (HULL, 1943; ANDERSON, 1993; SUN, 2009). Além disso, o agente deve concentrar suas atividades de forma que

suas necessidades e propósitos sejam consistentes, persistentes e contínuos (Foco) (TOATES, 1986; SUN, 2009). No entanto, podem haver situações temporárias ou permanentes nas quais o agente pode entrar em um estado de urgência (SLOMAN, 1987; SUN, 2009), quando um comportamento especial pode ser apropriado. Finalmente, o agente deve ser capaz de adaptar seus comportamentos de acordo com seus objetivos para obter melhores resultados (SUN, 2009). Esses critérios, facilitam a compreensão de que a dinâmica motivacional em animais ou seres humanos é uma parte crucial da composição de um comportamento final (SUN, 2009).

Os *drives* podem ser de origem fisiológicas ou sociais. As necessidades fisiológicas dão origem aos chamados *Drives Primários de Baixo Nível* (ou *Low-level Primary Drives*). Na literatura, *Drives Primários de Baixo Nível* são relatados por muitos autores (MURRAY, 1938; HULL, 1943; MCCLELLAND, 1951; TYRRELL, 1993; SUN, 2009). Por exemplo, necessidades como: fome, sede, sono e evasão de perigo físico, podem ser classificados como *drives* primários de baixo nível. Já as necessidades sociais dão origem aos *Drives Primários de Alto Nível* (ou *High-level Primary Drives*). Exemplos de *drives* primários de alto nível poderiam ser amizade, desejo de pertencimento a um grupo, desejo de reciprocidade, interesse em explorar o ambiente, curiosidade, autonomia e honra. Diversos trabalhos na literatura referem-se a *Drives Primários de Alto Nível* (MURRAY, 1938; MASLOW, 1970; REISS, 2004; MCDOUGALL, 2015). Tanto os *drives* de baixo nível como os de alto nível são entretanto exemplos de *drives* primários. De acordo com Buck (1988), *drives* primários são "distúrbios biológicos internos"que causam necessidades ao organismo, como por exemplo, necessidades de alimentos, água, ar, sexo, dor, evasão e assim por diante. A existência dessas necessidades determina ao organismo seus propósitos, com uma força relativa e estímulo persistente, energizando o comportamento animal. Por exemplo, a necessidade de alimentos pode causar "distúrbios internos"até que o animal encontre comida e satisfaça essa necessidade. Essa satisfação ou redução da intensidade do *drive* correspondente, é muito importante para a aprendizagem animal, porque serve como parâmetro de reforço associado a um estímulo animal, e consequentemente, dando origem a uma respectiva resposta. O aprendizado de *drives* primários pode resultar em *drives* secundários através do processo de "condicionamento clássico"ou "condicionamento operante".

Tanto o modelo original de Hull como o adaptado de Sun sugerem que motivações sejam usadas na forma de um sinal de referência em um sistema de controle por realimentação, orientando e otimizando comportamentos humanos para alcançar os melhores resultados considerando suas necessidades internas. Portanto, necessidades motivam humanos e animais a decidir a melhor escolha de ação, de modo que essas necessidades internas possam ser satisfeitas, e consequentemente, um estado desejado seja alcançado.

A noção de *drive* é muito importante para entender outra capacidade cognitiva

crítica: as emoções (CAñAMERO, 1997). Existe uma relação intrínseca entre motivações e emoções. O conceito de emoções veio da psicologia cognitiva e da filosofia, como uma maneira alternativa de abordar o problema da geração de comportamentos (BATES *et al.*, 1994; BUDAKOVA; DAKOVSKI, 2006; CAñAMERO, 1997; CAñAMERO, 1998; MEYER, 2008; PICARD; PICARD, 1997; REILLY, 1996; SEPTSEAULT; NÉDÉLEC, 2005). Não há consenso sobre o que realmente são emoções. Existem diferentes abordagens com visões distintas sobre o que são emoções e como podem ser modeladas. Por exemplo, Ortony *et al.* (1990) compreende as emoções como "reações valoradas a eventos, agentes ou objetos". Sloman (SLOMAN, 1998; SLOMAN *et al.*, 2001), por sua vez, entende emoções como "alarmes" internos que dão uma ênfase momentânea a certos grupos de sinais. Damasio (DAMASIO, 1994; DAMASIO, 1999) distingue entre "emoções" que afetam o corpo e "sentimentos", onde ocorre uma introspecção cognitiva dessas emoções. Outros autores têm opiniões completamente diferentes sobre o que são as emoções. Por exemplo, para Cañamero (1997), emoções funcionam como "amplificadores" de motivações, funcionando como processos homeostáticos relacionados a variáveis fisiológicas.

2.2 Arquiteturas Cognitivas Utilizando Modelos Motivacionais e Emocionais

Diversas arquiteturas cognitivas apresentam módulos ou subsistemas especiais usando algum modelo cognitivo para motivações e/ou emoções. Nesta subseção, são analisadas algumas delas.

A arquitetura CLARION *("Connectionist Learning with Adaptive Rule Induction ON-line")* é uma AC bastante conhecida que vem sendo desenvolvida pelo grupo de pesquisa de Ron Sun desde 1996. A arquitetura possui diversos módulos, com diferentes propósitos, sendo que existe um módulo específico que funciona como *Subsistema Motivacional*, responsável por melhorar a tomada de decisões e fornecer uma estrutura de comportamento baseada em objetivos e motivações. A arquitetura CLARION é descrita de maneira detalhada em (SUN *et al.*, 1996; SUN, 2003; SUN, 2004; SUN, 2007a; SUN, 2009).

A arquitetura CERA-CRANIUM *(Conscious and Emotional Reasoning Architecture - Cognitive Robotics Architecture Neurologically Inspired Underlying Manager)* é outra arquitetura cognitiva, proposta por Arrabales, inspirada em modelos cognitivos de consciência e teorias sobre emoções. Mais detalhes sobre a arquitetura CERA-CRANIUM podem ser encontrados em (ARRABALES *et al.*, 2009c; ARRABALES *et al.*, 2009a; ARRABALES *et al.*, 2009c; ARRABALES *et al.*, 2009b; ARRABALES *et al.*, 2011).

A arquitetura MicroPsi é uma AC proposta por Bach na Universidade Humboldt de Berlim, que tem como base a teoria "Psi"de Dietrich Dörner. Essa arquitetura combina

representações neuro-simbólicas, tomada de decisão autônoma e motivação baseada em aprendizagem. Além disso, o sistema motivacional da MicroPsi leva em conta necessidades cognitivas, sociais e fisiológicas. Os detalhes desta AC podem ser encontrados em (BACH, 2003; BACH; VUINE, 2003; BACH, 2005; BACH, 2012; BACH, 2015).

A arquitetura LEABRA *(Local, Error-driven and Associative, Biologically Realistic Algorithm)* é outra arquitetura cognitiva, introduzida por O'Reilly. Ela foi desenvolvida utilizando algoritmos de redes neurais implementando uma teoria computacional das funções cerebrais. O LEABRA pode ser descrita em dois níveis: a arquitetura em larga escala e o mecanismo neural básico. O núcleo da arquitetura em grande escala é composto por três tipos de sistemas cerebrais: o córtex posterior, o hipocampo e o córtex frontal. O córtex posterior e o hipocampo são responsáveis pelo processamento de informações perceptivas, semânticas e mecanismos de memória episódica. O córtex frontal é responsável por manter metas e outras informações de contexto. Também incorpora sistemas neuromoduladores que são conduzidos por áreas corticais e subcorticais envolvidas no processamento emocional e motivacional. Detalhes da arquitetura LEABRA podem ser encontrados em (O'REILLY, 1996; O'REILLY *et al.*, 2012).

A arquitetura LIDA é outra arquitetura cognitiva, proposta por Stan Franklin na Universidade de Memphis e originalmente introduzida em (FRANKLIN; Patterson Jr., 2006). A arquitetura LIDA é uma AC de propósito geral que tenta modelar a cognição em sistemas biológicos, implementando estruturas de controle para agentes de software e robôs. A arquitetura LIDA é composta por diferentes módulos que operam ao longo de um ciclo cognitivo. Ela apresenta um conjunto de módulos interligados, como a memória associativa perceptual, a memória espacial, a memória episódica transiente, a memória declarativa, os codelets de atenção, a memória sensório-motora e outras estruturas. A arquitetura baseia-se na "Teoria do Espaço de Trabalho Global"proposta por Baars (1997) e no conceito de *Codelets* introduzido por Hofstadter *et al.* (1994). Originalmente, a LIDA não apresenta módulos motivacionais ou emocionais explícitos em seu escopo. No entanto, algumas extensões desenvolvidas posteriormente incorporam modelos de motivações e emoções para a arquitetura (FRANKLIN *et al.*, 2016; MCCALL, 2014). Mais detalhes sobre a AC LIDA podem ser encontrados em (FRANKLIN *et al.*, 1998; FRANKLIN; Patterson Jr., 2006; FRANKLIN *et al.*, 2014; MCCALL, 2014; FRANKLIN *et al.*, 2016).

Finalmente, o SOAR é uma AC de propósito geral para desenvolver sistemas com comportamento inteligente. Foi proposto por John Laird, Allen Newell e Paul Rosenbloom na Universidade de Carnegie Mellon, mas atualmente é mantida pelo grupo de pesquisa de John Laird na Universidade de Michigan. A estrutura do SOAR é composta basicamente por uma memória de trabalho que representa o estado atual do agente e fornece interfaces para as memórias de longo prazo. Um Sistema Visuoespacial (SVS - *Spatial Visual System*) suporta representações relacionais necessárias para o raciocínio de alto nível e três tipos

de memórias de longo prazo: uma memória semântica; uma memória episódica e uma memória procedural. Além disso, a AC também possui um mecanismo de aprendizado por reforço, um sistema de avaliação que gera emoções e sentimentos, e um módulo de agrupamento que cria dinamicamente novos conceitos e símbolos. A versão original do SOAR foi desenvolvida em C++, mas existem conectores para diversas outras linguagens, tais como Java, TCL, JRuby, Jython, Rhino (JavaScript), Groovy, Scala, Clojure, etc., utilizando o SWIG (*Simplified Wrapper and Interface Generator*. Existe ainda uma versão do SOAR desenvolvida diretamente em Java por Dave Ray juntamente com o grupo SoarTech. Detalhes sobre a SOAR podem ser encontrados em (LAIRD *et al.*, 1984; LAIRD *et al.*, 1987; LAIRD, 2008; LAIRD, 2012; LAIRD *et al.*, 2012).

Existem diversas similaridades e algumas diferenças no modo como sistemas motivacionais são utilizados nas diferentes arquiteturas cognitivas avaliadas. Sem perda de generalidade, detalharemos a seguir os sistemas motivacionais das arquiteturas CLARION e LIDA, que foram as principais arquiteturas onde nos inspiramos para o sistema motivacional/emocional desenvolvido nesse trabalho.

2.3 O Subsistema Motivacional da Arquitetura CLARION

O primeiro subsistema motivacional que será detalhado neste trabalho, justamente porque nos inspiramos em diversas de suas estruturas na concepção de nosso sistema motivacional, é o da arquitetura CLARION. De acordo com Sun (2005), Sun (2009), a arquitetura CLARION é uma AC composta de 4 grandes subsistemas distintos, onde em cada um deles processa-se informações de dois níveis distintos: informações explícitas e implícitas. Uma informação explícita é geralmente simbólica e usualmente processada por um sistema baseado em regras. Uma informação implícita é geralmente sub-simbólica, sendo processada por meio de uma rede neural. Os quatro subsistemas principais do CLARION são os seguintes:

- *Action-Centered Subsystem* (ACS),
- *Non-Action-Centered Subsystem* (NACS),
- *Motivational Subsystem* (MS) e
- *Meta-Cognitive Subsystem* (MCS).

Cada subsistema é responsável por executar uma função específica na arquitetura. O ACS é responsável por gerar as ações do agente (reativas ou aleatórias). Já o NACS é responsável por gerenciar o conhecimento geral do agente, explícito ou implícito, na forma de um sub-sistema de memórias. O subsistema MS complementa o ACS, permitindo também que ações propositadas sejam geradas (o ACS original gera somente ações reativas

ou aleatórias). Por fim, o MCS é responsável pelo monitoramento e modificação das operações de todos os outros subsistemas, principalmente o subsistema ACS, permitindo um ajuste fino nas operações do próprio sistema.

A figura 1 apresenta o esquema geral da arquitetura. Cada subsistema apresenta dois níveis de representação. O nível superior (*top-level*), responsável pelo processamento e codificação de conhecimento explícito e o inferior (*bottom-level*), responsável pela codificação e processamento de conhecimento implícito. Estes dois níveis interagem e cooperam entre si, combinando as ações recomendadas em cada nível.

Sun (2007b) argumenta que uma característica distintiva da arquitetura CLARION é sua representação dual, considerando tanto conhecimento implícito como explícito. Na visão de Sun, representações implícitas são menos acessíveis e possuem um caráter "holístico". Em contraste, as representações explícitas são mais acessíveis e mais compreensíveis por parte de um ser humano (REBER, 1989; SUN, 2001). Na ciência cognitiva existe uma dicotomia semelhante entre representações simbólicas e subsimbólicas. Da mesma forma, na psicologia, a dualidade implícita *versus* explícita pode ser justificada por estudos empíricos sobre apreensão, memória e percepção (REBER, 1989; CLEEREMANS *et al.*, 1998; SUN, 2001).

De acordo com Sun (2009), o subsistema motivacional (MS) é responsável por complementar a tomada de decisões na arquitetura, fornecendo a infra-estrutura para a geração de comportamento propositado. Essa infra-estrutura básica dada pelo MS inclui *drives* no nível inferior (conhecimento implícito) e *goals* no nível superior (conhecimento explícito). Os *drives* estão diretamente relacionados às necessidades do agente cognitivo. Essas necessidades são os fatores-chave para o agente sobreviver em um ambiente. O CLARION fornece suporte nativo tanto a *drives* primários de baixo nível como de alto nível. Além disso, oferece também suporte a *drives* secundários, ou seja, *drives* derivados da combinação de outros *drives*. *drives* mutáveis, geralmente aprendidos através de um processo de condicionamento ou por meio de instruções de origem externa.

Como dito anteriormente, *drives* dão suporte à tomada de decisões no CLARION. Para ilustrar melhor a operação dos *drives*, imaginemos um agente cognitivo que depende de água, e em função dessa necessidade passa a ter sede. Quando a necessidade de água aumenta significativamente, a intensidade do *drive* aumenta. A consequência disto, é definir um novo *goal*, a fim de suprimir a intensidade do *drive* de sede. Alguns *drives* podem ser mais importantes do que outros, e, portanto, são organizados hierarquicamente. Os *drives* que estão em uma posição de maior hierarquia terão prioridade em sua satisfação. Um exemplo deste caso é o *drive* de auto-preservação, em comparação com um *drive* de necessidade de reprodução. Um agente não será motivado para se reproduzir se esse comportamento colocar o agente em uma situação perigosa. Neste caso, dizemos que o *drive* de auto-preservação é de maior prioridade do que o *drive* de reprodução. Um fator impor-

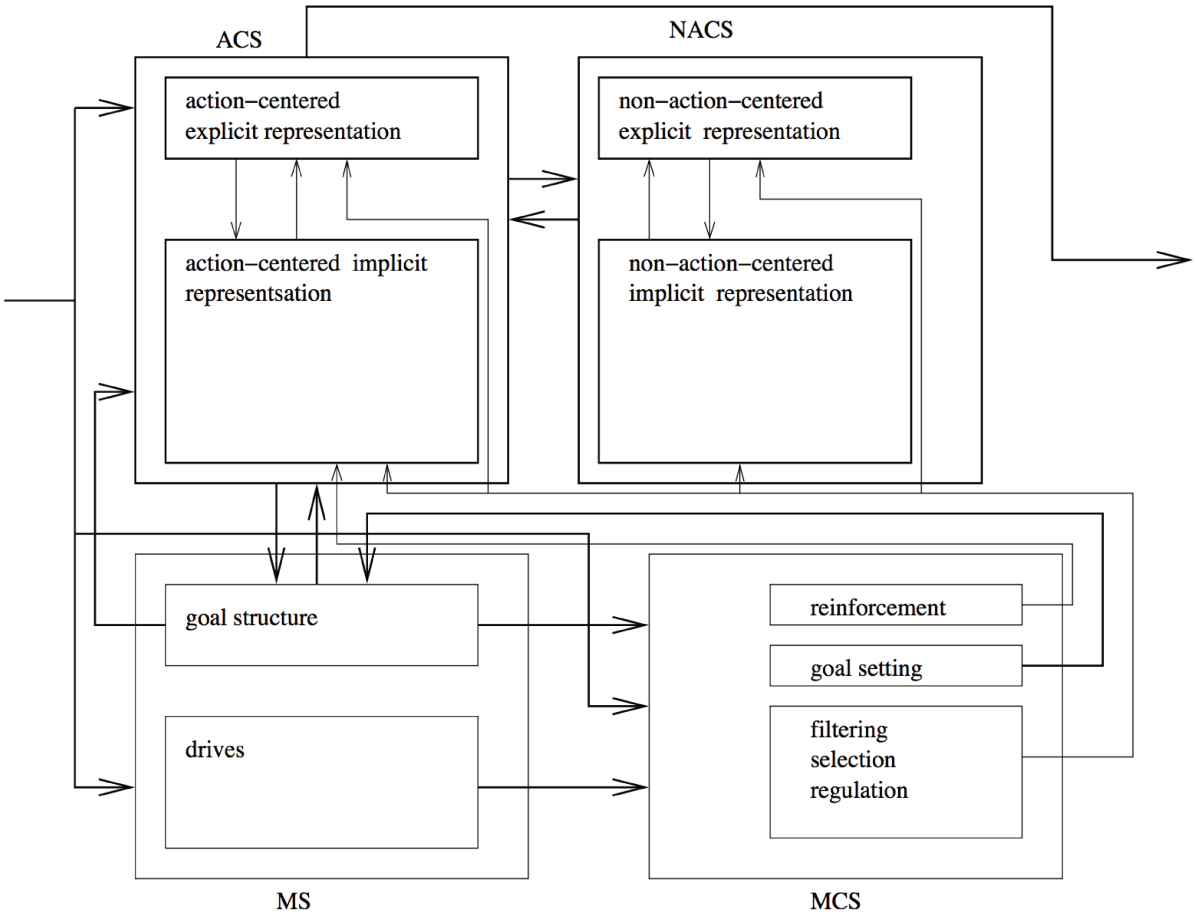

Figura 1 – Visão Geral da Arquitetura Cognitiva CLARION (SUN, 2005).

tante é a *acessibilidade* de um *drive*. Podem existir condições para que um *drive* possa ser satisfeito. Por exemplo, se um agente precisar de água, mas ainda não encontrou, o agente deve continuar sua busca pela água antes de tentar bebê-la. Desta forma, a necessidade de água determinará diretamente o conjunto de objetivos de um agente cognitivo. Mesmo que uma necessidade seja mais forte do que outras, o subsistema motivacional pode tentar satisfazer outra necessidade se um objeto requerido para a primeira necessidade não estiver disponível (SUN, 2003).

De acordo com Sun (2009), os *goals* são gerados diretamente pelos *drives*. Existem duas maneiras de definir um *goal* no CLARION. O primeiro é através do que o Sun chama de "balanceamento de interesses". Este método propõe uma competição entre *goals*. Cada *drive* vota em múltiplos *goals* com um valor numérico diferente (valores variando entre 0 e 9) indicando a prioridade para selecionar um determinado *goal*. O *goal* com o maior número de votos é selecionado para ser executado. O segundo método proposto por Sun é o famoso 'Winner-takes-all'. Este método seleciona um *goal* com base no *drive* de maior intensidade. Isso significa que o *goal* escolhido será aquele que satisfaça o *drive* vencedor.

2.4 O Subsistema Motivacional da Arquitetura LIDA

Em seu livro *"Artificial Minds"* (FRANKLIN, 1997), Stan Franklin discute diferentes teorias utilizadas no desenvolvimento da arquitetura LIDA (dentre elas, a arquitetura Copycat de Hoftstadter, a arquitetura de subsunção de Brooks, as Redes de Comportamento de Maes, as Redes de Esquemas de Drescher, etc.). Mesmo que cada módulo no LIDA venha de uma teoria diferente, todos compartilham um princípio comum. Todos os módulos no LIDA podem ser descritos como uma rede de *nós*, conectados por diferentes tipos de *links*, onde algum tipo de ativação se espalha pela rede. A inspiração em redes neurais é clara, mas as redes de espalhamento de ativação do LIDA não podem ser classificadas como redes neurais. Elas possuem um modelo sofisticado de propagação de ativação, que considera uma ativação básica para cada nó e link (utilizada no mecanismo de aprendizagem) e uma ativação atual, que devem se somar para gerar uma ativação total, que é utilizada no espalhamento. Essas ativações podem ter diferentes estratégias de excitação e decaimento ao longo do tempo, que podem ser diferentes em cada módulos do LIDA.

A arquitetura LIDA não apresenta originalmente nenhum módulo específico para gerar comportamento motivacional. No entanto, em sua tese de doutorado, McCall (2014) propôs um conjunto de modificações no *framework* LIDA, visando implementar um modelo de comportamento motivacional. Na realidade, McCall não cria um subsistema dedicado para o comportamento motivacional. Ao invés disso, ele propôs uma modificação no núcleo da arquitetura, e na sequência, criou um conjunto de experimentos para validar sua proposta. Como todos os módulos da arquitetura LIDA espalham ativações por seus nós e links, basicamente, McCall propôs para cada nó da rede, além dos diversos valores de ativação padrão, um novo atributo para representar uma valência afetiva ou uma saliência de incentivo (basicamente, uma desejabilidade) associada a este nó. Isso está em sinergia com a Teoria da Avaliação (LAZARUS, 1991). Na Figura 2 podemos ver a arquitetura LIDA, juntamente com uma marcação verde que destaca as áreas onde McCall propôs novos recursos.

McCall (2014) definiu uma série de conceitos em sua implementação de sistema motivacional para o LIDA. O primeiro é a noção de *receptor homeostático*. *Receptores homeostáticos* são tipos especiais de sensores que detectam o valor de atributos críticos no estado corporal de um agente. Exemplos de receptores em humanos incluem sensores para verificar baixos níveis de oxigênio ou glicose no sangue. Por serem sensores, são representados no LIDA apenas como mais um sensor no módulo de memória sensorial. Sensores são o ponto de partida do ciclo cognitivo do LIDA, a partir dos quais é possível ativar-se um nó associado na "Memória Associativa Perceptual"(do inglês Perceptual Associative Memory) - PAM, o módulo responsável pela percepção na AC. Para esse fim, McCall introduziu a noção de *nós de sentimento* (ou *feeling nodes*) na PAM. Esses

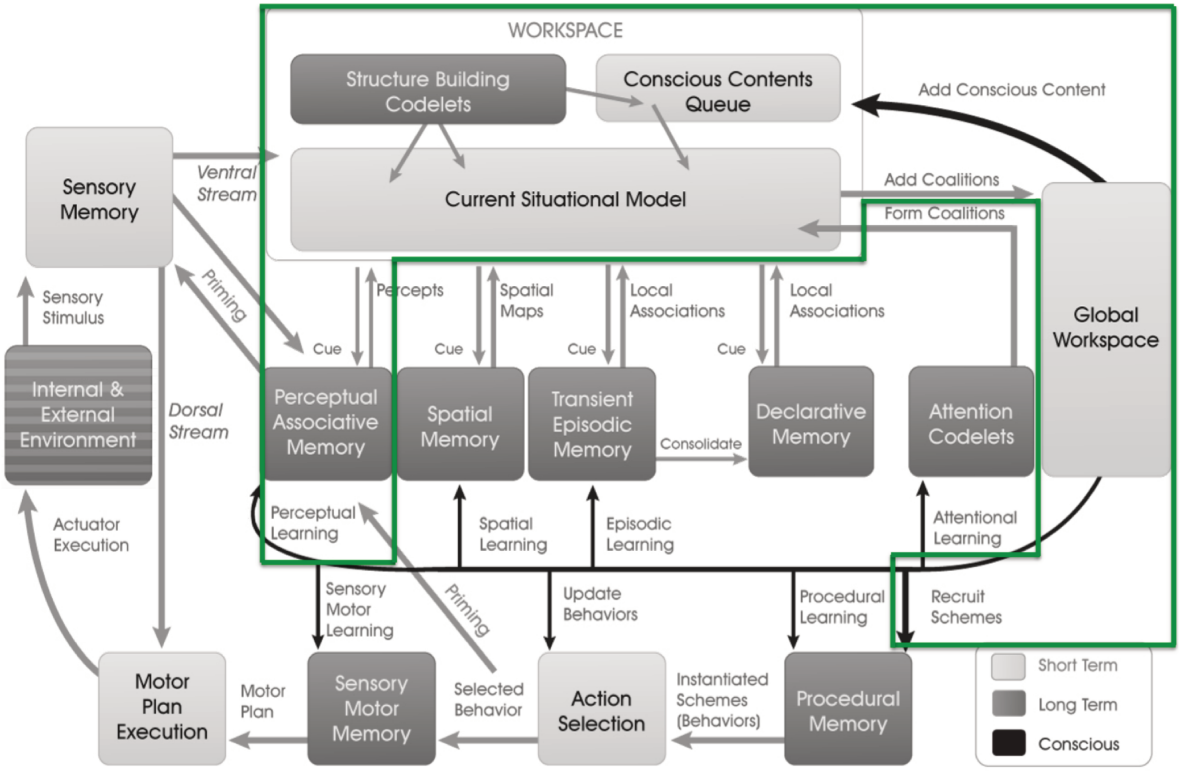

Figura 2 – A arquitetura cognitiva LIDA (MCCALL, 2014).

nós podem ser de dois tipos: *nós de sentimento de drives* (*drive feeling nodes*), em que representa o *drive* de um agente e *nós de sentimento interpretativo* (ou *interpretive feeling nodes*), representando uma interpretação perceptual a um estímulo. Nós de sentimento possuem um *sinal de valência*, que representa uma *valência afetiva*. O sinal de valência de um nó de sentimento, é associado à ativação atual do nó, onde ele tem a finalidade de representar a magnitude de "gostar" ou "não gostar" de algo. Além dos nós de sentimentos, McCall propôs a criação dos *nós de evento* (ou *event nodes*). Esses nós são responsáveis por descrever um evento específico que pode ou não ser executado pelo agente. Os nós de eventos podem ser disparados por nós de percepção e sentimento para realizar suas devidas ações. Se isso acontecer, McCall chama este tipo de situação de um "evento de sentimento". O nó de evento inclui um outro atributo denominado *saliência de incentivo* (*incentive salience*). Este atributo é um escalar que varia de negativo (temer) a positivo (querer), medindo a atratividade do evento para o agente cognitivo. Em outras palavras, esse atributo descreve se o agente cognitivo "quer"ou "não quer"um objeto ou evento. Berridge e Aldridge (2008) diferencia "querer" algo de "gostar" de algo, pois "gostar" é uma experiência presente e "querer" está relacionado a uma experiência hipotética futura de "gostar" de algo que está presente. O "gostar" de algo é produzido imediatamente pelo prazer obtido do contato com um estímulo, enquanto que "querer" algo está diretamente ligado à qualidade motivacional de um estímulo que o torna desejável e/ou atraente.

Emoções, no contexto do LIDA, podem ser interpretadas como um sentimento direcionado a *alguma coisa* ou a *alguém*. Por exemplo, a emoção de sentir-se "triste" porque se sofreu uma perda ou a emoção de sentir-se "com raiva" de alguém porque este alguém interferiu em algo que se queria. As emoções, como medo, raiva, alegria, tristeza, vergonha, constrangimento, ressentimento, arrependimento, culpa, etc. são sentimentos com conteúdo cognitivo. Não se pode simplesmente sentir vergonha, mas sim *vergonha de ter feito algo*.

A teoria da avaliação (LAZARUS, 1991) é descrita como fornecendo uma interpretação cognitiva do que sentimos ou percebemos. Esta teoria explica nossa avaliação de estímulos externos (ambientais) ou internos (dentro de nós) que dão origem a emoções. LIDA usa a Teoria da Avaliação junto com a Aprendizagem por Reforço, a Aprendizagem Livre de Modelos e a Aprendizagem Baseada em Modelos, como diferentes modalidades de aprendizagem.

2.5 Representações Motivacionais na Arquitetura SOAR

A AC SOAR é um mecanismo de busca e exploração de um espaço de estados, baseado em regras. A partir de um estado inicial, esse mecanismo busca aplicar operadores, que transformam o estado em um novo estado, e assim sucessivamente até que um estado final desejado seja alcançado. Com um conjunto de regras definidas a priori, o SOAR é capaz de realizar o processamento de informações de entrada, e consequentemente, tomar uma decisão e gerar um comportamento em sua saída visando a resolução de objetivos específicos. Um estado, é definido como uma árvore, onde cada folha da árvore corresponde a uma tripla (ID, ATTR, Value), conectando Identificadores (IDs), Atributos (ATTRs) e Valores (Values), chamada de "Working Memory Element"(WME). Particularmente, esses *valores* podem ser valores numéricos ou um identificador de outro WME. Assim, diversos WMEs podem ser conectados, compondo estados que podem ser tão sofisticados quanto o necessário. A Figura 3 demonstra a representação da árvore de WMEs presentes na "Working Memory" da AC SOAR.

O processo de busca, no SOAR, envolve a aplicação de operadores, transformando o estado inicial em um novo estado. Essa busca é feita por meio de um mecanismo de regras de produção. Entretanto, não devemos confundir regras com operadores. O mecanismo de busca usando operadores é construído a partir de um sistema de processamento de regras. Diferentes tipos de regras devem ser definidas, tais como regras para a proposição de operadores, regras para a aplicação de operadores, regras para definir a preferência na escolha do operador a ser aplicado, regras de derivação de conhecimento e regras para verificar se o estado desejado foi atingido. Para realizar a busca, o SOAR utiliza um mecanismo que realiza o processamento de regras, e que envolve um ciclo com 5 fases,

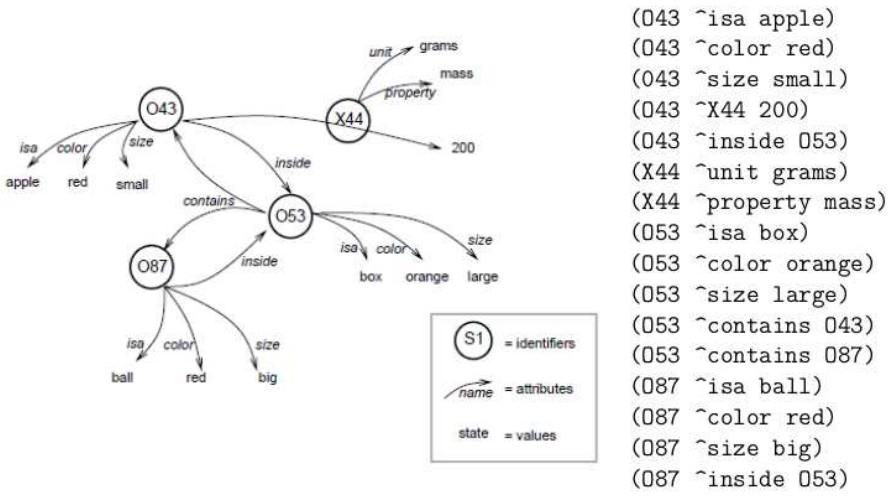

Figura 3 – Representação do WMEs na AC SOAR (LAIRD; CONGDON, 2014; LUCENTINI, 2017).

ou etapas, que se repetem de maneira periódica, até que uma etapa final HALT seja atingida. Dessa forma existem 6 fases possíveis no mecanismo de inferência do SOAR. As cinco etapas a seguir são executadas de maneira cíclica, e a 6a é a etapa HALT:

- INPUT.
- PROPOSE.
- DECISION.
- APPLY.
- OUTPUT.
- HALT.

O uso do SOAR na solução de um problema demanda portanto, em primeiro lugar, que o estado inicial seja definido, e que o conjunto de regras que irão realizar a busca esteja igualmente definido. A partir daí, inicia-se o procedimento de busca, onde sucessivos operadores devem ser aplicados ao estado inicial, até que o estado final desejado seja alcançado, quando a máquina de inferência do SOAR para. A partir daí, podemos averiguar qual foi a sequência de operadores aplicada, e esta constitui um plano para se alcançar o estado desejado a partir do estado inicial. Esse ciclo de inferência do SOAR envolve a aplicação cíclica das fases INPUT -> PROPOSE -> DECISION -> APPLY -> OUTPUT -> INPUT -> .... -> OUTPUT -> HALT, ou seja, o ciclo repete-se um certo número de vezes, sendo que cada sequência INPUT -> PROPOSE -> DECISION -> APPLY -> OUTPUT envolve a proposição, seleção e aplicação de um único operador,

causando uma mudança para um novo estado. O número de ciclos necessários para se chegar ao estado final determinará o tamanho do plano (em número de operadores), ou seja, o tamanho da sequência de ações necessárias para que o estado final seja atingido.

Vamos analisar cada uma dessas etapas/fases. Na etapa de "INPUT" o SOAR coleta os dados de entrada do sistema e preenche os WMEs no "input-link" do estado atual. Na fase "PROPOSE", são ativadas as regras de proposição de operadores, que devem verificar, a partir do estado atual, quais os operadores que têm condições de ser aplicados, ou seja, quais os ações que podem ser executadas, dado o estado atual. Nesta fase, o SOAR pode propor diversos operadores diferentes (vale a pena observar que no mecanismo do SOAR, os operadores também são definidos como WMEs que são agregados ao estado corrente). Embora possam haver mais do que um operador propostos, somente um operador pode ser selecionado e executado a cada ciclo. Esta seleção de operador é executada na fase "DECISION", onde o SOAR deve selecionar um único operador para a execução. Para determinar qual dos operadores propostos será escolhido, o SOAR faz o uso das regras de priorização entre operadores, que acabam por definir um único operador, ou a criação de um "impasse". Caso ocorra um "impasse", o SOAR cria um novo sub-problema, onde agora a tarefa é eliminar o impasse e decidir por um único operador e aplica de maneira recursiva o mesmo procedimento utilizado no problema principal, até que, ou um operador consiga ser escolhido, ou um limite de recursividade seja atingido, indicando que provavelmente o problema não tem solução, a partir do conjunto de regras definido. Isso pode implicar que novas regras devem ser desenvolvidas para que o problema possa ser resolvido. Caso um operador consiga ser determinado, passa-se então à próxima fase do procedimento de inferência, que é a fase "APPLY". Nessa, as regras de aplicação de operadores transformam o estado atual em um novo estado, modificando valores nos WMEs necessários para caracterizar o novo estado. Nesta fase, valores podem ser modificados ou WMEs inteiros podem ser deletados ou criados. Após a fase de "APPLY", o SOAR passa para a fase de "OUTPUT", onde WMEs podem ser escritas na estrutura "output-link". Caso alguma regra para verificar se o estado final foi atingido dispare (ou seja, execute o comando "HALT" em seu consequente), o SOAR passa para a fase "HALT" e finaliza execução. Caso contrario, a sequência reinicia na fase "INPUT" e um novo ciclo se inicia. A Figura 4 ilustra um exemplo de regra que propõe um operador no SOAR.

```
...
sp {blocks-world*propose*move-block
        (state <s>  ^problem-space blocks
                            ^thing <thing1> {<> <thing1> <thing2>}
                                ^ontop <ontop>)
        (<thing1> ^type block ^clear yes)
        (<thing2> ^clear yes)
        (<ontop> ^top-block <thing1>
                        ^bottom-block <> <thing2>)
-->
        (<s> ^operator <o> +)
        (<o> ^name move-block
                ^moving-block <thing1>
                ^destination <thing2>)
}
...
```

Figura 4 – Exemplo de regras utilizado pelo JSOAR durante a etapa de planejamento.

Diferentemente das demais arquiteturas anteriormente mencionadas, o SOAR não possui explicitamente nenhum módulo ou subsistem motivacional. Todavia, a arquitetura apresenta implicitamente características de caráter motivacional, que poderiam ser importantes para a construção e o desenvolvimento do modelo de subsistema motivacional do CST. Dentre essas características, destaca-se o fato de que o objetivo último do SOAR é chegar a um estado final, o que o transforma em um sistema de planejamento, e portanto algum tipo de motivação está implícito. Não é o objetivo deste trabalho, descrever o funcionamento de todas as estruturas e módulos presentes no SOAR. Porém, julgamos necessário exemplificar o funcionamento de alguns deles, tendo em vista um melhor entendimento das capacidades que este apresenta. A Figura 5 apresenta os módulos presentes na AC SOAR.

Cada módulo possui uma responsabilidade distinta na AC. Por exemplo, o Módulo de Percepção (ou *Perception Module*) tem a responsabilidade de capturar informações sobre os objetos existentes no mundo real (ou no ambiente da criatura virtual) e gerar as estruturas de entrada do ciclo cognitivo, que descrevem o estado atual (WMEs no "io.input-link"). Por outro lado, o Módulo de Ação (ou *Action Module*) identifica as tomadas de decisões (geradas pela AC a partir de seu ciclo cognitivo) coletando os WMEs gerados no "io.output-link" pelo ciclo cognitivo do SOAR e os transforma em ações efetivas no ambiente. A *Working Memory* é o repositório central de WMEs, sendo que existem diferentes mecanismos para modificar, remover e adicionar WMEs. O principal mecanismo é por meio das regras de produção. Entretanto, outros módulos da arquitetura também podem modificar os WMEs, pois atuam de maneira independente sobre a *Working Memory*. Dentre estes estão: a Memória Semântica (*Semantic Memory*), a Memória Episódica (*Episodic Memory*), o Módulo de Aprendizado por Reforço (*Reinforcement Learning Module*), além de outros. As regras de produção, definidas a priori no arquivo de configuração,

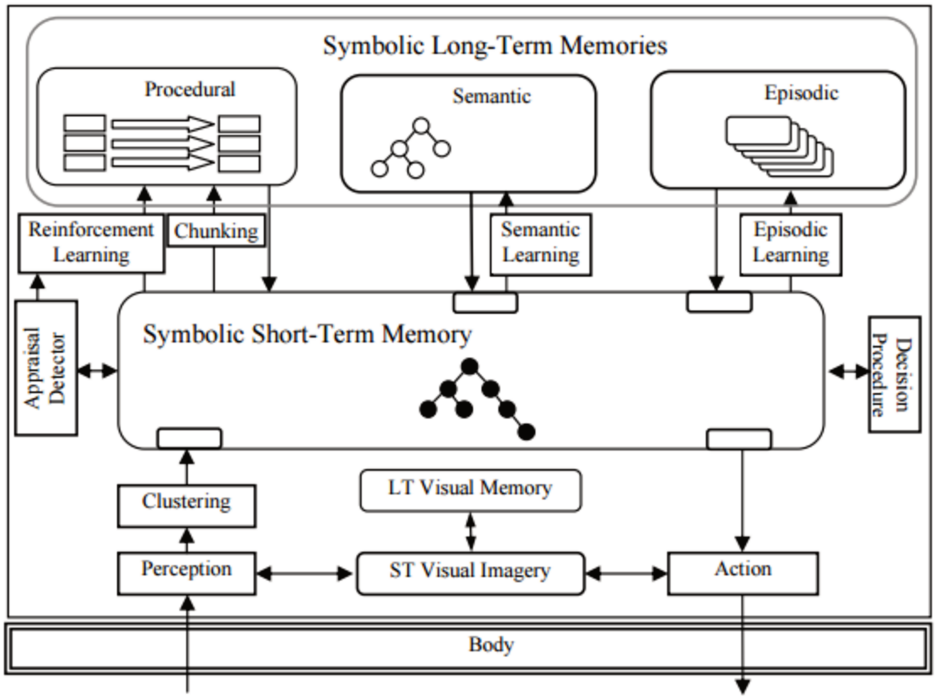

Figura 5 – A arquitetura cognitiva SOAR (LAIRD, 2008).

são carregadas pelo SOAR e armazenadas na Memória Procedural (*Procedural Memory*). Uma vez que as regras estejam na Memória Procedural, elas podem ser aplicadas pelo procedimento de decisão, consequentemente alterando a estrutura dos WMEs. Na Memória Semântica (*Semantic Memory*), são armazenados WMEs que envolvem conhecimento declarativo, ou seja, informações que representam conhecimentos gerais sobre elementos e objetos existentes do mundo. De acordo com a psicologia cognitiva, a memória semântica é um tipo de memória "Declarativa" que tem por finalidade armazenar informações de conhecimento geral, que podem ser referentes a: Tempo, Significados de Objetos, Entendimento sobre Eventos etc. Diferentemente da memória semântica, a Memória Episódica (*Episodic Memory*) é responsável por armazenar episódios e eventos que ocorreram ao longo dos diversos ciclos cognitivos do SOAR. Na realidade, para o SOAR, um episódio nada mais é do que uma "fotografia" incluindo todos os WMEs existentes na *Working Memory*, em um determinado instante de tempo, que é preservada historicamente e armazenada na memória episódica (LUCENTINI, 2017).

3 Modelo de Subsistema Motivacional

Neste trabalho, propomos um subsistema motivacional (SM) baseado em diferentes teorias sobre o comportamento motivacional e emocional disponível na literatura. Além da literatura sobre sistemas autônomos com comportamento emocional e motivacional, também investigamos diferentes ACs equipadas com estruturas motivacionais para obter ideias sobre os recursos e sutilezas a serem incluídos no projeto. Essas ACs geralmente possuem implementações computacionais em diferentes linguagens de programação, o que facilita o estudo e a compreensão de seus princípios internos através do estudo de seus códigos fonte. Foram estudados tanto tutoriais desenvolvidos pelos respectivos grupos de pesquisa, nas quais demonstram os seus recursos, como também o estudo do código-fonte juntamente com a execução e depuração de aplicativos de exemplo que estão disponíveis.

Nosso SM foi projetado para ser um subsistema do CST. O CST é um *toolkit* computacional que permite a construção de ACs de propósito geral. Ele foi desenvolvido na linguagem Java na Universidade de Campinas por diferentes pesquisadores sob a supervisão do Prof. Ricardo Gudwin. O *toolkit* depende de um conjunto de conceitos básicos que são descritos em (PARAENSE *et al.*, 2016). Entre estes conceitos, a noção de *codelet* é de extrema importância. Esse conceito foi originalmente introduzido por Hofstadter *et al.* (1994) e aprimorado por Franklin *et al.* (1998) e Franklin *et al.* (2014). Codelets são trechos de código executados de forma contínua e cíclica, sendo responsáveis por realizar qualquer atividade cognitiva em agentes artificiais. Codelets são os blocos de construção básicos utilizados para desenvolver a arquitetura mental de um agente cognitivo, implementando subsistemas para diversas funções cognitivas, tais como: atenção, percepção, emoções, aprendizado e outras. Os codelets do CST são bastante semelhantes aos usados na AC LIDA (FRANKLIN *et al.*, 2014; FRANKLIN *et al.*, 2016). Um segundo conceito central importante no CST é a noção de *Objetos de Memória* (ou *Memory Objects*), que são entidades conceituais (e computacionais) que representam diferentes tipos de memórias de um agente cognitivo. No contexto do CST, objetos de memória podem armazenar qualquer tipo de dados que serão utilizados por um codelet. Eles são usados como variáveis de entrada e saída dos codelets, e seu uso pode ser destinado para um armazenamento de longo ou curto prazo. Mais detalhes sobre o projeto CST podem ser encontrados em (RAIZER *et al.*, 2012; PARAENSE *et al.*, 2016).

Neste capítulo, detalhamos o design do Subsistema Motivacional. Conceitualmente, nosso modelo de Subsistema Motivacional é decomposto em duas partes. A primeira seção discute o Subsistema Motivacional de Subsunção - SMS (ou *Subsumption Motivational Subsystem - SMS*). A segunda seção, discute o Subsistema Motivacional de Alto Nível - SMAN (ou *High-Level Motivational Subsystem - HLMS*). A terceira seção

fornece uma visão integrada de todo o Sistema Motivacional. E por fim, a quarta seção, apresenta os detalhes de implementação do SM no projeto do CST.

3.1 O Subsistema Motivacional de Subsunção

Nosso SM foi construído em cima de diversas ideias coletadas da literatura, a partir das quais derivamos nossa proposta. Um dos fundamentos do nosso SM é a assim chamada *Arquitetura de Subsunção* (ou *Subsumption Architecture*). Com base nesta teoria, desenvolvemos a primeira parte de nosso modelo, o Subsistema Motivacional de Subsunção (SMS).

Arquitetura de Subsunção é um nome genérico para uma família de arquiteturas computacionais utilizadas no controle inteligente (particularmente na robótica), desenvolvida por Rodney Brooks na década de 90, dando origem à área de pesquisa em Robótica Comportamental (ARKIN, 1998). A proposta de Brooks foi desenvolvida no contexto da robótica móvel, sendo na época uma reação contra o programa de pesquisa em *Inteligência Artificial*, que sofria de uma série de limitações. Em vez de um *pipeline* serial envolvendo sensoreamento, percepção, modelagem interna, planejamento, execução de tarefas e atuação, como abordado na *Inteligência Artificial*, Brooks propôs uma estratégia paralela, denominada *Arquitetura de Subsunção*, onde diversos comportamentos competem entre si para ter acesso aos atuadores. Novos comportamentos podem ser desenvolvidos de forma isolada e posteriormente integrados na arquitetura.

Na proposta original de Brooks (BROOKS, 1986), cada comportamento é uma máquina de estados finitos, aumentada com variáveis de instância. Cada comportamento possui vários conexões de entrada e saída. Essas conexões são usadas para enviar e receber mensagens dos sensores e de outros comportamentos e daí para os atuadores.

A proposta original de Brooks também incluía nós de supressão e inibição. Estes eram utilizados para garantir que um nó fosse capaz de inibir ou suprimir um comportamento especifico em um determinado instante de tempo. Após o trabalho original, Connell (1989) propôs uma versão ligeiramente modificada da arquitetura de subsunção, em que, em vez de enviar mensagens, ou seja, comandos baseados em eventos, os comportamentos enviariam um sinal contínuo. A modificação introduzida por Connell tornou-se popular e obteve ampla aceitação dentro da comunidade, tornando-se a implementação padrão da arquitetura de subsunção a partir daí. A modificação de Connell tornou o nó inibitório obsoleto e, a partir disso, a maioria das arquiteturas de subsunção começou a usar apenas nós de supressão. Uma vez que os nós de supressão eram o único tipo de nó conector entre duas linhas, alguns autores decidiram não mais mostrar o nó de supressão de maneira explícita em sua notação.

Além disso, a nova notação tornou explícita outra característica importante de

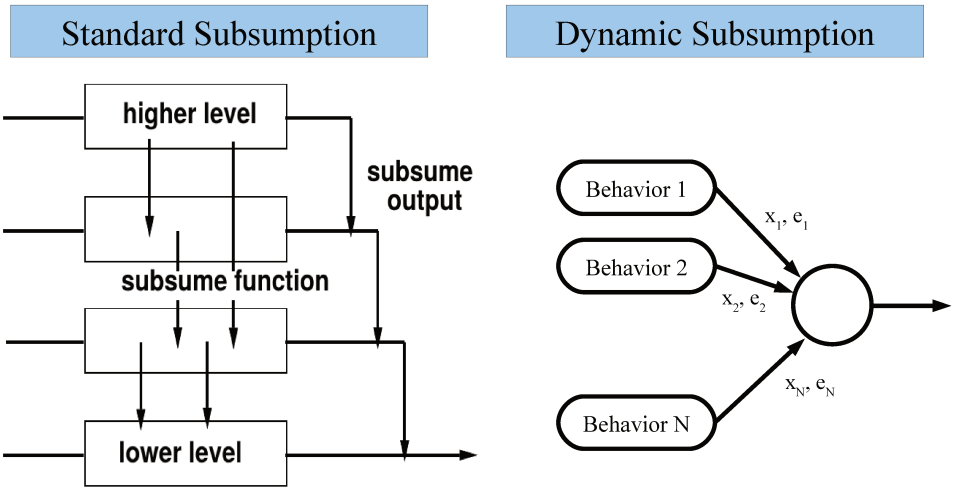

Figura 6 – Arquitetura de Subsunção Padrão x Subsunção Dinâmica

uma arquitetura de subsunção, que é a sua decomposição em *camadas*. Camadas estão relacionadas diretamente ao comportamento que alimenta a entrada dominante de um nó de supressão. O comportamento que alimenta a entrada não dominante fica em uma camada inferior. Em contrapartida, o comportamento que alimenta a entrada dominante fica em uma camada superior. Passou-se a representar a entrada dominante de um nó de supressão simplesmente como uma seta finalizando no meio de uma linha reta, como demonstrado no lado esquerdo da Figura 6. É importante observar que mesmo que os nós de supressão não sejam mostrados explicitamente, eles ainda estão lá.

O campo da robótica baseada em comportamento evoluiu muito desde a introdução da arquitetura de subsunção. Diversas variações começaram a surgir, incluindo os assim chamados "esquemas motores", além de outras arquiteturas reativas (ARKIN, 1998). A arquitetura de subsunção ainda é muito popular, e suas versões mais modernas geralmente constituem uma arquitetura híbrida com algum tipo de extensão.

Uma possível limitação na arquitetura padrão de subsunção é que os nós de supressão possuem nós dominantes e não dominantes que são sempre fixos. Isso significa que, um comportamento de nível superior sempre terá prioridade sobre outro de nível inferior. Apesar disso ser desejável, em um grande número de situações, podem existir situações onde seria interessante se reverter essa prioridade. Isso não é possível na arquitetura de subsunção padrão.

Para lidar com essa dificuldade, alguns autores (NAKASHIMA; NODA, 1998; HAMADI *et al.*, 2010; HECKEL; YOUNGBLOOD, 2010) propuseram um esquema de *Subsunção Dinâmica*, no qual não há entrada dominante fixa em um nó de supressão, mas essa dominância pode ser alterado dinamicamente no tempo, de acordo com situações específicas.

Existem diferentes implementações deste conceito de Subsunção "Dinâmica", mas

uma compreensão muito simples dessa ideia pode ser ilustrada na Figura 6. No lado esquerdo desta Figura, temos uma arquitetura de subsunção padrão, com sua prioridade fixa entre os comportamentos. No lado direito, temos uma possível implementação para o esquema de Subsunção Dinâmica. Juntamente com a mensagem de controle padrão x_i, adiciona-se uma informação de avaliação e_i que é gerada dinamicamente pelo próprio comportamento. Portanto, ao invés de escolher o valor de saída x usando um conjunto fixo de prioridades, o mecanismo dinâmico simplesmente escolhe o x_i que possui o maior valor de e_i. Esse valor e_i deve ser gerado dinamicamente de acordo com os requisitos do sistema.

A subsunção dinâmica é a maneira padrão pela qual o CST implementa comportamentos reativos. Nosso SMS implementa uma variação desse esquema de forma a implementar um primeiro nível de comportamento motivado (que será complementado, posteriormente, como veremos, pelo Sistema Motivacional de Alto Nível). Como vimos anteriormente na teoria de Hull (1943), um comportamento motivado é explicado em termos das *necessidades* (ou seja, dos *drives*) que motivam uma ação correspondente no meio ambiente. Sendo assim, essa ação motivada deve causar uma satisfação da necessidade correspondente e, portanto, uma redução do *drive* correspondente. Dessa forma, em alinhamento com a literatura, nosso *drive* deve ser aqui entendido como uma medida do nível de insatisfação de uma *necessidade*, que motiva o comportamento do agente cognitivo durante determinado instante de tempo. Nossa criatura artificial pode ainda apresentar diferentes *necessidades*, com diferentes níveis de intensidade, ao longo do tempo e portanto faz parte das atribuições do SMS, avaliar as intensidades dessas necessidades e escolher uma delas para ser atendida, gerando uma ação condizente.

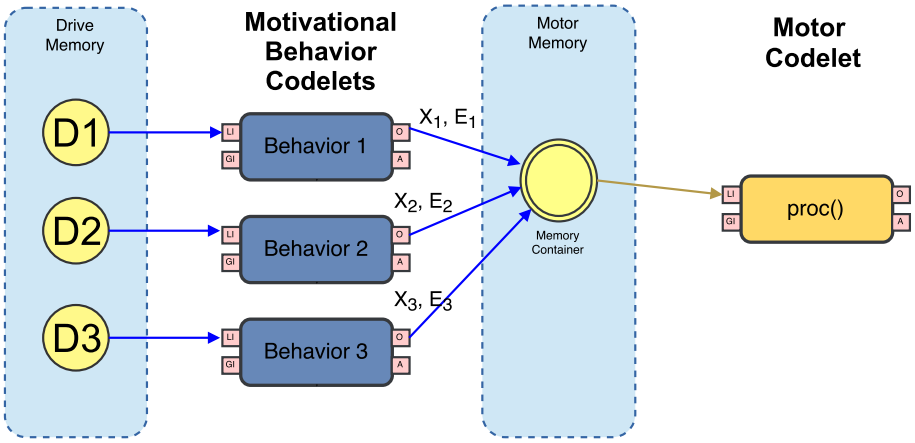

Figura 7 – Comportamentos Motivacionais sendo orientados por *drives* (FRóES, 2017).

A Figura 7 ilustra essa ideia de como os *drives* são usados para motivar um comportamento. Vale a pena ressaltar que os conceitos de codelets e objetos de memória do CST são utilizados para compor nosso SMS. Na figura, codelets são representados como

caixas arredondadas com duas entradas (LI - *Local Input* e GI - *Global Input*) e duas saídas (Out - *Standard Output* e A - *Activation*). Os objetos de memória são representados como círculos dentro de caixas arredondadas com linhas pontilhadas, representando as diferentes memórias do sistema (veja (PARAENSE *et al.*, 2016) para obter uma descrição mais elaborada sobre codelets e objetos de memória). A ideia representada na figura, é que diferentes necessidades, representadas pelos *drives* D1, D2 e D3 irão gerar comportamentos alternativos e portanto comandos motores alternativos para um mesmo atuador X. Na figura, os *drives* D1, D2 e D3 são utilizados para produzir três objetos de memória distintos $\{X_i, E_i\}$, onde X_i é um sinal motor e E_i um avaliador. Os três codelets comportamentais alimentam um único objeto de memória na memória motora. Esse objeto, ao contrário dos demais, é uma memória do tipo *Container*. *Containers* são um tipo especial de objeto de memória, que podem receber entradas de diversas codelets diferentes, fazendo uma seleção do objeto de memória que será transmitdo adiante. Assim, para escolher qual sinal motor X_i será transmitido para o Codelet Motor o objeto do tipo *Container* seleciona dentre seus objetos de memória de entrada, aquele com o maior valor de E_i.

Em nosso modelo de SMS, os *drives* encontrados na figura 7 são objetos de memória gerados anteriormente por *Codelets Motivacionais* (ou *Motivational Codelets*) através de duas maneiras:

i). diretamente de variáveis sensoriais ou

ii). derivados de outros *drives* existentes no agente.

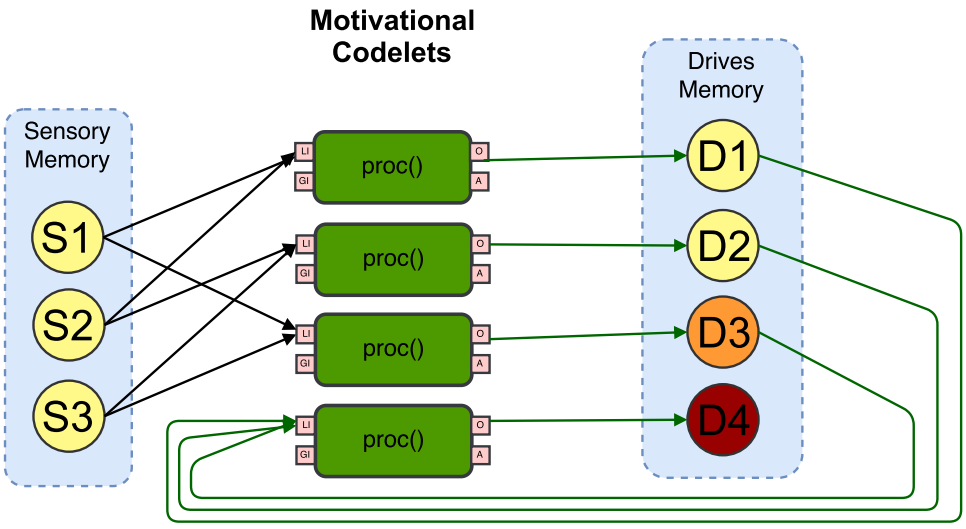

Figura 8 – Drives sendo gerados pelos Codelets Motivacionais (FRóES, 2017).

A Figura 8 demonstra diferentes maneiras como Codelets Motivacionais podem gerar um conjunto de *drives*. D1 e D2 são *drives* primários de baixo nível, D3 é um drive primário de alto nível e D4 é um drive secundário.

Diferentes ACs (SUN, 2009; MCCALL, 2014) propõem uma distinção funcional entre *drives* primários de alto nível e de baixo nível, e também *drives* secundários. Se todos os *drives* fossem funcionalmente exatamente iguais, poderia haver situações em que o sistema poderia dar preferência a um drive social, como por exemplo "aceitação social", ao invés de dar preferência a um drive fisiológico como a fome. Isso é aceitável em uma situação normal, mas se ambos os *drives* estivessem com intensidades muito altas, que poderíamos considerar como em um estado de "urgência", o sistema correria o risco de colapsar (imagine uma situação em que o nível energético do agente estivesse muito próximo de zero). Em situações críticas como essa, parece-nos óbvio que um drive de fome deva ter algum tipo de prioridade, devido às potenciais consequências desastrosas para o sistema (se o sistema colapsar por falta de energia, uma "urgência" social não tem nenhum significado). Em sistemas como o CLARION, essa situação é resolvida utilizando-se diferentes mecanismos para drives de baixo nível e alto nível. Com isso, o CLARION só considera os *drives* de alto nível se os *drives* de baixo nível estiverem com suas intensidades abaixo de um nível crítico. Caso algum *drive* de baixo nível tenha uma intensidade muito alta, os drives de alto nível não são considerados na competição, mesmo que tenham também uma intensidade alta. Isso garante que quando *drives* de baixo nível estejam com uma intensidade em nível crítico, o mesmo só concorrerá com outros *drives* de baixo nível. Essa solução resolve um grande número de problemas, mas não resolve a situação em que diferentes *drives* de baixo nível estejam com intensidade muito alta, e um desses *drives* seja mais crítico que o outro, do ponto de vista da sobrevivência do sistema. O *drive* vencedor será aquele com a intensidade mais elevada, mesmo que um deles seja mais crítico do que o outro. Para resolver esse problema, criamos em nosso SMS um tratamento especial para situações de "emergência", que consegue superar as deficiências encontradas no tratamento dado pelo CLARION. Este mecanismo funciona da seguinte forma. Além do *Nível de Atividade*, ou intensidade, já associado a um drive, incluí-se agora mais dois parâmetros: *Limiar de Urgência* e *Prioridade*. O *Limiar de Urgência* é um valor limite para o *Nível de Atividade* que, uma vez que esse valor seja alcançado, o drive é considerado em estado de urgência. No estado de urgência, ao invés de usar o *Nível de Atividade* para selecionar entre diferentes *drives*, o sistema usa a *Prioridade* atribuída ao drive, o que garantirá que o drive correto ganhe a competição pela execução do comportamento. Os detalhes deste mecanismo são explicados mais adiante no texto.

Antes de continuar, é necessário mencionar como os *drives* secundários são computados, quando processados pelos codelets motivacionais (veja um exemplo na figura 8). Como já dissemos, os *drives* secundários são diretamente dependentes de outros *drives*. Para calcular o nível de atividade de um drive secundário, os codelets motivacionais realizam uma média ponderada de seus *drives* de entrada, moduladas por um fator *Relevância*. O atributo de *Relevância* é uma variável numérica que varia entre 0 e 1, e que é multiplicada pelo nível de atividade do drive de entrada para calcular o nível de atividade

do drive secundário.

Finalmente, para modelar completamente nosso SMS, deve-se incluir o efeito das emoções. Para isso, é utilizado o modelo de emoções da Cañamero (1997) onde as emoções são vistas como distorções cognitivas no mapa dos *drives*, alterando por um determinado momento a intensidade relativa dos diferentes *drives* do sistema, ampliando a intensidade de alguns *drives* e diminuindo a de outros. Para esse propósito, incluímos outro parâmetro na concepção formal do drive: a *distorção emocional* (ou *distorção cognitiva*) . A *distorção emocional* é um valor (positivo ou negativo) que é somado com o *Nível de Atividade* de um drive, tendo em vista priorizar um comportamento motivado. Na arquitetura, a definição desta *distorção emocional* é atribuída a Codelets Emocionais, como ilustrado na Figura 9

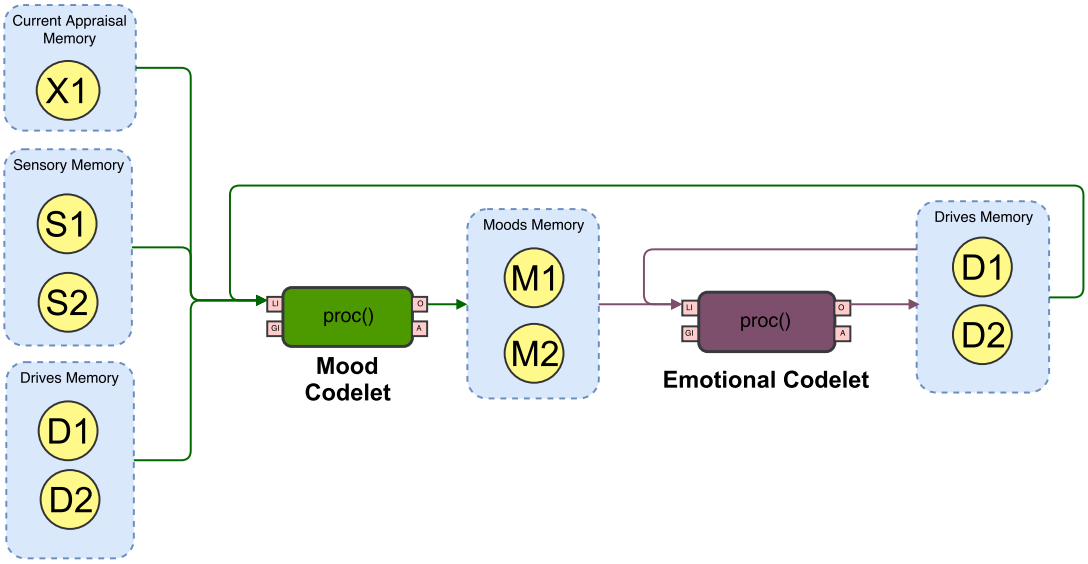

Figura 9 – Os Codelets Emocionais aplicando a distorção cognitiva nos drives.

Os codelets emocionais são modulados pelo *Humor* (ou *Moods*) do Agente. Os Moods são um tipo de estado emocional determinado pelos *Codelets de Humor* (ou *Mood Codelets*), com base em dados sensoriais adquiridos do meio ambiente, pelos *drives* existentes no agente cognitivo e/ou a partir da avaliação primária gerada pelos *Codelets de Avaliação* (a seção 3.2 demonstra detalhadamente as funções deste codelet). Dependendo do humor *normal*, ou em um humor alternativo como *sonolento*, *preocupado*, *ansioso*, *exaltado*, *aterrorizado*, *apaixonado*, etc., o codelet emocional pode determinar uma distorção cognitiva no escopo dos *drives*, gerando uma prioridade diferente para selecionar o comportamento motivado.

Após essa breve descrição funcional do modelo, de nosso SMS, seguimos com um modelo formal para suas diversas componentes.

Definição 3.1 *Um* drive *d é definido como uma tupla $d = \{A, \theta, \delta, p\}$ onde:*

- *A é o* nível de ativação *representando a intensidade do drive, $A \in [0, 1]$*

- θ *é o* limite de urgência *de modo que se* $A > \theta$ *então o drive está no estado de urgência,* $\theta \in [0, 1]$.
- δ *é a* distorção emocional *que deve ser adicionada a* A *para computar a intensidade do drive,* $\delta \in [-1, 1]$ *e* $0 \leq (A + \delta) \leq 1$.
- p *é a* prioridade, $p \in [0, 0.5]$, *utilizada para computar o valor final da intensidade do* drive *quando este se encontra em um estado de urgência.*

Deste modo, com base nesta definição de *drive*, pode-se definir formalmente como o cálculo do valor *Eval* nos Objetos de Memória (gerado internamente pelos Codelets de Comportamento Motivacional) deve ser realizado:

Definição 3.2 *O cálculo do parâmetro* Eval *de um Objeto de Memória gerado por um Codelet Motivacional deve adotar os seguintes critérios:*

$$Eval = \begin{cases} 0.5 + p & \text{if } A > \theta, \\ (A + \delta_a)/2 & \text{if } A \leq \theta. \end{cases} \tag{3.1}$$

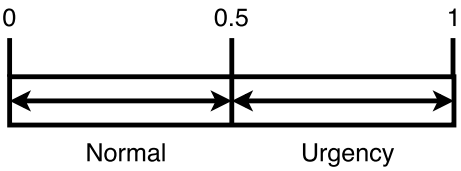

Figura 10 – Valor de Eval no estado normal ou de urgência.

A definição 3.2 implicitamente cria duas classes de comportamento, dependendo-se do valor da variável *Eval*. Para valores entre 0 e 0.5, entende-se que o *drive* está operando em modo normal. Caso os valores da intensidade do *drive* estejam entre 0.5 e 1, o mesmo está operando em modo de urgência. Usando esta convenção, podemos garantir que, quando o *drive* estiver operando em estado de urgência, o drive com a maior prioridade sempre será selecionado pelo mecanismo de subsunção dinâmico, independente do nível de atividade do mesmo. Isto é retratado na figura 10.

Dessa forma, ao contrário do que acontece com o CLARION, em nosso SMS podemos ter toda uma hierarquia de prioridades, para situações de urgência. Nesses casos, o valor da prioridade é que será decisivo na determinação da intensidade do *drive*, e não o valor da atividade A. Isso não é possível na atual implementação do CLARION.

O processo de concepção da definição 3.2 envolveu o estudo de diversas opções diferentes, sendo que a mesma foi concebida, reestruturada e testada diversas vezes até que pudéssemos convergir para a versão atual, incorporando apropriadamente (a nosso ver) os

conceitos sobre motivação e emoção elencados no trabalho, ou seja, tivemos que adaptar a equação de tal forma que as diversas propriedades de um drive (limite de urgência, nível de ativação e distorção emocional), pudessem ser incorporadas de maneira adequada. Além disso, a equação apresentada na definição foi aquela que produziu os melhores resultados.

3.2 O Subsistema Motivacional de Alto Nível

A arquitetura de subsunção é muito eficaz para criar um mecanismo flexível e eficiente para a coordenação do comportamento. O Subsistema Motivacional de Subsunção descrito na seção anterior fornece um mecanismo importante para o comportamento propositado em uma AC. Mas, apesar disso, existem diversas limitações conhecidas. Quando é possível determinarmos um comportamento que garantidamente leve a uma redução do *drive*, a arquitetura de subsunção funciona muito bem. Isso ocorre, por exemplo, em problemas de otimização onde o problema é convexo e existe somente um mínimo local. Entretanto, muitas vezes, pode ser que a superfície de necessidade possua múltiplos mínimos locais, e o sistema tenha que procurar um mínimo global. Pode acontecer, ainda, que o sistema tenha que lidar com situações excepcionais, que demandem um tratamento diferente do comportamento padrão. Pode ser necessário, ainda, que para atingir um estado desejado, o sistema necessariamente necessite de uma estratégia que demande a execução de uma sequência de ações diferentes, levando o sistema a diversos estados intermediários antes de atingir o estado desejado. Em todas essas situações, a arquitetura de subsunção pode ser insuficiente para garantir um comportamento adequado.

Para complementar nosso modelo de Subsistema Motivacional, de modo a que o mesmo esteja apto a lidar com estas situações, propomos um subsistema de motivação de alto nível, que deve funcionar de maneira coordenada com o subsistema motivacional de subsunção, e seja capaz de lidar com tais condições excepcionais.

O subsistema motivacional de alto nível conta com diversos tipos de codelets:

- Codelet de Planejamento (*Planning Codelet*)
- Codelet de Seleção de Planos (*Plan Selection Codelet*)
- Codelet de Apreciação de Situação Corrente (*Current Appraisal Codelet*)
- Codelet de Apreciação de Situações Hipotéticas (*Hypothesis Appraisal Codelet*)
- Codelet de Geração de *Goals* (*Goal Codelet*)

3.2.1 Codelets de Planejamento e de Seleção de Planos

Nosso subsistema motivacional de alto nível utiliza um sistema de planejamento simbólico baseado em regras, capaz de detectar situações específicas e executar um plane-

jamento no espaço de estados, até que um estado desejado seja atingido. Em nosso caso, utilizaremos para isso o SOAR, disponibilizado para o CST por meio do assim chamado Soar Codelet. O SOAR (LAIRD, 2012) é uma arquitetura cognitiva simbólica desenvolvida por John Laird na Universidade de Michigan para realizar a exploração de um espaço de estados, por meio da aplicação de operadores a um estado inicial. Esses operadores são definidos na forma de regras de produção, e transformam o estado atual em um próximo estado. A partir do estado inicial, um conjunto de operadores é proposto, sendo que a partir de regras de seleção entre operadores, o SOAR decide qual o operador aplicar à situação atual, transformando o estado atual em um novo estado. O algoritmo é então aplicado de maneira recursiva, até que um estado final desejado é atingido, interrompendo o processo de busca. As regras no SOAR são de diferentes tipos. Há regras para identificar possíveis operadores a serem aplicados, regras para a seleção de operadores quando mais de um operador for possível e regras para ocasionar mudanças no estado atual. Os estados no SOAR são definidos a partir dos assim chamados WMEs *Working Memory Elements*, que podem formar verdadeiras árvores, a partir de triplas do tipo (identificador, atributo, valor). O SOAR Codelet no CST, permite transformarmos todo o comportamento do SOAR em um codelet, que pode ser integrado com outras funcionalidades do SOAR, traduzindo objetos do CST em WMEs, executando o ciclo de inferência do SOAR e posteriormente transformando novamente as WMEs de saída do SOAR, incluindo a sequência de operadores encontrada pelo SOAR, funcionando como um plano de ação, permitindo o uso de seus recursos de planejamento baseado em regras, para que os mesmos sejam integrados a uma AC sendo desenvolvida com CST. Conforme podemos ver na figura 11, o subsistema de motivação de alto nível, é divido entre um codelet de Planejamento (implementado por meio do SOAR), que a partir da situação atual fica gerando planos de ação que alcancem alguma das metas designadas, e uma memória de trabalho (*Working Memory*), dividida em um conjunto de submemórias contendo elementos específicos de conhecimento que são úteis para proporcionar comportamento motivacional. Essa memória de trabalho é disponibilizada para o SOAR, e ao final de seu ciclo de inferência, os WMEs do SOAR são traduzidos novamente para objetos de memória na Memória de Trabalho.

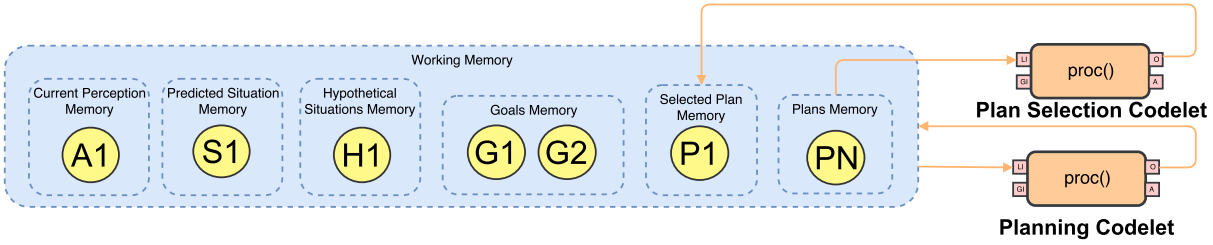

Figura 11 – Os Codelets de Planejamento e Seleção de Planos interagindo com as subestruturas da Memória de Trabalho.

3.2.2 Codelets de Apreciação

Em nosso modelo de subsistema motivacional de alto nível, temos dois Codelets de apreciação: o codelet de apreciação de situação corrente e o codelet de apreciação de situações hipotéticas. Antes de explicar cada um desses codelets, é importante trazer alguns subsídios da Teoria da Apreciação (*Appraisal Theory*) em psicologia e ciências cognitivas. De acordo com Lazarus (1991) e McCall (2014), a Teoria da Apreciação descreve um processo de apreciação/avaliação cognitiva que ocorre sobre cada percepto obtido por meio da percepção. A teoria da apreciação fornece uma explicação sobre a forma como cada estímulo externo (objetos no ambiente) ou interno (estruturas mentais internas) dá origem simultaneamente a um processo avaliativo, e como este processo está ligado às emoções.

Roseman e Smith (2001) fazem uma conexão entre essas avaliações e motivações/objetivos humanos, estendendo a noção de motivação para além de simplesmente os *drives* até agora utilizados. Lazarus (1991) propõe dois tipos de métodos de apreciação/avaliação: i) avaliação primária, quando o organismo humano determina a significância e a importância de um evento; ou ii) avaliação secundária, quando o organismo humano determina se pode ou não controlar as consequências do evento. Com a estrutura do subsistema motivacional de alto nível, é possível implementar ambas propostas defendidas por Lazarus (apesar de que, nos experimentos realizados neste trabalho - vide o capitulo 4 - exploramos somente o caso da avaliação primária). Em nosso subsistema motivacional de alto nível, re-interpretamos a proposição original de Lazarus em dois tipos de apreciações:

- Apreciação da Situação Corrente.
- Apreciação de uma Situação Hipotética.

A apreciação da situação corrente basicamente utiliza a percepção da situação corrente para avaliar o grau de satisfação do agente com essa situação corrente. Em um primeiro momento, poderíamos imaginar que essa avaliação é semelhante a um *drive*, mas ela pode considerar outras questões. Por exemplo, podemos utilizar o Codelet de Apreciação da Situação Corrente para avaliar a capacidade de predição do sistema, ou seja, para avaliar o quanto a situação atual se assemelha com o que se poderia prever da mesma. Dessa forma, a situação corrente é comparada com alguma possível predição da situação corrente, disponível na memória de situações previstas, e se a situação percebida atual é próxima do que poderíamos antecipar, esta avaliação será positiva. Se a situação percebida atual for muito diferente da situação predita, isso significa que o sistema foi incapaz de predizer a situação atual, e portanto a avaliação será negativa, indicando que o sistema deve aprender alguma coisa nova. O codelet de apreciação da situação corrente pode ainda utilizar os *drives* gerados no SMS para compor sua apreciação da situação, ou

diretamente as informações oriundas da memória sensorial. O mecanismo final acaba sendo dependente da aplicação, e utilizará os recursos necessários para avaliar a desejabilidade da situação corrente. Esse mecanismo está ilustrado na figura 12

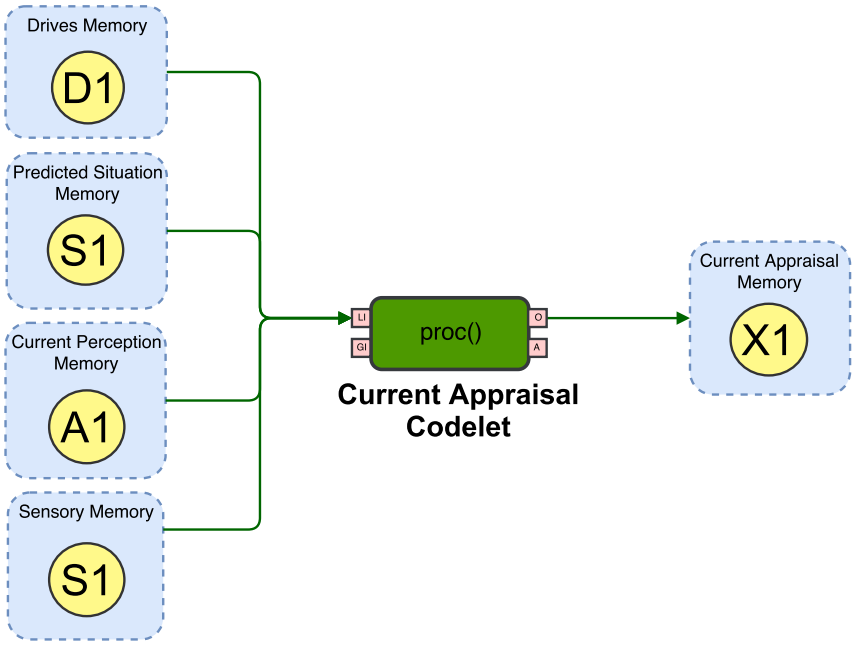

Figura 12 – Codelet de Avaliação realizando a avaliação primária do agente através da memória sensorial.

A apreciação de situações hipotéticas, de certa forma, considera a noção de avaliação secundária de Lazarus, ou seja, a determinação das etapas necessárias para que uma situação imaginada no futuro se torne verdadeira. A ideia é que o SOAR ou outro módulo seja capaz de gerar situações futuras hipotéticas e o subsistema motivacional de alto nível precisa avaliar a conveniência dessas possíveis situações hipotéticas futura. Isso é realizado pelo Codelet de Apreciação de Situações Hipotéticas, conforme apresentado na figura 13.

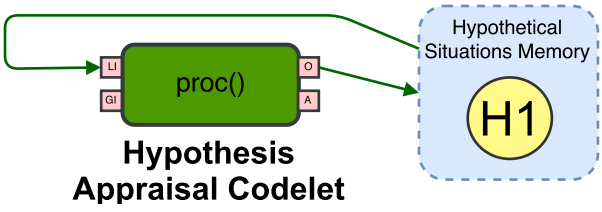

Figura 13 – Hypothetical Situations Appraisal.

3.2.3 Codelet de Criação de *Goals*

Basicamente, o que o Codelet de Apreciação de Situações Hipotéticas está fazendo é "marcar" a representação de uma situação hipotética com uma avaliação de sua

desejabilidade. A partir daí, um outro codelet de criação de *goals* (*Goal Codelet*) pode analisar diversas situações hipotéticas e escolher uma delas como um *goal* a ser buscado. Este *goal* corresponde a um estado futuro desejado, que o codelet de Planejamento irá utilizar para gerar possíveis planos de ação. Além das situações hipotéticas, o Codelet de Criação de Goals pode ainda utilizar dados da memória perceptual corrente e da memória sensorial. O mecanismo de geração de *goals* pelo Codelet de Geração de *Goals* está ilustrado na figura 14.

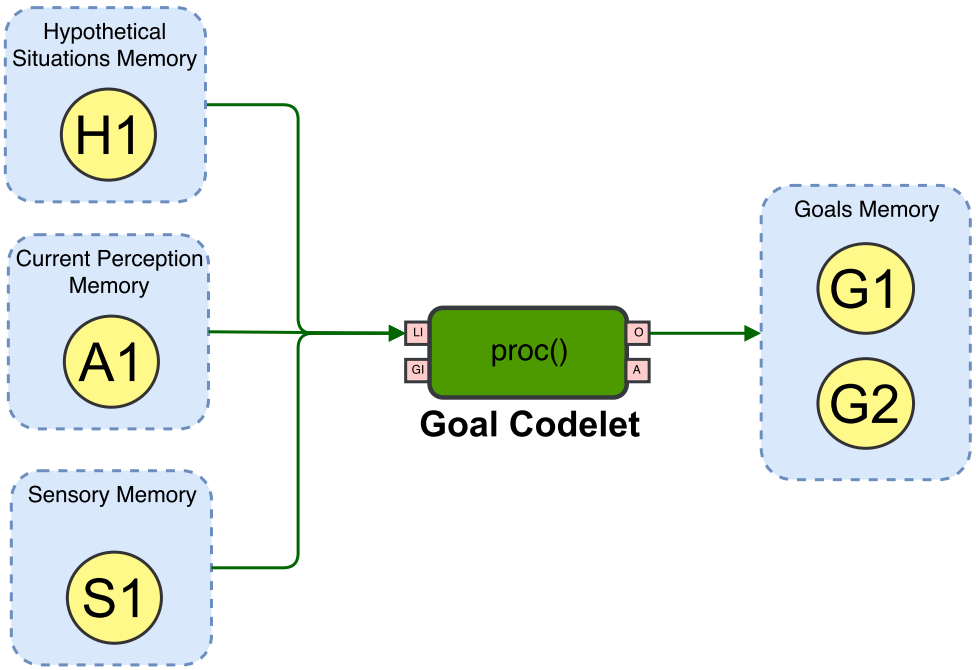

Figura 14 – Codelets de Criação de Goals (ou Goal Codelets).

O objetivo do Codelet Gerador de *Goals* é analisar as diversas possíveis situações hipotéticas disponíveis na Memória de Situações Hipotéticas, juntamente com os perceptos existentes na Memória Perceptual Corrente, e a partir da consideração da atual situação, encontrar uma situação hipotética futura que seja considerada satisfatória e portanto se transforme em um *goal* do sistema. Isso implicará na transformação dessa situação hipotética em um *goal* que é enviado para a a Memória de *Goals*. Esses *goals* serão então usados pelo Codelet de Planejamento para gerar planos de ações que possam atingí-los.

3.2.4 Integrando os Diferentes Codelets

A partir do momento que um plano é gerado, ele é armazenado na Memória de Planos. Nesta memória, ficam armazenados todos os planos gerados pelo Codelet de Planejamento, aguardando o momento em que um plano seja escolhido para execução. Para que um plano seja executado, o Codelet de Seleção de Planos (ou Plan Selection Codelet) faz a seleção de um dos possíveis planos da memória de planos e o aloque para execu-

ção, colocando-o na memória de planos selecionados para execução. A forma como este codelet faz a seleção depende diretamente da aplicação escolhida. Este Codelet, além de selecionar planos, apresenta algumas funções auxiliares relacionadas aos planos existentes na memória de planos, como: verificar se os planos existentes na memória são factíveis de serem executado, validar se um novo plano apresenta similaridade com outro na memória de planos, e claro, selecionar um plano para ser executado. A partir do momento, que um plano é selecionado pelo Codelet de Seleção de Planos, o mesmo é enviado para a Memória de Planos Selecionados, e a partir daí, é enviado para os Codelets de Comportamento para ser executado. Nesse instante, ele entrará em competição com os comandos enviados pelo SMS. A decisão final de quem deve afetar os atuadores é realizada dentro dos codelets comportamentais, e depende de diversos fatores. Podemos ilustrar uma situação como essa imaginando o sistema de controle de respiração que temos no corpo humano. Em princípio, nossa respiração é completamente inconsciente (o que aconteceria em nossa AC como quando os codelets comportamentais resolvem ouvir os comandos enviados pelo SMS). Entretanto, em algumas situações, é possível controlar conscientemente a respiração. Esse controle, entretanto, tem limites. Se decidirmos parar nossa respiração, podemos fazê-lo por um tempo, mas após um determinado tempo, nosso sistema inconsciente retoma o controle e nos faz respirar, independente de nossa vontade consciente. Esse é o tipo de relacionamento que o SMS tem com o sistema motivacional de alto nível. Os comandos vindos do sistema motivacional de alto nível em princípio têm prioridade sobre os comandos oriundos do SMS. Entretanto, em determinadas situações críticas (comandadas pelo nosso *drive* de sobrevivência) o SMS retoma o controle, independente da ordem dada pelo sistema motivacional de alto nível.

Em um primeiro olhar, o leitor poderia perguntar-se qual é a diferença entre um *goal* no subsistema motivacional de alto nível e um *drive* no SMS. Podemos esclarecer essa diferença usando o CLARION como exemplo. Nos módulos da CLARION (SUN, 2009), há dois tipos de representações internas: explícitas ou implícitas. As representações explícitas são simbólicas, e geralmente relacionadas a algum tipo de processamento baseado em regras. Já as representações implícitas são subsimbólica, e geralmente estão associadas a uma rede neural. Isso acontece não apenas no ACS e NACS, mas também no MS (Motivational Subsystem). As estruturas motivacionais implícitas estão diretamente ligadas a estruturas mentais internas, como *drives* básicos, necessidades básicas, desejos básicos e motivos intrínsecos. Eles são considerados como implícitos, porque não são facilmente acessíveis cognitivamente (HULL, 1943). Além disso, os processos motivacionais implícitos são considerados mais perenes do que processos motivacionais explícitos (SUN, 2001).

Anderson e Lebiere (1998) propõem que representações motivacionais explícitas consistem em objetivos explícitos de um agente. Objetivos explícitos referem-se a ações específicas e tangíveis, com uma conclusão. Motivações implícitas são mudadas constante-

mente, dependendo da situação. As motivações explícitas persistem até que seus objetivos sejam alcançados (EPSTEIN, 1982; SUN, 2009). Assim, tanto Drives quanto Goals estão relacionados a motivações, mas de diferentes maneiras. Os *drives* são mais como uma medida das necessidades de um sistema e modelam um conjunto de motivações mais perenes para o sistema que está sendo controlado. Os *goals*, diferentemente de *drives*, modelam uma espécie de motivação que tem uma conclusão. Depois que um *goal* é alcançado, ele cessa sua influência sobre o sistema. Em outras palavras, *drives* são usados para fins de longo prazo do sistema, algo que sempre será importante para motivar o comportamento do sistema, uma vez que uma necessidade se torna mais intensa. Os *goals* são mais como motivações que têm um fim, que cessa quando o objetivo é atingido. Juntos, *drives* e *goals* são importantes para a construção do modelo de subsistema motivacional proposto por esse trabalho. Os *drives* são utilizados para configurar um comportamento padrão. Já os *goals* são usados para responder a situações excepcionais que requerem medidas especiais, mas depois que essas motivações são satisfeitas, o sistema pode retornar ao seu modo operacional padrão.

3.3 Visão Geral do Subsistema Motivacional

Depois de apresentar cada um dos componentes de nosso modelo de subsistema motivacional, de maneira isolada, é possível propor uma visão integrada do mesmo. A Figura 15 demonstra uma visão geral do subsistema motivacional desenvolvido neste trabalho.

O Subsistema Motivacional completo integra o SMS e o SMAN. Sendo assim, este modelo é capaz de lidar com *drives* (medições do nível de insatisfação de necessidades que se mostram perenes no funcionamento do sistema) e *goals* (situações excepcionais particulares que se espera que aconteçam, e que uma vez atingidas, deixam de fomentar o comportamento do sistema).

Por fim, é necessário detalhar como esses dois sub-sistemas interagem um com o outro. Como já explicamos brevemente anteriormente, isso acontece nos Codelets de Comportamento Motivacional. A ideia é que esses codelets recebem *drives* conectados a eles e usarão as informações destes *drives* para escolher entre os comportamentos disponíveis para satisfazer as necessidades do agente cognitivo. Além disso, eles também recebem informações da Memória que contém o Plano Selecionado, uma das sub-memórias da Memória de Trabalho (*Working Memory*). Esta memória armazena "comandos" que precisam ser satisfeitos, gerados pelo sistema de planejamento. A ideia é que isso deve ser usado apenas em situações excepcionais. Se ocorrer uma situação excepcional, o sistema de planejamento a partir do SOAR gerará um plano para satisfazer a exceção, o Codelet de Seleção de Planos selecionara um plano a ser executado e o colocará na Memória de

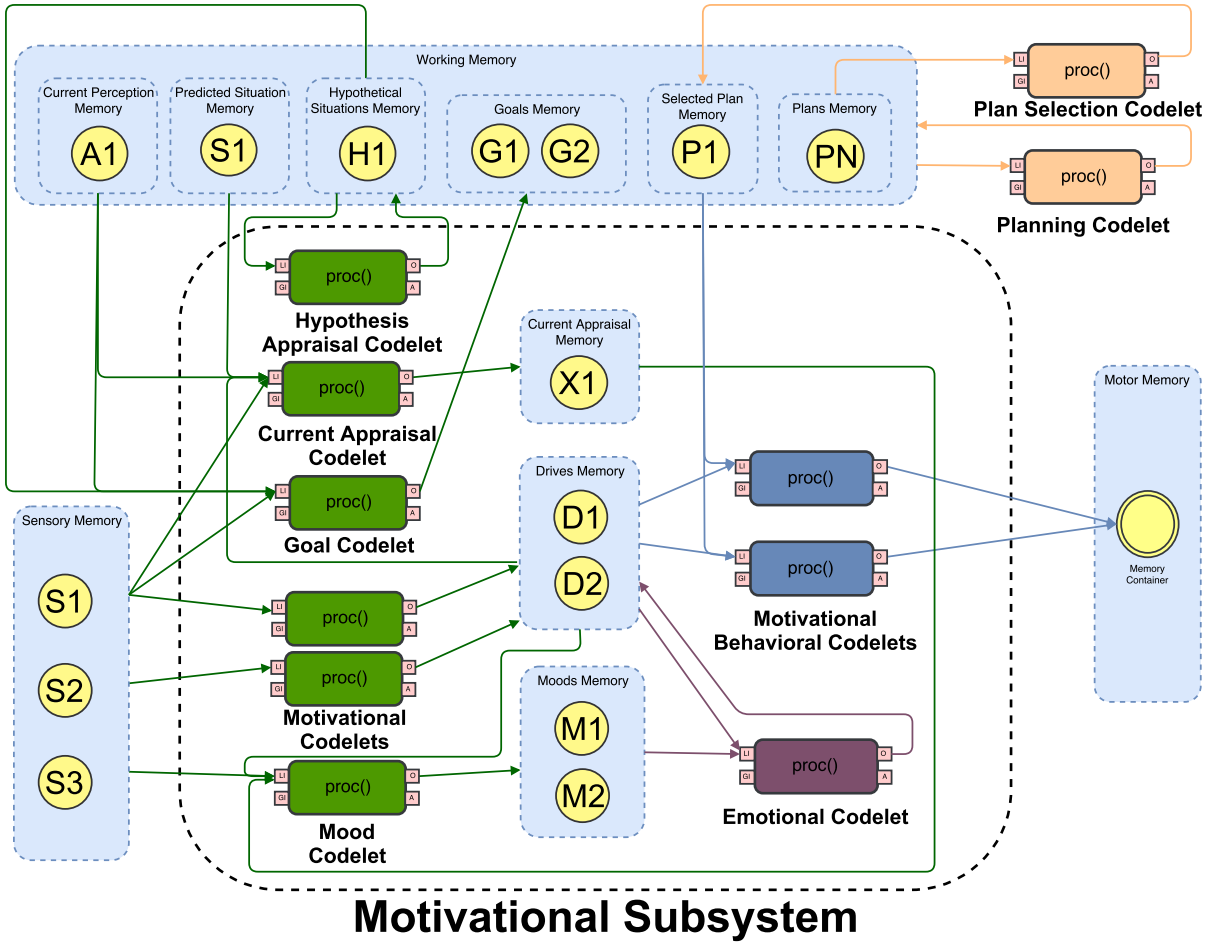

Figura 15 – Visão Geral do escopo do Subsistema Motivacional presente no CST.

Plano Selecionado. Normalmente, esta memória está sempre vazia e quando isso acontece, o mecanismo de subsunção prevalece, e tem o controle dos atuadores. Mas, no caso de uma situação excepcional, o sistema de planejamento gera um plano, o Codelet de Seleção de Planos seleciona um plano e manda para os Codelets Comportamentais do sistema. No caso de existir um plano na memória de plano selecionado, todos os Codelets Comportamentais tentarão satisfazer o plano. Nesse caso, cada Codelet Comportamental irá verificar se o plano é destinado a eles. Se sim, o codelet irá produzir uma saída atendendo o comando implícito no plano. Assim que o *goal* for atingido, o codelet de Seleção de Planos "marca" o plano como realizado e retira-o memória. Deste modo, a memória de plano selecionado que fica novamente vazia e o mecanismo de Subsunção retorna o controle dos codelets comportamentais.

Com a estrutura geral apresentada, é possível agora apresentarmos o controlador motivacional completo, junto com as experiências que imaginamos para verificar se o Subsistema Motivacional funciona como projetado e, consequentemente, avaliar seu desempenho.

3.4 O Desenvolvimento e Implementação do Subsistema Motivacional

O Subsistema Motivacional, como descrito nesse capítulo, foi desenvolvido e incorporado ao código-fonte do projeto CST. Como o CST é um projeto baseado em Java, o Subsistema Motivacional foi implementado também em Java. As classes foram incluídas em um novo pacote *br.unicamp.cst.motivational* e também no pacote *br.unicamp.cst.bindings.soar*. As classes incluídas no pacote motivacional são as seguintes: *Drive*, *Goal*, *Mood*, *Appraisal*, *MotivationalCodelet*, *GoalCodelet*, *MoodCodelet* e *AppraisalCodelet*. Já no pacote de planejamento foram incluída as classes: *Planning* e *Plan Selection Codelet*. A figura 16 ilustra uma visão dos pacotes do CST com detalhes dos conteúdos do pacote motivacional e de planejamento, utilizando o JSOAR.

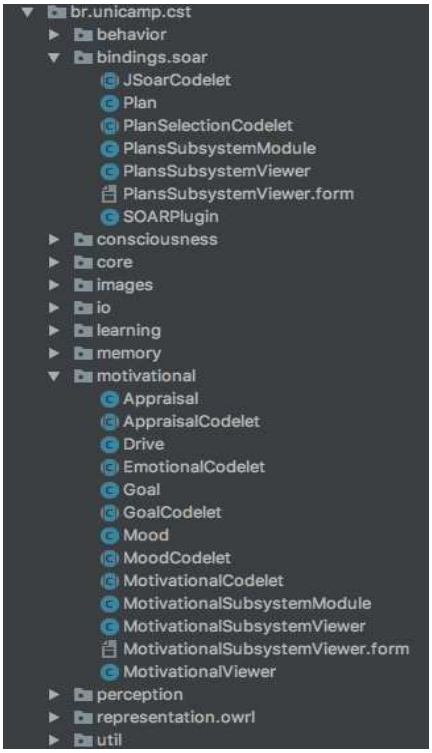

Figura 16 – Pacotes do projeto do CST.

As classes de pacotes são entidades (classes especializadas para representar objetos de memória) ou codelets. As classes Codelets são responsáveis por gerar e manter os conteúdos das entidades Drive, Goal, Mood, Appraisal e Plan. A classe JSOARPlugin é responsável por realizar a integração entre as classes do CST e o módulo de processamento de regras do JSOAR. Consequentemente, o JSoarCodelet é a classe que faz o processamento e tradução das entidades do CST para o núcleo de processamento do JSOAR. Além disso, algumas classes derivadas da classe *Codelet* são abstratas. Isso acontece porque é responsabilidade do desenvolvedor do Controlador ou de uma Arquitetura Cognitiva

personalizar métodos polimórficos das classes para construir seu comportamento final. Também vale ressaltar que foram desenvolvidos testes de unidade e integração de classes do Subsistema Motivacional, verificando o comportamento de cada classe separadamente e em conjunto com o simulador WS3D. A figura 17 mostra um diagrama UML com as classes do pacote, que são retratadas em amarelo e sua relação com outras classes CST em azul.

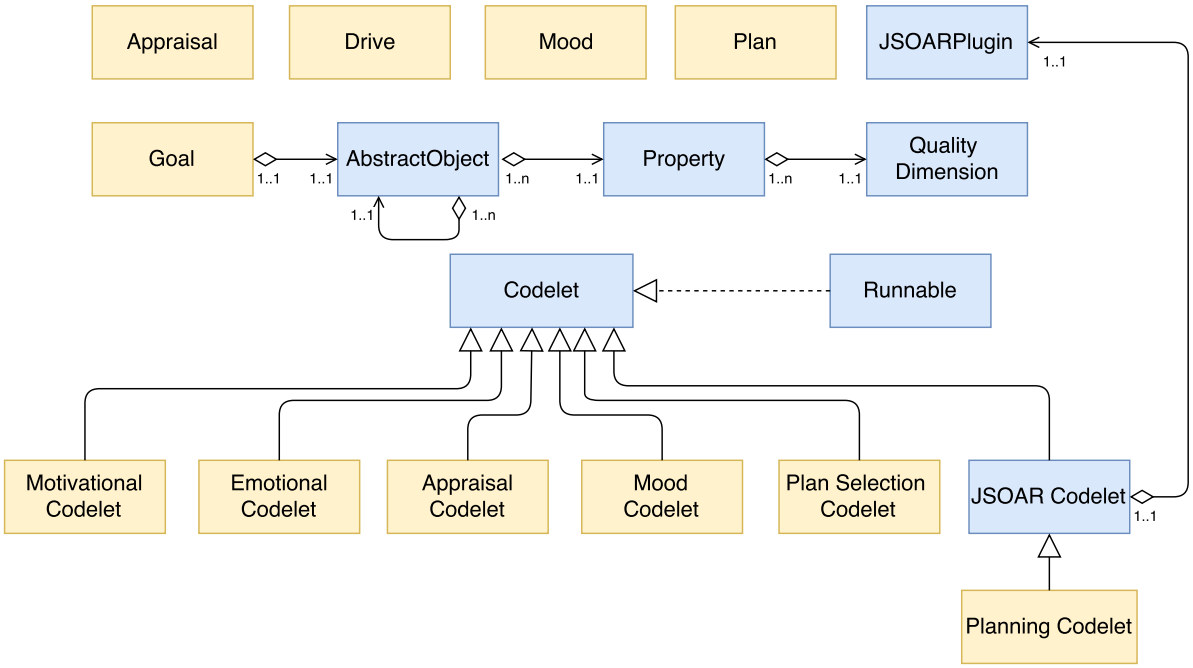

Figura 17 – Diagrama UML das classes presentes no Subsistema Motivacional.

4 Controlador de Agentes Inteligentes Baseado no Modelo Motivacional.

O modelo de subsistema motivacional apresentado no capítulo anterior é genérico, tendo sido implementado em Java e incluído dentre as classes disponíveis do CST. Neste capítulo, utilizamos esse modelo genérico em uma aplicação específica, o controle motivacional de uma criatura virtual em um ambiente virtual 3D, utilizado como teste para o desenvolvimento de nosso modelo.

4.1 A Criatura Artificial e seu Ambiente Virtual

Para testar e avaliar o controlador motivacional, utilizamos o *World Server3D Application*(WS3D). Este simulador é um ambiente virtual desenvolvido pelo nosso grupo de pesquisas na FEEC-UNICAMP, e utilizado em diversos outros trabalhos do grupo (CASTRO; GUDWIN, 2013; LUCENTINI, 2017). A figura 18 fornece uma captura de tela da interface WS3D. O WS3D permite a simulação de um ambiente virtual onde os usuários podem testar diferentes mentes de agentes para uma criatura artificial, onde um personagem pode possuir uma série de tarefas durante a simulação. A criatura "vive" em um mundo virtual 3D, onde pode encontrar comida (maçãs e nozes) e objetos de valor (joias de cores diferentes). Um agente pode perceber esses objetos e executar um conjunto de ações neles. No caso de joias, uma criatura pode capturá-las (colocar na sua bolsa) ou enterrá-las (ocultá-la no ambiente). No caso de nozes e maçãs, um agente pode capturá-las (colocar no saco), enterrá-las ou comê-las. Uma vez que um objeto aparece no ambiente, ele permanece lá até que seja colhido ou comido (no caso de ser um alimento). A diferença entre as maçãs e as nozes é que as maçãs se tornam podres depois de algum tempo (perdem sua capacidade de fornecer energia), mas as nozes não. Uma *Tabela de Trocas* (ou *leaflet*, conforme a terminologia utilizada na proposição original do experimento (CASTRO; GUDWIN, 2013)) fornece combinações de joias com determinadas cores que podem ser trocadas por pontos no assim chamado "Ponto de Troca" (*Delivery Spot*, na proposição original do experimento). O objetivo da criatura é maximizar seus pontos, mantendo seu equilíbrio energético. Para isso, a criatura precisa avaliar sua tabela de trocas e tentar escolher joias que maximizem o seu ganho de pontos, ao mesmo tempo que deve manter seu equilíbrio energético, evitando que suas reservas de energia caiam a zero. O WS3D permite que mais de uma criaturas sejam controladas no ambiente virtual. Portanto, de certa forma, uma criatura pode ou não interferir no desempenho da outra. Sendo assim, as criaturas podem enterrar maçãs, nozes ou joias, para evitar que

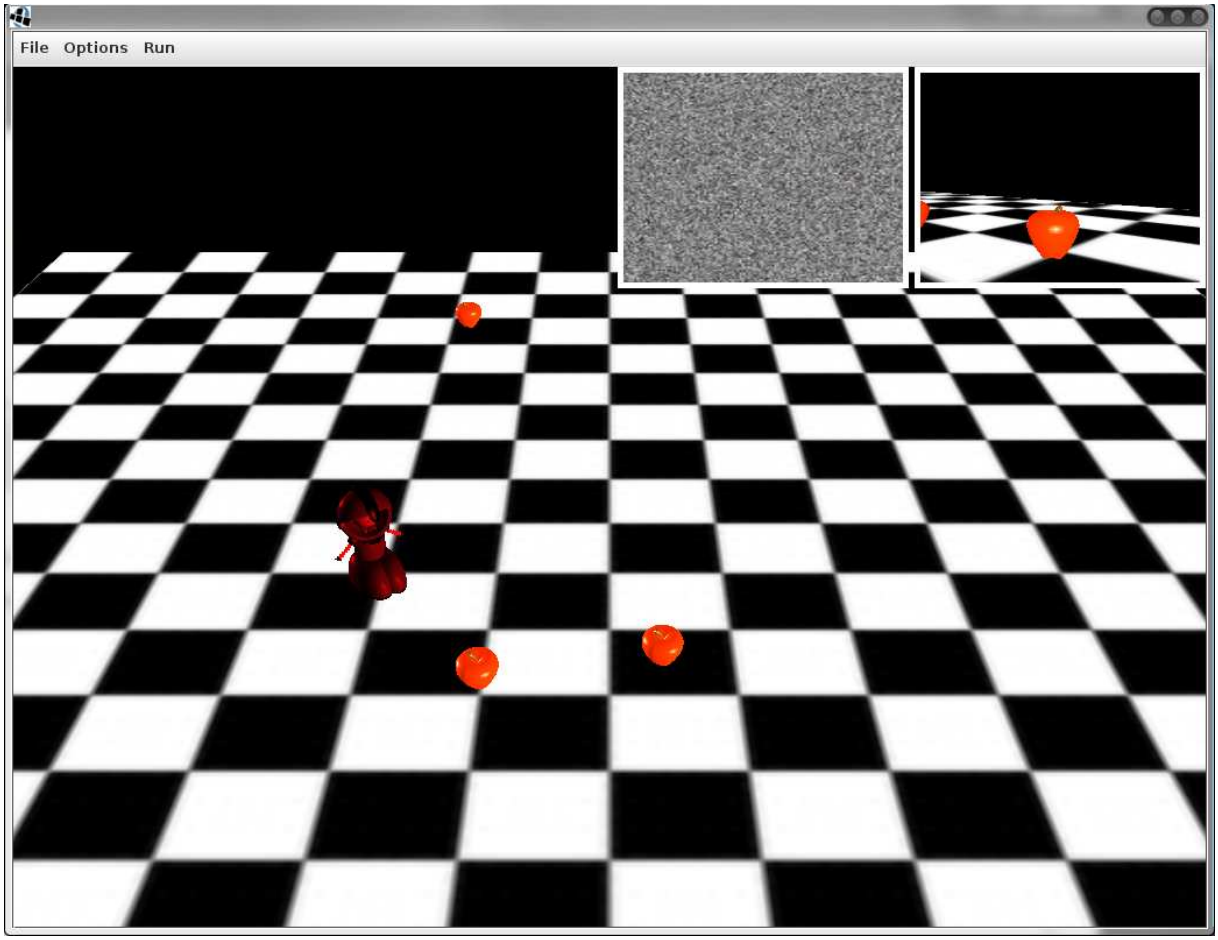

Figura 18 – A Aplicação World Server 3D (FRóES, 2017).

outras criaturas as escolham, caso estejam competindo no ambiente virtual. Conhecendo a posição exata onde foram enterrados, a criatura pode retornar mais tarde e encontrar o objeto primeiro e fazer uso dele, após desenterrá-lo. Esta poderia ser uma estratégia, caso sua mala esteja cheia. Se o nível de energia da criatura atingir o valor 0, a criatura morre.

O WS3D implementa a comunicação entre criaturas e suas mentes usando *sockets* TCP/IP. Assim, é possível executar o ambiente virtual em uma máquina e as mentes na mesma ou em outras máquinas, caso o esforço computacional necessário seja muito intenso.

4.2 Experimentos

Para verificar a eficiência e eficácia do controlador motivacional, foram desenvolvidas cinco aplicações, controlando agentes inteligentes no ambiente virtual WS3D. A primeira é uma aplicação de referência, que apresenta um comportamento puramente reativo utilizando somente a classes Codelet e*Memory Object*do CST. A segunda aplicação, implementa um controlador de agentes inteligentes utilizando somente a arquitetura cognitiva JSOAR (sem nosso controlador motivacional). Na terceira aplicação, implementa-

mos um controlador que faz uso da arquitetura cognitiva CLARION, também sem nosso controlador motivacional. É importante ressaltar que o CLARION também possui um subsistema motivacional. Desta forma, nosso intuito é podermos comparar nossa arquitetura com outra possuidora de sistema motivacional. Na quarta, utilizou-se a arquitetura cognitiva LIDA para o desenvolvimento do controlador (também sem o uso de nosso controlador motivacional). Todas essas arquiteturas foram desenvolvidas aqui para efeito de comparação, sendo que SOAR, CLARION e LIDA são arquiteturas cognitivas bastante diferentes entre si e muito populares na comunidade, sendo bastante representativas da diversidade de arquiteturas cognitivas existentes (LUCENTINI, 2017). E por fim, a quinta aplicação é justamente a implementação do controlador motivacional desenvolvido neste trabalho. Outro fator que vale a pena mencionar é que as aplicações foram executadas individualmente, ou seja, em cada simulação somente existe uma única criatura, que é controlada pela respectiva aplicação sendo executada no momento.

Todas as aplicações se relacionam com o mesmo cenário, que consiste no jogo em que uma única criatura deve capturar um conjunto de joias prescrito em sua tabela de trocas, trocá-las por pontos no "Ponto de Troca" para obter o máximo de pontos possível e ao mesmo tempo manter o seu nível de energia acima de zero. No início de cada simulação, o ambiente gera três combinações de troca para cada criatura, e uma quantidade fixa de joias (de diferentes tipos) é distribuída aleatoriamente no ambiente. As joias têm um conjunto finito de cores: azul, magenta, branco, vermelho, verde e amarelo. O número de joias em cada combinação é 3, sendo que 3 trocas são disponibilizadas por tabela, de tal forma que o número de joias de uma tabela de trocas completa é igual a 9. O nível de energia da criatura pode variar de um máximo de 1000 a um mínimo de 0, quando a criatura morre. Este nível de energia diminui por etapas de 50 ao longo do tempo de simulação. Tendo isso em mente, os aplicativos são capazes de realizar os seguintes comportamentos:

- Movimentação Aleatória.
- Rotacionar a criatura em busca de objetos.
- Evitar Colisões com obstáculos.
- Capturar Joias desejadas.
- Comer Alimentos (Maçãs e Nozes).
- Enterrar e Desenterrar Joias e Comidas.
- Ir em direção à Joia mais próxima.
- Ir em direção à Comida mais próxima.

Embora esses comportamentos estejam presentes nas aplicações, sua implementação é diferente em cada uma delas, pois utilizam tecnologias e arquiteturas cognitivas

diferentes. Para maiores detalhes sobre as implementações proposta neste trabalho, o capítulo 7 fornece os links dos repositórios onde estão armazenadas todos as aplicações desenvolvidas neste trabalho.

O tempo estipulado de simulação para todos os controladores é de 10 minutos. Para cada experimento, executamos ao menos dez simulações para cada aplicação controladora, implicando em dez cenários diferentes em termos de joias, alimentos e obstáculos. Além disso, são geradas aleatoriamente três novas joias e uma comida no ambiente a cada 1 minuto de simulação. Esses novos objetos são essenciais para a execução do experimento, pois a criatura necessariamente precisa de novas joias para que a mesma consiga completar todas as combinações de sua tabela de trocas, e consequentemente, obter a máxima pontuação possível, e de comida para que o agente não fique completamente sem energia durante o processo de simulação. Apesar de novos objetos serem gerados durante a simulação, o que de certa maneira dá suporte à criatura em manter seus pontos e seu nível energético elevados, cabe ao controlador da criatura artificial garantir que a máxima pontuação e a sobrevivência do agente sejam atingidas, independentemente de uma abundância de objetos no ambiente. Assim como no início da simulação, os novos objetos são gerados em posições aleatórias no ambiente. Não somente as posições como os tipos de objetos também são definidos aleatoriamente.

4.3 Detalhes do Controlador Motivacional

Nesta seção descrevemos de maneira detalhada, cada um dos codelets utilizados em nosso controlador motivacional. Nas sub-seções seguintes, especificamos cada codelet individualmente e sua respectiva função dentro do agente inteligente. Para apresentar uma visão geral do controlador motivacional, a Figura 19 demonstra um diagrama que apresenta todos os codelets existentes dentro da arquitetura do controlador motivacional, de acordo com o modelo proposto por este trabalho.

4.3.1 Codelets Sensoriais

Para que a criatura possa identificar objetos presente no ambiente virtual, sensorear a energia gasta durante a simulação e verificar quais joias já foram coletadas, o controlador faz uso de Codelets Sensoriais. Dentre os Codelets Sensoriais existentes no controlador motivacional estão:

- Codelet de Visão (*Vision Codelet*).
- Codelet Proprioceptivo (*Inner Sense Codelet*).

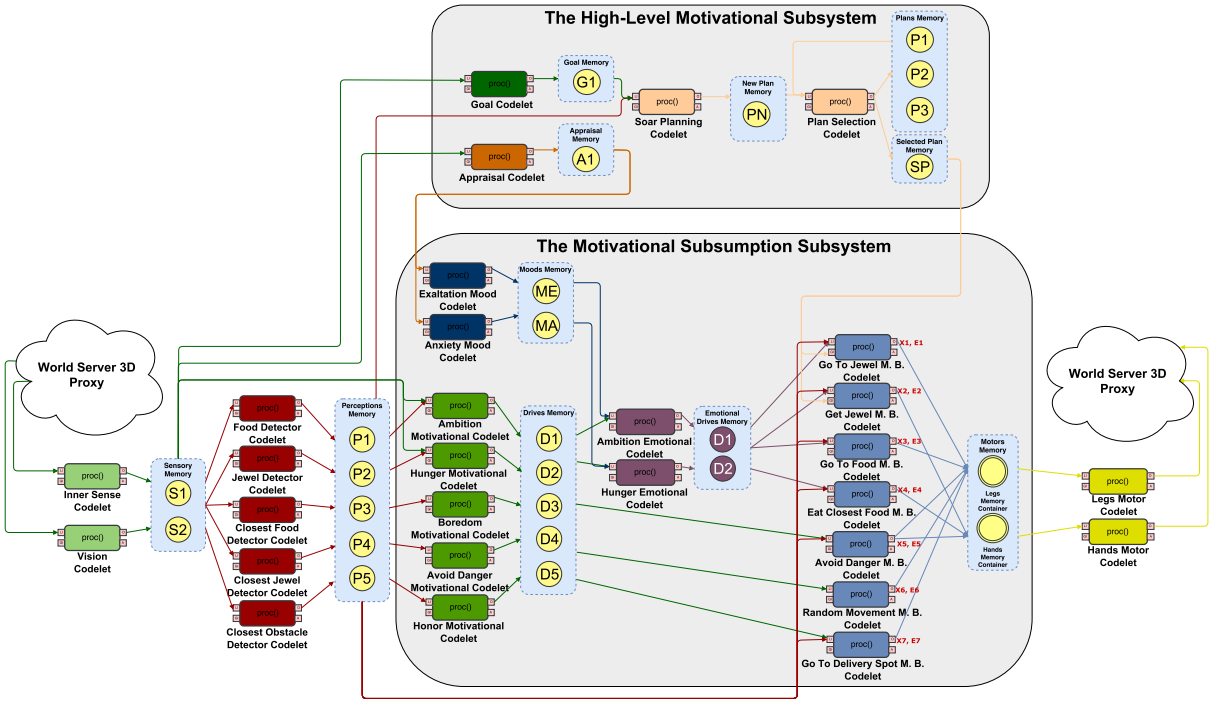

Figura 19 – Arquitetura Cognitiva para o Controle do Agente - Instância-exemplo do Subsistema Motivacional desenvolvido neste trabalho.

Esses Codelets são responsáveis por capturar e processar dados sensoriais da criatura e transformá-los em informações relevantes para outras estruturas da mente do agente inteligente. Os Codelets Sensoriais são fundamentais para a arquitetura motivacional como um todo, pois o modelo de controlador motivacional baseia-se completamente nas informações geradas por esses Codelets. O controlador faz uso da biblioteca "WS3DProxy" para recuperar os dados de simulação. O *World Server 3D Proxy* (*WS3DProxy*) é ao mesmo tempo um *proxy* de recuperação de dados do simulador e também um mensageiro de comandos que controla a simulação corrente. Com ele, o controlador pode executar diferentes comportamentos nos atuadores da criatura artificial, controlar todos os objetos da simulação podendo gerar e/ou destruir objetos no ambiente. Da mesma forma que o CST e o World Server 3D, a biblioteca *WS3DProxy* também foi desenvolvida em linguagem de programação Java por diferentes pesquisadores da Universidade Estadual de Campinas. A biblioteca utiliza-se do protocolo TCP/IP para troca de mensagens entre o controlador e o simulador.

O Codelet de Visão (*Vision Codelet*) é responsável por identificar todos os objetos dentro do campo de visão da criatura - objetos como: Obstáculos, Maçãs, Nozes e Jóias. O Codelet de Visão, apesar do nome, não realiza nenhuma operação de visão de fato. Ele utiliza um recurso do *WS3DProxy* que, dada a posição da criatura e seu ângulo de visão, retorna uma lista com todos os objetos visualizados pela criatura, computando-se as devidas oclusões no campo de visão. Podemos entender portanto o Codelet de Visão

como uma simplificação do processo de visão, uma vez que diversas etapas de visão, como reconhecimento de padrões, segmentação e modelagem de objetos é realizada de maneira implícita, uma vez que nosso intuito não é reproduzir completamente o processo de visão, desde a captura de pixels até a modelagem do ambiente em termos de objetos, mas saber o que fazer uma vez que esses objetos estejam disponíveis. Uma vez que os dados visuais são coletados, o Codelet de Visão armazena as informações dos objetos da cena dentro da Memória Sensorial (*Sensory Memory*). Já a principal responsabilidade do Codelet Proprioceptivo é coletar e processar dados referente ao estado interno da criatura, ou seja, prover informações como: Energia da criatura, Tabela de Trocas, Número de Pontos Adquiridos, Porcentagem de Joias Capturadas, Posição do Agente no Ambiente e Angulo do Agente com relação a sua posição atual no ambiente. Embora sua função seja diferente do Codelet de Visão, o Codelet Proprioceptivo também armazena as informações sobre a criatura artificial na Memória Sensorial. A Figura 20 demonstra a interação entre o "WS3DProxy" e os Codelets Sensoriais.

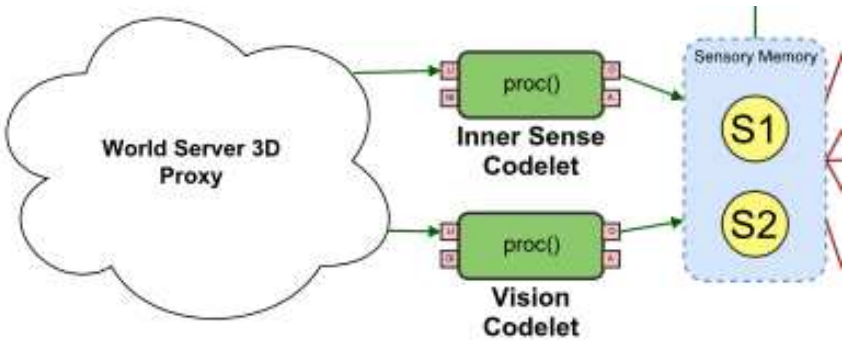

Figura 20 – Interação entre Codelets Sensoriais e o World Server 3D Proxy.

4.3.2 Codelets Perceptuais

A partir das memórias sensoriais geradas pelos Codelets Sensoriais, os Codelets Perceptuais realizam filtros sobre dados existentes nas memórias sensoriais, mais especificamente na memória sensorial fornecida pelo Codelet de Visão. A lista abaixo apresenta os Codelets Perceptuais existentes na arquitetura do controlador motivacional.

- Codelet Detector de Jóia (*Jewel Detector Codelet*).
- Codelet Detector de Comida (*Food Detector Codelet*).
- Codelet Detector de Proximidade de Jóia (*Closest Jewel Detector Codelet*).
- Codelet Detector de Proximidade de Comida (*Closest Food Detector Codelet*).
- Codelet Detector de Proximidade de Obstaculos (*Closest Obstacle Detector Codelet*).

Cada Codelet Perceptual tem a finalidade de aplicar um tipo de filtro específico nas informações sensoriais, separando os objetos que estão dentro do campo visual da criatura em perceptos, de acordo com as categorias dos objetos. Por exemplo, o *Food Detector Codelet* seleciona os objetos referentes à categoria dos alimentos, que neste caso, podem ser Maçãs ou Nozes. Em contrapartida, o *Jewel Detector Codelet* seleciona os objetos que pertencem à categoria das Joias. Vale a pena ressaltar que ambos os Codelets selecionam os seus respectivos objetos independentes de qualquer parâmetro, a não ser a categoria a que o objeto pertence. Por outro lado, os Codelets como *Closest Food Detector*, *Closest Jewel Detector* e *Closest Obstacle Detector* levam em consideração o critério da distância do agente com relação aos objetos do ambiente. Isso acontece porque a finalidade desses Codelets é detectar possíveis colisões que a criatura teria com os objetos existentes no cenário. Por exemplo, *Closest Food Detector* e *Closest Jewel Detector* identificam colisões do agente com Joias ou Alimentos a uma distância de "65" pixels. Por outro lado, o *Closest Obstacle Detector Codelet* pode detectar colisões com obstáculos (Paredes ou Ponto de Troca) a uma distância de 50 pixels. Esta diferença no valor dos parâmetros de distância entre esses Codelets, se dá porque a capacidade do agente em detectar uma colisão com um obstáculo acaba sendo menos frequente do que colisões com joias ou alimentos. Isso acontece porque em grande parte do tempo de simulação, há mais joias e alimentos do que obstáculos. Sendo assim, o número de objetos que o Codelet necessita processar interfere diretamente no tempo de processamento que ele leva para perceber uma colisão. Se os *Codelets Closest Jewel Detector* e *Closest Food Detector* apresentam muito objetos para processar, eles levam mais tempo para calcular as distâncias dos objetos e, consequentemente, mais tempo para ordená-los em ordem crescente, e somente depois verificar se o objeto está próximo do limiar de 65 pixels ou não. Outro fator onde a diferença das distâncias interfere no comportamento do agente com objetos do ambiente, é justamente quando existem um ou mais objetos desejáveis muito próximos a obstáculos. A proximidade desses objetos a obstáculos pode fazer com que a criatura detecte um obstáculo antes mesmo de detectar um objeto plausível de ser capturado. Esse tipo de situação pode fazer com que a criatura desvie de um obstáculo sem conseguir coletar um objeto desejável. Portanto, configurar o limiar de detecção de jóias e comidas com valor maior do que a distância de detecção de obstáculos, faz com que a criatura detecte os objetos antes mesmo de colidir com um obstáculo, e consequentemente, capture-os. Os parâmetros de distância de cada Codelet foram ajustados e otimizados várias vezes durante a fase de testes do aplicativo, justamente para evitar esse tipo de problema nas situações descritas anteriormente. A Figura 21 demonstra a interação entre os Codelets Sensoriais e os Perceptuais. A Tabela 1 prescreve as funções de cada Codelet Perceptual juntamente com os seus respectivos parâmetros de distância e categoria de objetos.

Tabela 1 – Parâmetros dos Codelets Perceptuais.

Codelet	Função	Categoria do Objeto	Distância
Food Detector	Detectar Alimentos.	Maçãs e Nozes	-
Jewel Detector	Detectar Joias.	Joias	-
Closest Food Detector	Detectar Colisão com Alimentos.	Maçãs e Nozes	65 pixels
Closest Jewel Detector	Detectar Colisão com Joias.	Joias	65 pixels
Closest Obstacle Detector	Detectar Colisão com Obstáculos.	Paredes ou Ponto de Troca	50 pixels

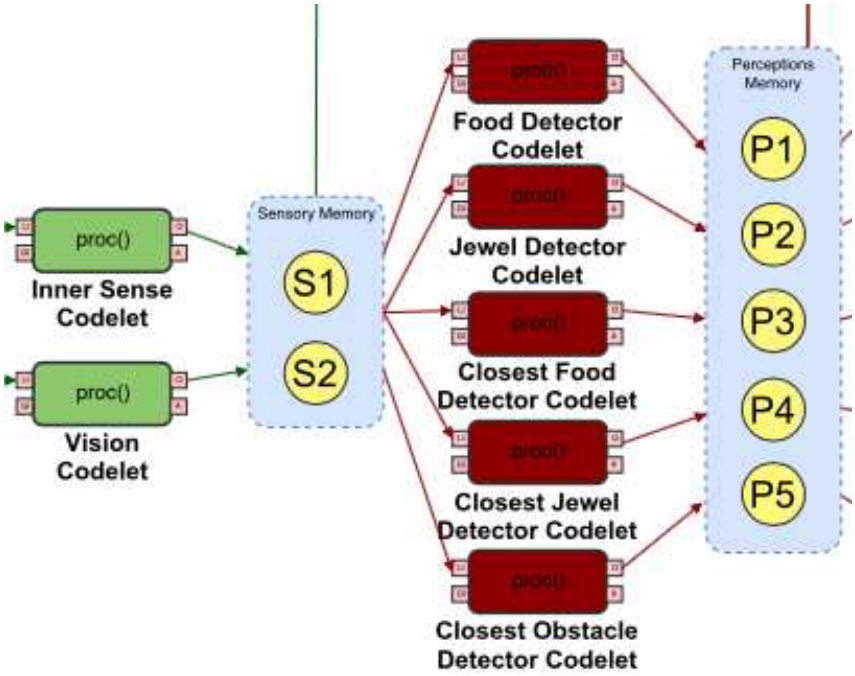

Figura 21 – Codelets Perceptuais gerando perceptos na memória perceptual a partir da memória sensorial.

4.3.3 Codelets Motivacionais

Seguindo o modelo de subsistema motivacional proposto no capítulo 3, foram desenvolvidos os seguintes Codelets Motivacionais:

- *Hunger Motivational Codelet* - Gera e mantém um drive que simboliza a Fome da criatura.
- *Ambition Motivational Codelet* - Gera e mantém um drive que representa a Ambição da criatura.

- *Honor Motivational Codelet* - Gera e mantém um drive que representa a Honra da criatura.
- *Boredoom Motivational Codelet* - Gera e mantém um drive que simboliza o Tédio da criatura.
- *Avoid Danger Motivational Codelet* - Gera e mantém um drive que representa o senso de segurança do agente, ou até mesmo, um drive simbolizando a prevenção de danos da criatura.

Esse codelets são responsáveis por manter e gerar drives para diferentes comportamentos específicos da criatura. Por exemplo, o drive de Fome estimula a criatura a ir à procura de alimentos no ambiente. Este drive, é baseado no nível de energia da criatura. Portanto, quando a energia da criatura é baixa, o nível de ativação do drive de Fome será alto. Conforme a criatura encontra alimentos no ambiente virtual e os captura, o nível de ativação do drive de Fome cai gradativamente. Em contrapartida, o drive de Ambição faz com que a criatura artificial explore o ambiente em busca de joias para completar todas as combinações de sua tabela de trocas. Este drive, é estimulado pela porcentagem total de joias obtidas pelo agente durante o tempo de simulação e também pelas joias presentes no ambiente que estão dentro de sua percepção visual. Diferentemente do drive de Fome, o drive de Ambição tende sempre a estar alto, pois, conforme a criatura obtém novas joias, maior será sua Ambição. Este tipo de comportamento reproduz e se aproxima da Ambição humana/animal real, em que, quanto mais um indivíduo obtém algo que lhe satisfaz, mais ambicioso ele fica. Entretanto, do ponto de vista do comportamento humano/animal, existe uma situação inversa, onde o indivíduo fica ambicioso mesmo quando alcança um objetivo específico. Nosso modelo de controlador motivacional contempla os dois casos. Quando a criatura completa uma ou mais das combinações de sua tabela de trocas, a criatura vai até o Ponto de Troca e efetua a troca das joias coletadas por pontos. Quando esta situação ocorre, a porcentagem total de joias coletadas diminui drasticamente, e consequentemente, isso afeta diretamente o drive de Ambição diminuindo o seu nível de ativação. O controlador motivacional também implementa o drive de Honra. Ele é responsável por detectar se alguma combinação da tabela de trocas do agente foi completada durante a captura de joias. O comportamento prescrito é a movimentação da criatura até o Ponto de Troca, e consequentemente, a efetivação da troca das joias por pontos. Caso o cenário anteriormente descrito seja verdadeiro, a ativação do drive fica em nível alto. Por outro lado, se este cenário é falso, o nível de ativação se mantém baixo. Outro drive presente no controlador é o drive de Tédio. A ativação deste drive aumenta conforme a criatura fica em uma mesma posição durante muito tempo (20 segundos). Geralmente, isso ocorre quando a criatura não encontra os objetos que deseja, e se mantém em uma mesma posição rotacionando em seu próprio eixo. Este drive encoraja a criatura a buscar

um comportamento aleatório, direcionando a criatura a diferentes posições aleatórias, e consequentemente, possibilitando que o agente possa detectar objetos que não poderiam ser detectados na posição anterior devido a obstáculos (Paredes ou Ponto de Troca) que poderiam estar interferindo em seu campo visual. Finalmente, o drive de Prevenção de Danos tem a finalidade de incitar a criatura para que a mesma não colida com objetos não coletáveis do ambiente. Se de alguma maneira, durante a movimentação o agente arrisca-se a colidir com um objeto indesejável (obstáculo, comida ou joia), a ativação do drive aumenta significativamente e o encoraja a tomar uma decisão de acordo com o tipo de objetos que está em sua frente. Por exemplo, se a criatura está em risco de colidir com uma joia que não está presente nas combinações de sua tabela de trocas ou um alimento (maçã ou noz) que está em seu caminho, então a criatura os enterra no ambiente. No caso das comidas enterradas, o agente tem a capacidade de guardar as posições de onde elas foram enterradas, e se necessário, posteriormente desenterrá-las para comê-las. Já no caso das joias indesejadas, simplesmente a criatura as enterra para não colidir com elas. Do ponto de vista de obstáculos, o agente simplesmente os evita mudando sua trajetória até que os mesmos não atrapalhem o seu caminho, ou até mesmo, escolhendo outra possível joia existente no ambiente e que ainda não foi adquirida.

Os Codelets Motivacionais baseiam as ativações dos drives que geram nos percep-tos armazenados nas memórias perceptuais. Quando o agente percebe uma maçã ou noz dentro do seu campo visual, a ativação do drive de Fome cresce em "20%". Além disso, a ativação do drive de Fome somente será diferente de zero, se a criatura visualizar algum alimento no ambiente, ou seja, quando a criatura apresentar um estímulo visual oriundo da percepção de maçãs e/ou nozes. Por exemplo, em uma situação hipotética, é possível haver casos em que não existam alimentos no ambiente virtual e também a criatura esteja com um nível alto de fome. Neste caso, a abordagem anteriormente descrita permite que o agente evite a busca excessiva de comida no ambiente (uma vez que não há comidas no ambiente), e sendo assim, possibilita que a criatura realize outras tarefas importantes, como a busca e captura de joias.

O drive de Ambição e Prevenção de Danos também apresenta reação a estímulos. Para o drive de ambição isso acontece quando a criatura percebe uma joia que pertence a uma das combinações de sua tabela de trocas. Neste caso, a ativação do drive aumenta em "20%". Já para o drive de Prevenção de Danos, se o agente visualizar um obstáculo, sua ativação cresce em "0.05".

Vale a pena ressaltar que alguns dos Codelets Motivacionais como Fome e Ambição se baseiam tanto nas memórias perceptuais (a partir de estímulos visuais) quanto na memória sensorial gerada a partir do Codelet Proprioceptivo. Por exemplo, o drive de Ambição baseia-se na Porcentagem de Joias Coletadas da Tabela de Trocas da criatura. Em contrapartida, o drive de Fome baseia-se no Nível Energético do agente.

A tabela 2, descreve quais foram os parâmetros utilizados na composição dos drives e também a heurística empregada no cálculo de ativação dos mesmos. A Figura 22 mostra a interação entre sensores de percepção e os Codelets motivacionais.

Tabela 2 – Parâmetros dos Drives gerados pelos Codelets Motivacionais.

Drive	Prioridade	Urgência	Formula
Fome	0.25	0.8	$A = \begin{cases} 0.95 * \max(F_d, F_d * (1 + F_s)) & if F_s > 0 \\ 0.0 & if F_s <= 0 \end{cases}$
Ambição	0.2	0.9	$A = 0.95 * \max(A_d, A_d * (1 + J_s))$
Tédio	0.35	0.65	$A = T_{sp}/20000$
Honra	0.3	1.0	$A = \begin{cases} 1.0 & if LF_i = 1.0 \\ 0.0 & if LF_i < 1.0 \end{cases}$
Prevenção de Danos	0.4	0.7	$A = \begin{cases} 0.7 + B_s & if B_d <= 40 \\ B_s & if B_d > 40 \end{cases}$

onde A é ativação do drive, F_d é o deficit de comida (ou fome), F_s é o estímulo de comida (maçãs ou nozes do ambiente), A_d é o deficit de ambição, J_s é o estímulo de joias, T_{sp} é o tempo da criatura na mesma posição (em milissegundos), LF_i é a porcentagem da i-ésima combinação da tabela de trocas da criatura, B_s é o estímulo do obstáculo e B_d é a distância do obstáculo até o agente.

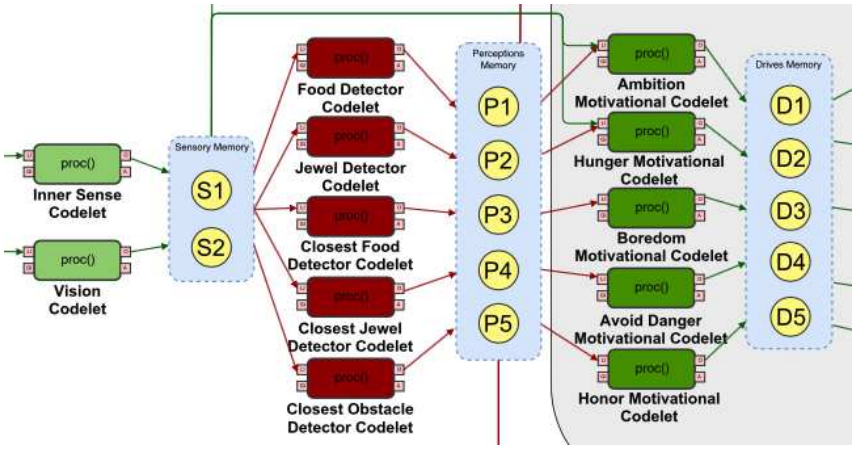

Figura 22 – Interação entre Codelets Sensoriais, Perceptuais e os Motivacionais.

4.3.4 Codelets de Apreciação

De acordo com Lazarus (1991), umas das características fundamentais para o surgimento das emoções, é a capacidade do ser humano em avaliar eventos que afetam a si próprio ou outros indivíduos. Como descrito nas seções 2.1 e 3.2, Lazarus (1991) descreve o raciocínio avaliativo em duas vertentes plausíveis, representadas por: Avaliações

Primárias e Secundárias. O modelo de subsistema motivacional proposto por esse trabalho compreende ambos os tipos de avaliações. Entretanto, no controlador motivacional somente utilizamos a avaliação primária, cujo objetivo é realizar a avaliação do estado interno do agente, julgando suas condições fisiológicas e sociais. Sendo assim, com este tipo de julgamento é possível obter três níveis diferentes de avaliação do estado interno do agente, que neste caso, podem ser descritos como: Ruim, Normal ou Bom.

A avaliação primária é gerada e mantida pelos Codelets de Apreciação (ou *Appraisal Codelets*). Para tal função, o codelet utiliza-se da estrutura de dados "Appraisal" (classe) para armazenar as informações do estado interno da criatura. Esta classe foi implementada durante o desenvolvimento do modelo de subsistema motivacional e está presente nos pacotes do CST. A classe é capaz de armazenar a avaliação corrente através de uma variável numérica (entre 0 e 1) denominada "Evaluation" - Eval (avaliação) e também através da variável "Current Appraisal State" - CAS (ou Estado Avaliativo Corrente), onde sua finalidade é classificar (ou descrever) a variável *Eval*, tendo em vista possíveis estados que o agente pode alcançar (Ruim, Normal ou Bom). Por exemplo, é possível que durante uma simulação a criatura possa estar com a seguinte situação, onde a variável *Eval* do objeto "Appraisal" apresente o valor 0.8 e o valor de CAS classificado como "Bom". Neste caso, o valor 0.8 é classificado como *Bom* pela situação corrente do agente, e consequentemente, esta avaliação pode fazer como que a criatura seja estimulada a ter uma emoção que incite comportamentos que satisfaçam as necessidade internas do agente cognitivo. Esse tipo de abordagem utilizando a variável CAS, permite que o codelet possa dar uma significância para o estado corrente da criatura, além de somente avaliar o estado interno da criatura através de variável numérica. Além disso, não necessariamente as emoções estarão atuando o tempo todo na mente da criatura, mas as emoções podem atuar quando a avaliação estiver em níveis denominados "críticos". No caso do controlador motivacional isso acontece quando a classificação da variável CAS está em nível "Ruim" ou "Bom". Por outro lado, quando a avaliação da criatura é classificada como "Normal", as emoções não são necessárias para afetar diretamente o comportamento da criatura, já que o estado atual não é crítico. Neste caso, somente os drives (com suas ativações padrão) são suficientes para orientar o comportamento da criatura para equacionar suas necessidades internas.

Ambas variáveis são importante na determinação do Humor do agente inteligente, pois os Codelets de Humor baseiam-se diretamente no objeto Appraisal. A instância da classe Appraisal é armazenada em um *Memory Object*, e em seguida, armazenada nas memórias de apreciação (ou *Appraisal Memory*). Para determinar as variáveis de um objeto Appraisal, o Codelet de Apreciação faz uso das informações sensoriais, como memória de entrada do codelet. A Figura 23 demonstra a interação entre o codelet de Apreciação e as memórias sensoriais.

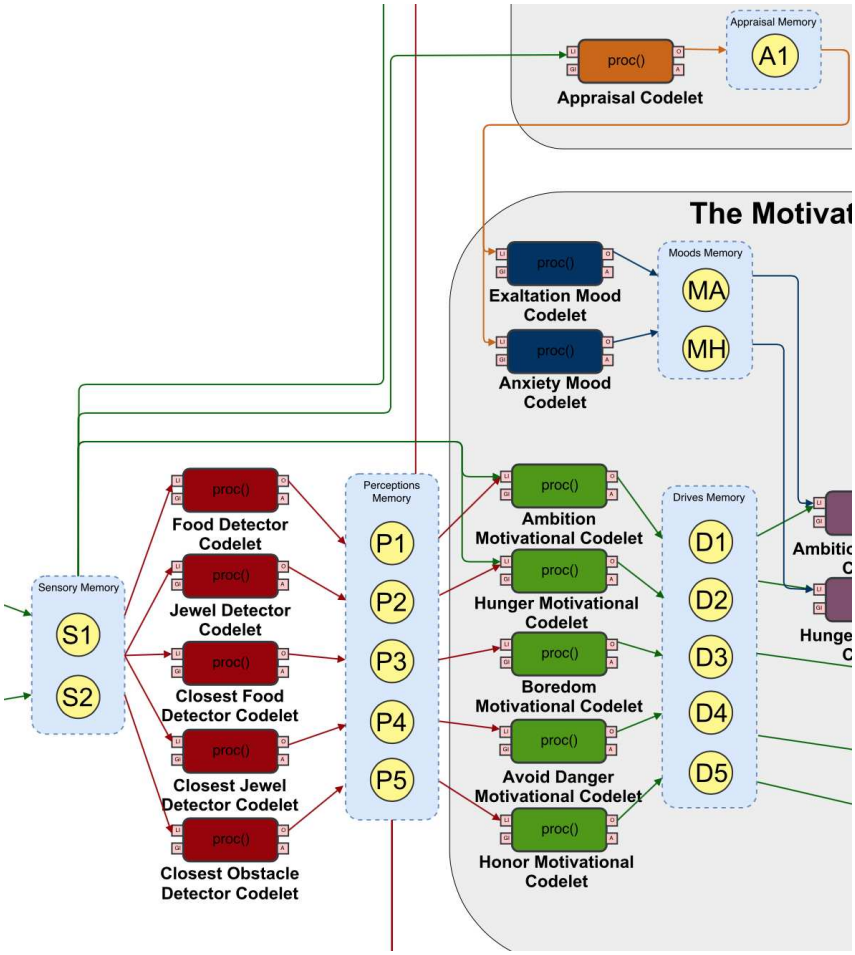

Figura 23 – Codelet de Apreciação realizando a avaliação primária do agente inteligente a partir das informações das memórias sensoriais.

O Codelet de Apreciação gera a avaliação primária do agente cognitivo implementando um modelo Fuzzy de Mamdami para determinar as variáveis *Eval* e *CAS*. Para isso, o Codelet utiliza-se de informações da memória sensorial, neste caso, diversas informações da criatura como: Porcentagem de Joias Coletadas da Tabela de Trocas e Nível Energético da Criatura. Estas memórias sensoriais são fundamentais, pois o Codelet faz uso das mesmas para estabelecer as funções de pertinência e as regras do sistema fuzzy. O objetivo deste trabalho não é discutir sobre sistemas fuzzy, tampouco avaliar os modelos existentes. Todavia, informações sobre parâmetros como "Métodos de Defuzzificação", "Conjunção" e "Implicação" (T-Normas), "Disjunção" e "Agregação" (S-Normas), "Regras Fuzzy", "Variáveis Linguísticas" e "Funções de Pertinência", são essenciais para a compreensão do modelo, e é claro, para o entendimento do funcionamento do Codelet de Apreciação. A Tabela 3 demonstra os parâmetros utilizados no sistema fuzzy de Mamdami implementado.

Um fator importante para todo sistema fuzzy é a questão da granularidade das funções de pertinência. O processo de "granularização" tem por objetivo estabelecer o

conjunto de variáveis linguísticas do sistema (juntamente com seus estados possíveis), a função de pertinência, a variável da saída do sistema, e também as regras de ativação fuzzy. No contexto do controlador motivacional, a variável "Energy" (que representa a energia da criatura) foi granularizada em três estados possíveis: "Hungry", "Normal" e "Satisfied". O estado "Hungry" simboliza a situação quando a criatura está com baixa energia, ou de certa forma "Faminta". Já o estado "Normal", pode ser entendido como um estado de homeostase energético do agente inteligente. E por fim, o estado "Satisfied" descreve a circunstância em que a criatura detém energia além da necessária para executar ações no ambiente virtual. O mesmo paradigma acontece com a variável "Leaflet Percentage". Porém, seu comportamento é logicamente diferente. O estado "Ambicious" descreve uma possível situação em que a criatura ainda necessita coletar joias, pois até o presente momento de simulação, o agente não conseguiu coletar uma quantidade significativa de joias para satisfazer as combinações de sua tabela de trocas. O estado "Normal" pode ser descrito da mesma forma que com a variável "Energy", ou seja, o estado "Normal" simboliza um estado homeostático no que diz respeito à ambição do agente pela prospecção e coleta de joias. O estado "Satisfied" descreve a ocasião quando o agente cognitivo está próximo de satisfazer sua ambição momentânea, através do término da coleta de joias necessárias para completar sua tabela de trocas. A variável de saída CAS foi fundamentada em três estados avaliativos, que podem ser descritas como: "Bad", "Normal" e "Good". Após testes realizados durante a concepção do controlador, estes estados mostraram-se suficientes para certificar se o agente está em estado crítico ou não. Como descrito anteriormente, somente em estados críticos as emoções podem incentivar a mudança de comportamento da criatura. Na seção 4.3.5, é possível verificar como os estados críticos de avaliação primária são fatores decisivos para geração de Humor na criatura e também para a definição das distorções cognitivas nos drives de Ambição e Fome. A Figura 25 demonstra as funções de pertinência e o conjunto de variáveis linguísticas em uso pelo sistema.

Baseado nas funções de pertinência e nas variáveis linguísticas, o Codelet de Apreciação utiliza-se do método de defuzzificação (centroid) da função de pertinência de saída do sistema, para determinar o valor da variável $Eval$. Já no caso da variável CAS, o seu valor é determinado conforme a saída da regra que apresenta a maior ativação momentânea. Neste contexto, o modelo implementa seis tipos diferentes de regras fuzzy para avaliar possíveis situações que a criatura pode enfrentar durante o processo de simulação. O algoritmo da Figura 24, prescreve as regras utilizadas pelo sistema.

```
IF Energy is HUNGRY THEN CAS is STATE_BAD
IF Energy is NORMAL and Leaflet Percentage is AMBITIOUS THEN CAS is STATE_GOOD
IF Energy is NORMAL and Leaflet Percentage is NORMAL THEN CAS is STATE_NORMAL
IF Energy is NORMAL and Leaflet Percentage is SATISFIED THEN CAS is STATE_NORMAL
IF Energy is SATISFIED THEN CAS is STATE_GOOD
IF Energy is SATISFIED and Leaflet is SATISFIED THEN CAS is STATE_NORMAL
```

Figura 24 – Regras Fuzzy Implentadas no Modelo de Mamdami presente no Codelet de Avaliação.

A primeira regra define a situação em que a criatura está com o nível energético "Bad". Em particular, este caso não depende da variável "Leaflet Percentage", pois nesta circunstância, independentemente da criatura estar "Ambiciosa" ou não, o agente necessita priorizar o seu "instinto de sobrevivência" e manter-se vivo para que a mesma consiga a posteriori obter o máximo de joias possível. Por esse motivo, o estado sugerido por essa regra é definido como "Bad". A segunda, terceira e a quarta regras, descrevem as situações em que a criatura apresenta nível energético "Normal". Em todas elas, a variável "Leaflet Percentage" participa na definição da avaliação primária. Por exemplo, na segunda regra, quando "Leaflet Percentage" é "Ambitious" a avaliação selecionada é "Good". Já na terceira e quarta regra, quando o "Leaflet Percentage" é definido como "Normal" ou "Satisfied", CAS é definida como "Normal". A quinta e sexta regras tratam o caso em que a criatura está com o estado energético configurado como "Satisfied". Entretanto, a quinta regra depende somente da variável "Energy". Nesta regra, quando "Energy" é atribuída como "Satisfied" a criatura apresenta o estado "Good" independentemente do valor da variável "Leaflet Percentage". E por fim, a sexta regra descreve a circunstância em que a criatura está com nível energético igual a "Satisfied" e também quando o percentual da tabela de trocas completada esteja também com estado "Satisfied". Por via de regra, quando o agente encontra-se nesta situação, pode-se dizer que ele já completou ou esta próximo de completar todas as combinações de sua tabela de trocas (de certa forma pronto para trocar suas joias por pontos no Ponto de Troca). Quando de fato isso acontece, não é necessário qualquer intervenção das emoções no comportamento da criatura, tendo em vista que o objetivo de completar todas as combinações da tabela de trocas já foi alcançado ou está próximo de ser atingido. Portanto, esta regra define o estado avaliativo como "Normal", deixando somente para os drives a função de finalizar a necessidade atual.

4.3.5 Codelets de Humor

Tanto na arquitetura do controlador quanto no modelo de subsistema motivacional, está prevista uma estrutura responsável por representar o "humor" do agente inteligente, algo como um estado de espírito mais duradouro, que influencia o comportamento do agente, por meio dos codelets emocionais. No modelo original, proposto na

Tabela 3 – Parâmetros do Sistema Fuzzy utilizado para determinar a Avaliação Primária do Agente Cognitivo.

Parâmetro	Valor
Modelo	Mamdami
Método de Defuzzificação	Centróide
Conjunção e Implicação (T-Norma)	Produto Algébrico ($at_pb = ab$).
Disjunção e Agregação (S-Norma)	Soma Probabilística ($as_pb = a + b - ab$).

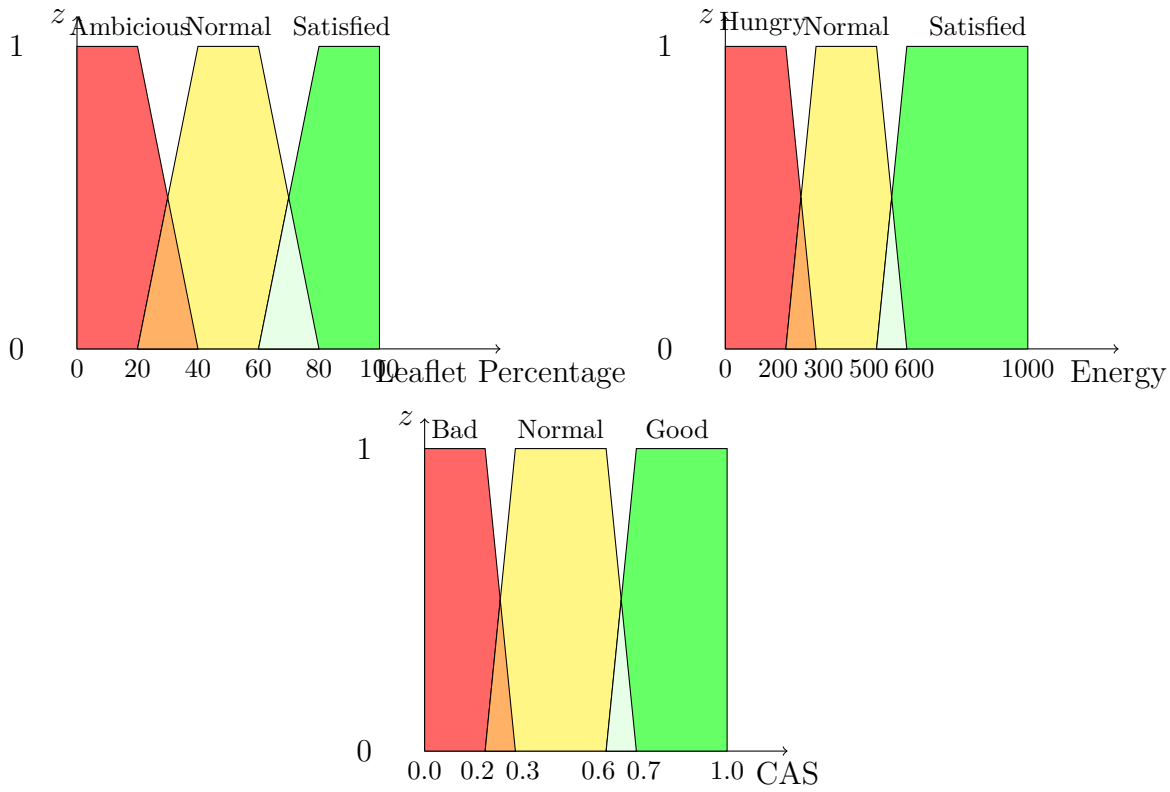

Figura 25 – Imagens das funções de pertinência utilizadas para determinar a Avaliação Primaria do agente cognitivo.

figura 15, a determinação deste humor pode utilizar três fontes de informação distinta: a memória de drives, a memória sensorial e a memória de apreciação. Todavia, na instância do controlador motivacional desenvolvida neste capítulo, utilizamos somente a apreciação da situação corrente fornecida pelo Codelet de Apreciação e armazenada na memória de apreciação. A determinação do humor da criatura é realizada pelos Codelets de Humor (ou *Mood Codelets*). Em nossa aplicação-exemplo, a geração de Humor é feita por dois Codelets distintos:

- Codelet de Humor de Ansiedade (*Anxiety Mood Codelet*).
- Codelet de Humor de Exaltação (*Exaltation Mood Codelet*).

Embora o humor da criatura seja divido e computado de forma separada, ambos

os Codelets utilizam como fonte de informação a memória de apreciação gerada pelo Codelet de Apreciação. O conteúdo dessa memória reflete uma avaliação do estado interno da criatura e é fator crucial para determinar se a criatura está ansiosa ou exaltada (ou seja, com um humor de ansiedade ou exaltação). Os codelets de Ansiedade e de Exaltação utilizam as variáveis *Eval* e *CAS* disponíveis no objeto "Appraisal" para computar o humor corrente da criatura. Os ciclos de processamento dos codelets de humor atualizam o humor da criatura, armazenando seus parâmetros na forma de um objeto da classe "Mood". A classe "Mood", assim como as demais estruturas do modelo de subsistema motivacional, foram disponibilizadas no pacote motivacional do projeto CST. Cada Mood gerado por cada Codelets de Humor é salvo dentro de um Memory Object, e em seguida, armazenado dentro das Memórias de Humor. A Figura 26 demonstra a interação entre o Codelet de Apreciação, a Memória de Apreciação e os Codelets de Humor.

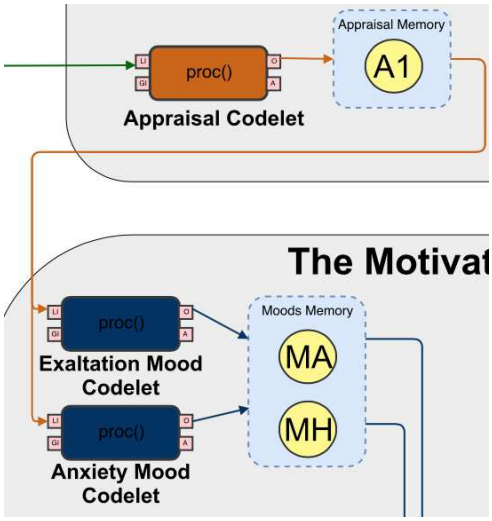

Figura 26 – Codelets de Humor gerando os Humores de Ansiedade e Exaltação, a partir da memória de avaliação do agente inteligente.

O atributo mais importante da classe Mood é o atributo de ativação do Humor, chamado aqui de μ_n. O atributo μ_n é uma variável numérica que varia entre -1 e 1. Quando valor de μ_n é maior do que zero, pode-se dizer que o respectivo humor está ativo e influenciando positivamente uma emoção específica do agente cognitivo. Em outra palavras, com μ_n positivo, o Mood faz com que uma respectiva emoção seja incentivada positivamente (se tornando ativa), gerando uma distorção cognitiva (δ) também positiva. Com δ positivo, as ativações dos drives afetados se elevam, fazendo com que os Codelets de Comportamento Motivacional mudem o comportamento da criatura, para que o agente possa satisfazer as necessidades modificadas pela emoção corrente. Caso contrário, se μ_n for menor do zero, o humor arrefece esta mesma emoção, ou seja, o Mood estimula o Codelet Emocional a gerar um δ negativo, resultando no decaimento do nível de ativação do drive, suprimindo os comportamentos ligados a ele. Este tipo de abordagem, pode

ser compreendida através da análise das equações 4.3 e 4.4, e também da definição 3.2. Vale a pena ressaltar que essa situação depende diretamente da forma como o usuário do subsistema motivacional implementa a heurística dos Codelets de Humor e Emocionais, ou seja, de acordo com as funções de ativação escolhidas para definir μ e δ.

Como dito anteriormente, dentro da estrutura do controlador motivacional existem dois Codelets de Humor responsáveis pela definição do humor da criatura artificial. O primeiro é o *Anxiety Mood Codelet* (ou Codelet de Humor de Ansiedade). Este Codelet tem a finalidade de representar a ansiedade do agente. Esta ansiedade é disparada quando a avaliação do estado interno da criatura atinge um nível energético crítico "Ruim", ou seja, quando o Codelet de Avaliação define o estado interno da criatura como sendo igual a "Bad". Quando a apreciação do estado corrente está classificada como "Good", o Mood ao invés de estimular o agente a ir à procura de comida, ele faz com que a variável *Eval* dos comportamentos orientados pelo drive de Fome fiquem em nível baixo. Sendo assim, pode-se dizer que neste caso o humor de ansiedade suprime os comportamentos de busca por alimentos quando o sistema está neste estado.

O Codelet de Humor de Exaltação (ou *Exaltation Mood Codelet*) trabalha de forma oposta ao Codelet de Humor de Ansiedade. Quando a avaliação do estado interno da criatura for configurado como "Bad", o humor de exaltação arrefece todos os comportamentos ligados ao drive de Ambição. Se a classificação for "Good", o humor de exaltação incentiva todos os comportamentos ligados a este drive.

Para ambos os Codelets de Humor, existe a possibilidade da Avaliação Primária ser definida como "Normal". Em particular neste caso, ambos os codelets determinam seus Moods com o valor de ativação igual a zero ($\mu_n = 0$). Neste contexto, dependendo das funções de ativação selecionadas para os Codelets Emocionais, o valor $\mu_n = 0$ não afetará a distorção cognitiva e nem mesmo o comportamento da criatura. Em outras palavras, quando μ_n for igual a zero, a função de ativação dos Codelets Emocionais também determinará distorções cognitivas nulas (ou seja, $\delta_n = 0$). As equações 4.1 e 4.2 definem o modo de operação dos Codelets de Humor de Ansiedade e Exaltação, onde *Eval* e CAS são as variáveis de avaliação do estado interno da criatura, sendo que μ_a e μ_e são as ativações dos Moods de Ansiedade e Exaltação.

$$\mu_a = \begin{cases} -Eval & \text{if } CAS = Good, \\ Eval & \text{if } CAS = Bad, \\ 0 & \text{if } CAS = Normal. \end{cases} \tag{4.1}$$

$$\mu_e = \begin{cases} Eval & \text{if } CAS = Good, \\ -Eval & \text{if } CAS = Bad, \\ 0 & \text{if } CAS = Normal. \end{cases} \tag{4.2}$$

4.3.6 Codelets Emocionais

Segundo o modelo de subsistema motivacional proposto neste trabalho, um comportamento emocional envolve a geração de uma "distorção cognitiva" na mente do agente inteligente. De acordo com as ideias propostas por Lazarus (1991) e Cañamero (1997), esta distorção cognitiva deve basear-se nas necessidade internas (drives), no humor e na apreciação atual do agente cognitivo. Conforme descrevemos no capitulo 3, são os Codelets Emocionais os responsáveis por implementar tal função no modelo de subsistema motivacional. Em nossa arquitetura de controlador motivacional utilizamos dois Codelets Emocionais:

- Hunger Emotional Codelet.
- Ambition Emotional Codelet.

Ambos os Codelets são responsáveis por computar as distorções cognitivas e portanto modificar os drives de Fome e Ambição, intensificando esses drives em situações excepcionais. Nas memórias de entrada de cada Codelet Emocional devem estar disponíveis um drive (que sofrerá a distorção cognitiva) e um Mood que será utilizado como base para computar o sinal da distorção cognitiva (positivo ou negativo). O Hunger Emotional Codelet utiliza um drive de Fome e um Mood de Ansiedade. Este codelet utiliza a ativação do Mood de Ansiedade (μ_a) para determinar a distorção cognitiva relacionada à Fome (δ_h), seguindo a função matemática definida pela equação:

$$\delta_h = \tanh\left(\frac{5\mu_a}{2}\right) \tag{4.3}$$

Na equação 4.3, a variável δ_h representa a distorção cognitiva do drive de Fome e μ_a a ativação do respectivo Mood (Humor de Ansiedade). O Ambition Emotional Codelet parte dos mesmos pressupostos que o Codelet anterior, utilizando a ativação do Mood de Exaltação para computar a distorção cognitiva, determinada pela equação:

$$\delta_a = \sinh\left(\frac{7\mu_e}{8}\right) \tag{4.4}$$

Na equação 4.4, δ_a e μ_e correspondem respectivamente à distorção cognitiva do drive e à ativação do Mood de Exaltação. Essas funções matemáticas podem ser visualizadas na Figura 27.

Conforme discutido na seção 2.1, não há consenso entre os autores sobre o que realmente são as Emoções, tampouco funções matemáticas canônicas para descrevê-las. A escolha das funções 4.3 e 4.4 foi baseadas em algumas propriedades matemáticas que estas funções apresentam. Por exemplo, calculando-se os limites $\lim_{\mu\to-1}\delta(\mu)$ e $\lim_{\mu\to1}\delta(\mu)$

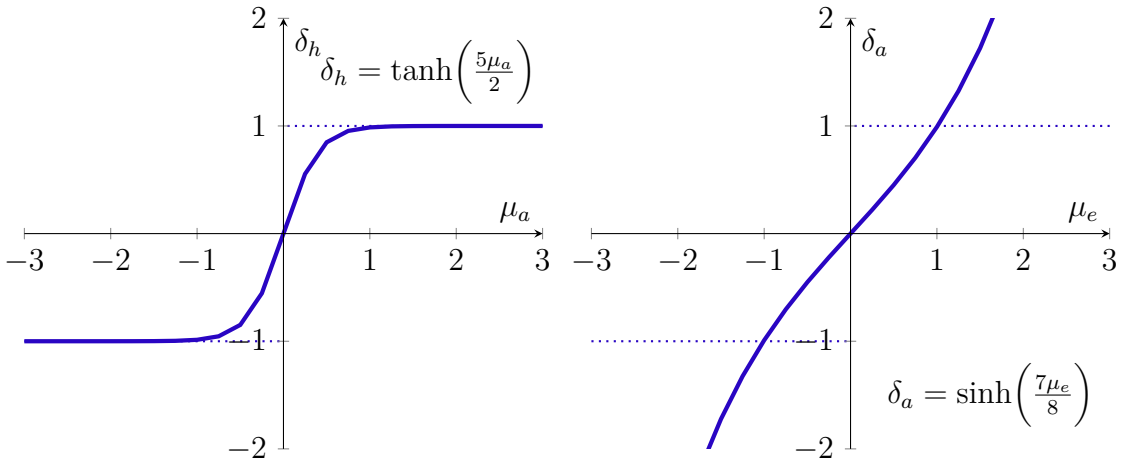

Figura 27 – A imagem da esquerda, demonstra a função matemática utilizada para computar a distorção cognitiva do drive de Fome e a imagem da direita apresenta a função matemática utilizada para computar a distorção cognitiva do drive de Ambição

tanto para a função da equação 4.3 quanto para 4.4, podemos ver que a distorção cognitiva δ gerada encontra-se entre -1 e 1. Essa característica encaixa-se perfeitamente com as condições demandadas pela definição 3.1, com relação à distorção cognitiva. Outra característica relevante é o fato de ambas serem funções continuas e não-lineares. Do ponto de vista do comportamento do agente, continuidade e não-linearidade das funções permite que o agente inteligente não sofra mudanças abruptas no comportamento quando há incidência de emoções na criatura, e consequentemente, permitindo que a criatura altere seu comportamento gradativamente conforme seu humor atual. A Figura 28 demonstra os Codelets Emocionais computando as distorções cognitivas dos drives de Fome e Ambição, e consequentemente, interferindo no comportamento do agente a partir dos codelets de Comportamento Motivacional.

4.3.7 Codelets de Geração de Goals

Em nosso controlador motivacional, o SMAN tem como finalidade solucionar o desafio da busca e captura de joias no ambiente visando maximizar a pontuação obtida. Funções como controle energético não fazem parte do escopo deste subsistema. Para que o agente cognitivo consiga obter a máxima pontuação, a arquitetura deve gerar *Goals* que resolvam este problema, e em seguida gerar planos que contemplem esses *goals*. Os Codelets responsáveis por essa tarefa são os Codelets de Planejamento e de Seleção de Planos. Entretanto, o funcionamento desses codelets depende anteriormente da existência de *goals* para que os planos possam ser elaborados. Assim, o Codelet de Geração de *Goals* além de fornecer parâmetros para o sistema de planejamento, também tem o objetivo de gerar *Goals* que descrevam um estado futuro desejável. Para isso, esses Codelets utilizam informações das memórias sensoriais. No caso do nosso controlador, o Codelet utiliza

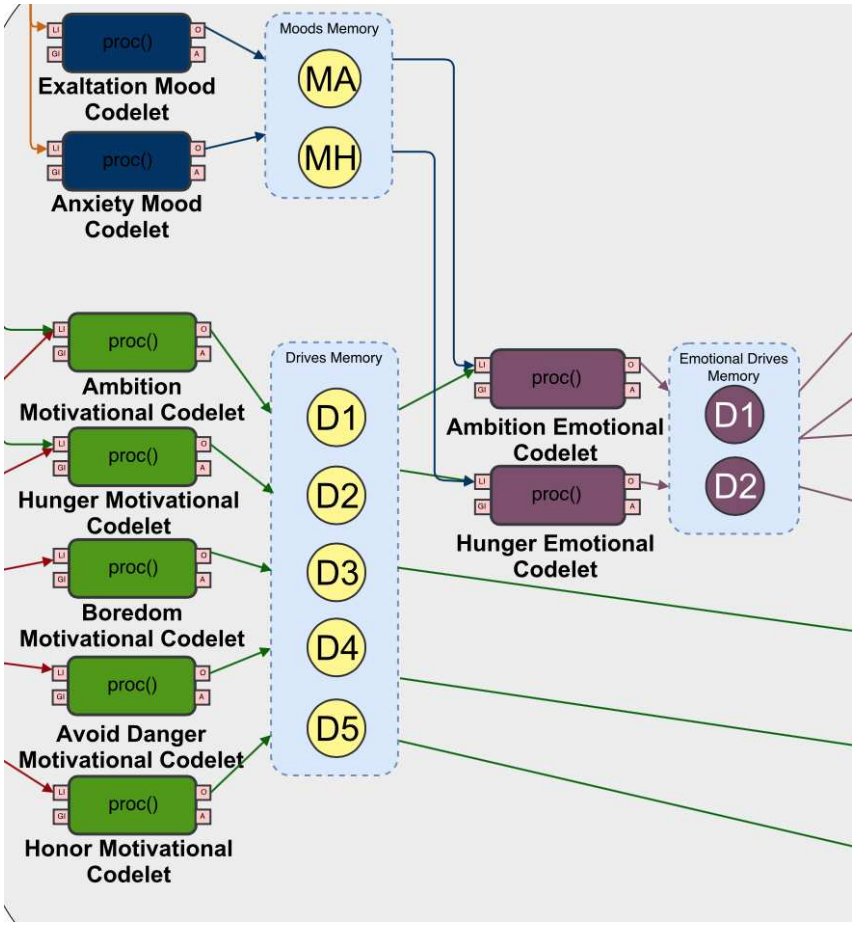

Figura 28 – Codelets Emocionais computando as distorções cognitivas e interferindo no comportamento motivacional.

informações da tabela de trocas para poder definir o objetivo de coletar as joias que ainda precisam ser capturadas.

Em nosso pacote de software, o Codelet de Geração de *Goals* utiliza a classe *Goal* para armazenar parâmetros de busca e descoberta de joias. Assim como as demais estrutura de dados citadas neste capitulo, a classe *Goal* faz parte do pacote motivacional do CST. Na estrutura da classe, existe um atributo *Goal Abstract Objects*, que é utilizado para descrever um estado futuro desejável. Este atributo armazena uma lista de objetos da classe *Abstract Object*, que descreve objetos existentes em uma cena de simulação observada pelo agente inteligente. Neste lista, é possível armazenar objetos de simulação e descrever propriedades desses objetos, tais como "Categoria", "Cor", "Posição" etc. Essa classe também permite a descrição de propriedades internas do próprio agente, como: "Energia", "Tabela de Trocas", "Joias e Maçãs Percebidas" etc. A Figura 29 demonstra a estrutura da classe *Abstract Object*. A estrutura de um *Abstract Object* pode ficar bastante complexa, incluindo Objetos Compositos (partes) e Agregados (para a descrição coletivos),

Propriedades e *Affordances*[1].

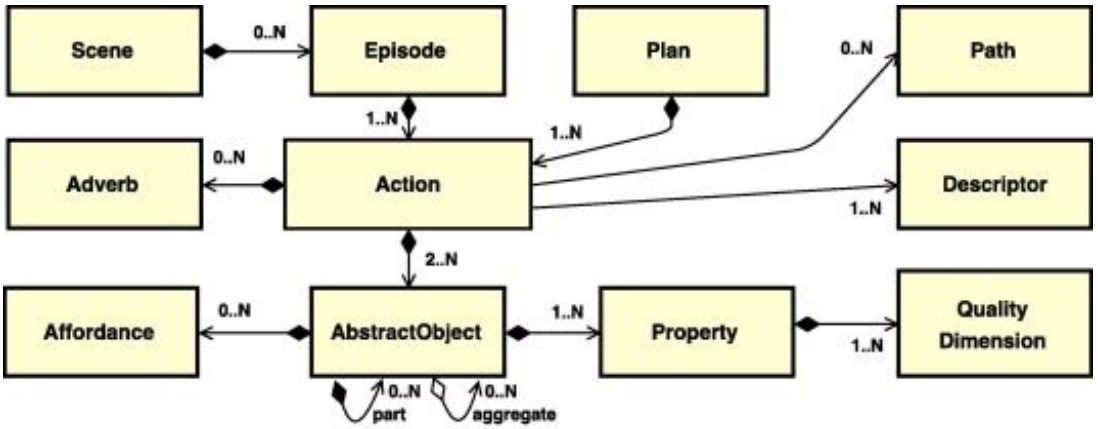

Figura 29 – A classe *Abstract Object* e suas relações com outros objetos do CST (GUDWIN *et al.*, 2017).

A classe *Abstract Object* foi concebida para ser capaz de representar diferentes tipos de objetos do mundo real, desde objetos simples com uma única propriedade (como um objeto *sensor*, com uma única propriedade *valor*), como coletivos de objetos com diferentes tipos de partes (por exemplo, uma frota de carros, onde cada carro possui diversas partes como rodas, bancos, volante, vidros, etc., e roda por sua vez pode ser composta por pneu, parafusos, etc.). De uma maneira primitiva um *Abstract Object* pode ser definido por uma lista de Propriedades que definem o objeto. Por exemplo, se queremos representar um objeto *carro*, poderíamos ter propriedades como a cor do carro, o modelo, ou o ano de fabricação. Propriedades são definidas a partir de dimensões de qualidade. No exemplo do carro, a propriedade "Cor" pode possuir as dimensões de qualidade "R", "G" e "B" para determinar uma determinada cor bem específica, sendo que para cada uma dessas dimensões de qualidade haveria um número associado como valor. Mas a representação de um objeto não se limita a um conjunto de propriedades. Pode também ser definida a partir de uma lista de partes (outros *Abstract Object*), chamadas de objetos "Compositos" ou de uma lista de "Agregados" (também outros *Abstract Object*). A lista de objetos "Compositos" diz respeito a partes que necessariamente devem existir e que compõem o objeto. Os objetos "Agregados", ao contrário, são utilizados para definir *Abstract Objects* que funcionam como coletivos de outro objetos, entendidos como sendo um objeto *de per si*. Por exemplo, quando descreve-se um carro, pode-se imaginar que o motor, as rodas, o chassi e carroceria são partes necessárias de um carro, pois sem eles um carro não poderia exercer sua função de carro. Portanto, esses sub-objetos são armazenados nas lista de objetos "Compositos" de um objeto carro. Já se queremos representar um objeto *frota*, podemos representá-lo agregando diversos objetos carro como "Agregados" do objeto *frota*. Vale a pena ressaltar, que tanto a lista de objetos "Compositos" quanto

[1] Ações que podem ser executadas sobre o objeto

a lista de objetos "Agregados" instancia também objetos do tipo *Abstract Object*. Desse modo, cada um destes pode, recursivamente, possuir objetos "Compositos" e "Agregados", gerando potencialmente uma estrutura de dados bastante complexa.

No caso do atributo "Goal Abstract Objects" da classe *Goal*, na lista de objetos "Compositos" estão todas as combinações da tabela de trocas da criatura. Dentre as propriedades que descrevem as combinações da tabela de trocas estão: o "Score" da tabela de trocas (utilizado para a troca de joias por pontos no Ponto de Troca) e também as joias coletadas com suas respectivas cores e quantidades, organizadas de acordo com seu tipo (por exemplo, três da cor azul, zero da vermelha, e assim por diante). O valor do "Score" e da lista de joias são armazenados como dimensões de qualidade de cada propriedade.

O Codelet de Geração de *Goals* em nosso controlador motivacional utiliza o atributo *Goal Abstract Objetcs* para armazenar a quantidade e as cores das joias que já foram capturadas e também das joias que ainda faltam. O codelet insere o *Goal* gerado dentro de um *Memory Object*, e em seguida, armazena-o nas Memórias de *Goals* (ou Goal Memory). A Figura 30 demonstra a geração de *Goals* a partir das informações sensoriais.

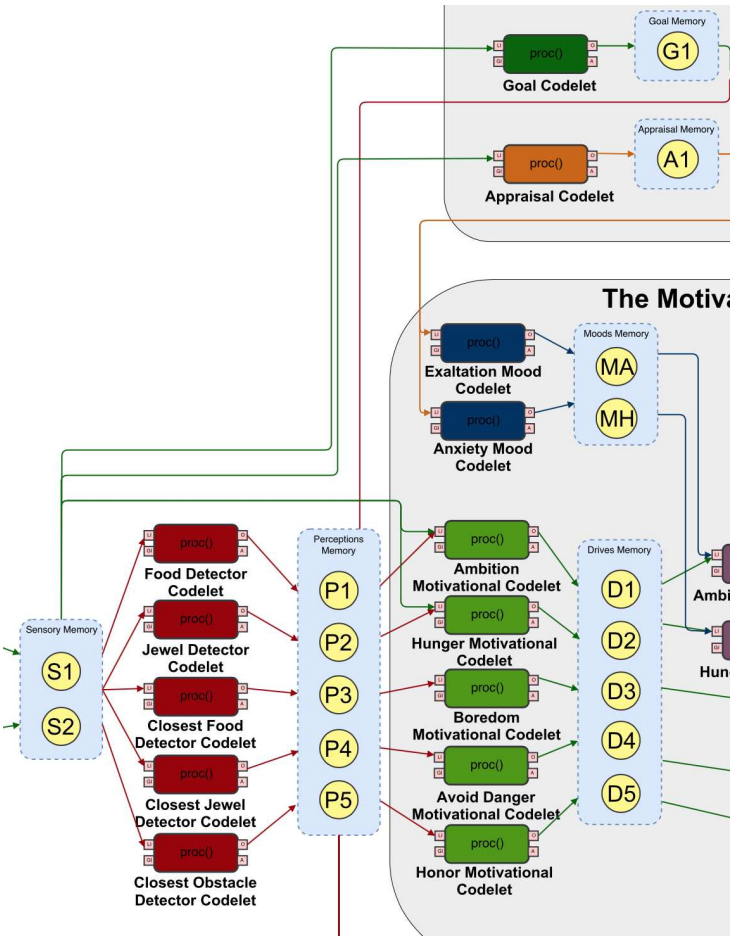

Figura 30 – Codelet de Geração de*Goals* utilizando informações sensoriais para gerar e definir um *Goal*.

4.3.8 Codelets de Planejamento e Seleção de Planos

Com o *Goal* gerado a partir do *Goal Codelet*, o Codelet de Planejamento é capaz de gerar planos destinados a resolver o problema da coleta de joias. Para que um plano seja gerado, o Codelet de Planejamento utiliza a estrutura de processamento de regras e planejamento da AC JSOAR. O JSOAR é uma versão em java da arquitetura cognitiva SOAR. O SOAR é uma arquitetura cognitiva simbólica onde utiliza-se de processamento de símbolos e regras para realizar o controle de agentes inteligentes (mais detalhes sobre o JSOAR podem ser encontrados nos links descritos no capítulo 7). No projeto do CST existem codelets especializados em fazer a integração do JSOAR com as estruturas do CST, como o "JSoarCodelet" e o "JSoarPlugin". Os Codelets de Planejamento utilizam esses codelets para que o controlador motivacional possa fazer uso do módulo de planejamento do JSOAR. Na seção 2.5, foi possível observar uma breve descrição do ciclo cognitivo AC SOAR e suas fase (INPUT->PROPOSE->...->HALT). Além disso, módulos essenciais foram descritos, juntamente com o entendimento de como as regras processam as informações sensoriais de entrada e como elas são dispostas na saída. Nos parágrafos a seguir, descrevemos como foram estruturadas os WMEs no módulo de percepção da AC, quais foram as regras implementadas e como as mesmas se comportam para a geração planos. O algoritmo da Figura 31 ilustra o estado inicial de uma busca.

```
(I2,CREATURE,I1)
   (I1,GOAL,I84)
      (I84,LEAFLET,I90)
         (I90,JEWELSINLEAFLET,I91)
            (I91,GREEN,1.0)
            (I91,YELLOW,1.0)
            (I91,MAGENTA,0.0)
            (I91,BLUE,1.0)
            (I91,WHITE,0.0)
            (I91,RED,0.0)
         (I90,INDEX,2.0)
         (I90,SCORE,18.0)
      (I84,LEAFLET,I88)
         (I88,JEWELSINLEAFLET,I89)
            (I89,GREEN,0.0)
            (I89,YELLOW,0.0)
            (I89,MAGENTA,0.0)
            (I89,BLUE,0.0)
            (I89,WHITE,2.0)
            (I89,RED,1.0)
         (I88,INDEX,1.0)
         (I88,SCORE,12.0)
```

```
      (I84,LEAFLET,I86)
         (I86,JEWELSINLEAFLET,I87)
            (I87,GREEN,2.0)
            (I87,YELLOW,0.0)
            (I87,MAGENTA,0.0)
            (I87,BLUE,0.0)
            (I87,WHITE,0.0)
            (I87,RED,1.0)
         (I86,INDEX,0.0)
         (I86,SCORE,26.0)
      (I84,GOALNAME,I85)
         (I85,VALUE,PickUpRemainingJewels)
   (I1,PERCEPTION,I4)
      (I4,JEWELS,I3)
        (I3,THING,I79)
            (I79,MATERIAL,I83)
               (I83,TYPE,GREEN)
            (I79,DISTANCE,I82)
               (I82,VALUE,281.090729797472)
            (I79,POSITION,I81)
               (I81,Y1,66.0)
               (I81,X1,339.0)
            (I79,NAME,I80)
               (I80,VALUE,Jewel_1511289694234)
```

Figura 31 – Estrutura do WMEs na disposição de um grafo dentro do JSOAR.

No algoritmo da Figura 31, o identificador “I2” é raiz do grafo. O ID “I2”, faz referência a todas as informações referentes à criatura, desde informações relevantes ao *Goal* até a percepção do agente cognitivo. Este identificador, está associado ao atributo “CREATURE”, que consequentemente está interligado com o ID “I1”. A “I1”, temos associados os atributos “GOAL”, com valor “I84” e “PERCEPTION”, com valor “I4”. Dessa forma, cada WME se conecta a outro WME e assim por diante, criando toda uma estrutura. Esta estrutura é somente uma sub-estrutura do estado dentro do SOAR, também chamada de “io.input-link”. Além do “io.input-link”, o SOAR coleta diversas outras informações, de maneira análoga, que são úteis no processo de busca. Outras sub-estruturas relevantes para nossos propósitos incluem o assim chamado “io.output-link”, em que podem ser armazenadas informações das decisões tomadas pelo sistema e a Memória Semântica (SMEM), com a capacidade de armazenar qualquer tipo de conhecimento perene adquirido pela criatura, que não se modifica a cada ciclo de inferência. No exemplo do nosso controlador motivacional, a SMEM é utilizada para armazenar os planos que ainda estão em concepção. Um vez que um plano seja construído, o SOAR limpa a SMEM e o ciclo planejamento se inicia novamente.

A partir do algoritmo da Figura 32, pode-se observar as regras utilizadas no controlador motivacional para geração de planos.

```
watch 5

sp {propose*init
    (state <s> ^superstate nil
               -^name)
-->
    (<s> ^operator <op> + =)
    (<op> ^name initSmem)
}

sp {apply*init
   (state <s> ^operator.name initSmem
              ^smem.command <cmd>)
-->
   (<s> ^name initSmem)
   (<cmd> ^plans <a> <b> <c>)
   (<a> ^index 0.0)
   (<a> ^counter 0)
   (<a> ^score 0)
   (<a> ^jewels <jewels1>)
   (<a> ^jewelsinleaflet <jewelsInLeaflet1>)
   (<b> ^index 1.0)
   (<b> ^counter 0)
   (<b> ^score 0)
   (<b> ^jewels <jewels2>)
   (<b> ^jewelsinleaflet <jewelsInLeaflet2>)
   (<c> ^index 2.0)
   (<c> ^counter 0)
   (<c> ^score 0)
   (<c> ^jewels <jewels3>)
   (<c> ^jewelsinleaflet <jewelsInLeaflet3>)
}

sp {propose*initPlans
    (state <s> ^name initSmem
               -^name initPlans)
-->
    (<s> ^operator <op> + =)
    (<op> ^name initPlans)
}

sp {apply*initPlans
   (state <s> ^operator.name initPlans
              ^smem.command <cmd>
              ^io.input-link <il>)
   (<il> ^CREATURE <creature>)
   (<creature> ^GOAL <goal>)
   (<creature> ^PERCEPTION <perception>)
   (<goal> ^LEAFLET <leaflet>)
   (<leaflet> ^JEWELSINLEAFLET.RED <red>)
   (<leaflet> ^JEWELSINLEAFLET.GREEN <green>)
   (<leaflet> ^JEWELSINLEAFLET.BLUE <blue>)
   (<leaflet> ^JEWELSINLEAFLET.YELLOW <yellow>)
   (<leaflet> ^JEWELSINLEAFLET.MAGENTA <magenta>)
   (<leaflet> ^JEWELSINLEAFLET.WHITE <white>)
   (<leaflet> ^INDEX <index>)
   (<leaflet> ^SCORE <score>)
```

```
   (<cmd> ^plans <plans>)
   (<plans> ^index <index>)
   (<plans> ^score <score2>)
   (<plans> ^jewelsinleaflet <jewelinleaflet>)
-->
   (<s> ^name initPlans)
   (<plans> ^score <score2> -
                        (+ <score2> <score>))
   (<jewelinleaflet> ^RED <red>)
   (<jewelinleaflet> ^GREEN <green>)
   (<jewelinleaflet> ^BLUE <blue>)
   (<jewelinleaflet> ^YELLOW <yellow>)
   (<jewelinleaflet> ^MAGENTA <magenta>)
   (<jewelinleaflet> ^WHITE <white>)
}

sp {propose*move*jewel
   (state <s> ^io.input-link <il>
              ^smem.command <cmd>)
   (<il> ^CREATURE <creature>)
   (<creature> ^GOAL <goal>)
   (<creature> ^PERCEPTION <perception>)
   (<perception> ^JEWELS <jewels>)
   (<jewels> ^THING <thing>)
   (<thing> ^MATERIAL.TYPE <color>
               ^NAME.VALUE <name>
               ^POSITION.X1 <x1>
               ^POSITION.Y1 <y1>
               ^DISTANCE.VALUE <distance>)
   (<cmd> ^plans <plans>)
   (<plans> ^jewelsinleaflet.<color> > 0
            ^index <index>
            ^score <score>)
   (<plans> ^jewels <jewelPlan>)
   -(<jewelPlan> ^SoarJewel.name <name>)

-->
   (<s> ^operator <o>)
   (<o> ^name getJewel)
   (<o> ^color <color>)
   (<o> ^jewelName <name>)
   (<o> ^x1 <x1>)
   (<o> ^y1 <y1>)
   (<o> ^distance <distance>)
   (<o> ^score <score>)
   (<o> ^index <index>)
}

sp {apply*move*jewel
   (state <s> ^operator <o>
              ^io <io>
              ^smem.command <cmd>)
   (<io> ^input-link <il>
        ^output-link <ol>)
   (<o> ^name getJewel)
```

```
   (<o> ^jewelName <name>)
   (<o> ^x1 <x1>)
   (<o> ^y1 <y1>)
   (<o> ^color <color>)
   (<o> ^distance <distance>)
   (<o> ^score <score>)
   (<o> ^index <index>)
   (<cmd> ^plans <id>)
   (<id> ^index <index>)
   (<id> ^counter <num>)
   (<id> ^jewels <jewels>)
   (<id> ^jewelsinleaflet <jewelinleaflet>)
   (<jewelinleaflet> ^<color> <qtd>)
-->
   (<id> ^counter <num> -
                         (+ <num> 1))
   (<jewelinleaflet> ^<color> <qtd> -
                         (- <qtd> 1))
   (<jewels>  ^SoarJewel <newJewel>)
   (<newJewel>  ^name <name>)
   (<newJewel>  ^color <color>)
   (<newJewel>  ^x1 <x1>)
   (<newJewel>  ^y1 <y1>)
   (<newJewel>  ^distance <distance>)
   (<newJewel>  ^score <score>)
}

sp {propose*monitoring*desire*state
   (state <s> ^smem.command <cmd>)
   (<cmd> ^plans <plans>)
   (<plans> ^index <index>)
   (<plans> ^counter 3)
-->
   (<s> ^operator <o> +)
   (<o> ^name desireState)
   (<o> ^planIndex <index>)
}

sp {apply*desire*state
   (state <s> ^operator <o>
              ^smem.command <cmd>
              ^io <io>)
   (<io> ^input-link <il>
         ^output-link <ol>)
   (<o> ^name desireState)
   (<o> ^planIndex <index>)
   (<cmd> ^plans <plans>)
   (<plans> ^index <index>)
   (<plans> ^jewels <jewels>)
-->
   (<ol> ^SoarPlan <goal>)
   (<goal> ^ArraySoarJewels <jewels>)
   (halt)
}
```

```
sp {move*jewel*preferences*monitoring*desire*state
   (state <s>  ^operator <o> +
                         <o2> +)
   (<o> ^name desireState)
   (<o2> ^name getJewel)
-->
   (<s> ^operator <o> > <o2>)
}

sp {move*jewel*preferences*move*jewel
   (state <s>  ^smem.command <cmd>
               ^operator <o> +
                         <o2> +)
   (<o2> ^score <score2>)
   (<o> ^score <score> > <score2>)
-->
   (<s> ^operator <o> > <o2> =)
}

sp {move*jewel*preferences*initSmem
   (state <s> ^operator <o> +
                        <o2> +)
   (<o> ^name initSmem)
   (<o2> ^name getJewel)
-->
   (<s> ^operator <o> > <o2>)
}

sp {move*jewel*preferences*initPlans
   (state <s> ^operator <o> +
                        <o2> +)
   (<o> ^name initPlans)
   (<o2> ^name getJewel)
-->
   (<s> ^operator <o> > <o2>)
}
```

Figura 32 – Exemplo de regras utilizado pelo JSOAR durante a etapa de planejamento.

Em nosso exemplo, dividimos as regras em três tipos (algumas delas executam mais de uma função em uma única regra). O primeiro tipo foi batizado de propose*<algo>, onde <algo> varia com o nome do operador que está sendo proposto. Estas regras servem para a proposição de operadores. O segundo tipo foi batizado de apply*<algo>, e promove a modificação de estado quando o operador <algo> foi selecionado para aplicação. O terceiro tipo de regras foi batizado de move*jewel*preferences*<algo>, e é utilizada para definir a prioridade de seleção quando dois operadores distintos disputam para serem selecionados.

Quando a criatura visualiza uma joia que deveria ser coletada, a regra de proposição “propose*move*jewel” sugere que ela seja inserida dentro da SMEM, para fazer parte de um plano futuro que possa ser gerado. Como dito anteriormente, é possível em um ciclo que diversos operadores sejam propostos ao mesmo tempo. Esses operadores

podem ser originados de regras diferentes ou de uma mesma regra. No caso da regra "propose*move*jewel", a mesma pode ser ativada mais de uma vez, pois o agente pode visualizar outras joias no ambiente e sugeri-las para que sejam inseridas em um plano. Neste caso, se o SOAR propuser mais do que uma instância do operador "move*jewel", então a regra de seleção de operadores "move*jewel*preferences*move*jewel" seleciona o operador específico que busca a joia associada à combinação da tabela de trocas com maior "Score". Subsequentemente, com este operador criado e selecionado pela regra de priorização, a regra de aplicação "apply*move*jewel" armazena os dados referente à joia dentro da SMEM. Conforme as joias são inseridas dentro dos planos presente na SMEM, a regra de proposição "propose*monitoring*desire*state" tem por objetivo verificar se os planos que estão sendo compostos dentro da SMEM estão completos ou não. Caso o operador desta regra seja ativo, a regra de aplicação "apply*desire*state" estrutura o plano completo na saída do ID "output-link". Vale a pena ressaltar que no caso da regra de proposição "propose*monitoring*desire*state", também pode haver momentos de impasse com operadores gerados a partir da regra "propose*move*jewel". Sendo assim, a regra "move*jewel*preferences*monitoring*desire*state" prioriza o operador da regra "propose*monitoring*desire*state" ao invés do operador gerado pela regra "propose*move*jewel". A regra "apply*desire*state", além de armazenar o plano completo na saída "output-link", também executa a instrução "(halt)", ou seja, também detecta que o estado desejado foi atingido. Como dito anteriormente, quando uma regra executa esta instrução, o SOAR finaliza o processamento de regras. Encerrando-se o ciclo de inferência do SOAR, o codelet de planejamento cria uma instancia da classe "Plan" (classe também presente nos pacotes do CST), encapsula o mesmo em um *Memory Object* e o disponibiliza na "Memória de Novo Plano". A Figura 33 demonstra a interação entre a memória de objetivo, memória perceptual e a memória de novo plano. Dentre as regras existentes, também estão as regras de proposição "propose*init" e "propose*initPlans" , juntamente com as regras de aplicação "apply*init" e "apply*initPlans". Elas são executadas no início do processamento de regras do SOAR e são utilizadas para estruturar a memória semântica (SMEM) permitindo que a mesma possa armazenar os planos de acordo com o score, o índice e as joias de cada tabela da criatura artificial.

Com o plano gerado e armazenado na "Memória de Novo Plano", o Codelet de Seleção de Planos transfere essa memória para a "Memória de Planos" (conforme demonstrado na figura 33). Antes de adicionar um novo plano na memória de planos, o Codelet primeiramente verifica se esse plano ainda é factível de ser executado, e também verifica se já existe um plano equivalente na memória. Caso já exista um plano equivalente, o novo plano é descartado. Caso contrário, o plano passa pela verificação de "factibilidade". Essa verificação é um fator importante para o controlador como um todo, pois caso alguma joia dentro de um plano já não exista mais no ambiente, então este plano já não é mais factível de ser executado. Esse tipo de situação é passível de acontecer, pois uma vez que

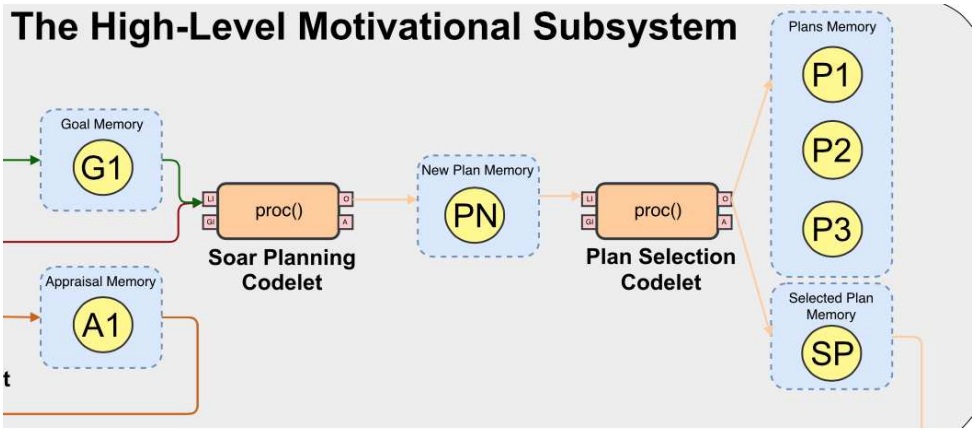

Figura 33 – Os Codelets de Planejamento e de Seleção de Planos interagindo com as memórias de Goals, perceptual e de novo plano.

a criatura também é controlada pelo subsistema motivacional de subsunção, o drive de ambição pode estimular a criatura a capturar as joias de um plano antes mesmo que ele tenha sido gerado e selecionado pelos Codelet de Planejamento e Seleção de Planos. Caso o plano seja categorizado como infactível, o Codelet altera o estado do plano e ele não será mais selecionado. Por outro lado, se o plano ainda é plausível de ser executado, o mesmo continua com o estado de "ativo" até que o Codelet selecione-o para execução. Este algoritmo de verificação, é executado no início do processamento do Codelet de Seleção, para todos os planos que ainda estão ativos. Após esta verificação, o Codelet seleciona os planos conforme a ordem de geração e inserção na memória de planos, ou seja, o plano que foi inserido primeiramente será o primeiro a ser executado (implementando uma fila de planos do tipo FIFO - First In, First Out). Com o plano selecionado, o Codelet reorganiza as joias do plano de forma crescente, levando em consideração distâncias das joias até a criatura, e em seguida, o mesmo é transferido para a "Memória de Planos Selecionados".

4.3.9 Codelets de Comportamento Motivacional

Os Codelets de Comportamento Motivacional (ou *Motivational Behavior Codelets*) pertencem a uma parte do subsistema motivacional que deve definir um comportamento especifico para a criatura, enviando instruções de atuação para os Codelets Motores (ou *Motor Codelets*). Na saída de cada Codelet Comportamental existem duas variáveis que são fundamentais para a operação do sistema de subsunção dinâmico, que são: E_i e X_i. A variável E_i representa a avaliação de um comportamento. Da mesma forma que os drives, os Codelets Comportamentais apresentam uma variável de ativação (que neste caso é E_i), cujo valor é utilizado pelo sistema de subsunção para definir qual comportamento será executado em um determinado instante de tempo. No caso dos Codelets Comportamentais, o valor de E_i é computado utilizando as ativações fornecidas por um ou mais drives. O cálculo realizado para determinar o valor da variável, depende diretamente da

heurística utilizada pelo usuário do Subsistema Motivacional. Entretanto, no controlador aqui proposto, os Codelets Comportamentais somente apresentam um único drive em sua memória de entrada, portanto, o valor da variável E_i é definido de acordo com a ativação (conforme definido na definição 3.2) do único drive de entrada. Outro parâmetro definido por esses Codelets é a variável X_i. A variável X_i contém o comando motor a ser enviado aos Codelets Motores. Nesta variável, é possível armazenar qualquer tipo de dado, porém no modelo de controlador motivacional, pode-se inserir instruções como: movimentação da criatura para uma determinada posição, informar posições de objetos que estão no ambiente, comandos para o agente enterrar e desenterrar objetos, comandos para comer maçãs e nozes, comandos para capturar joias, ou até mesmo comandos de desvio para que o agente evite colisões. Consequentemente, o conteúdo desta variável também depende diretamente do tipo de comportamento que o Codelet Comportamental é capaz de executar, mas no exemplo atual, as informações inseridas na variável X_i são uma composição das informações perceptuais juntamente com instruções de comando para execução de um comportamento. Essa informação é estruturada utilizando-se a tecnologia JSON, e em seguida, enviadas para os Codelets Motores. No controlador motivacional, existem sete tipos diferentes de Codelets de Comportamento Motivacional que são:

- *Go To Jewel Motivational Behavioral Codelet.*
- *Go To Food Motivational Behavioral Codelet.*
- *Get Jewel Motivational Behavioral Codelet.*
- *Eat Food Motivational Behavioral Codelet.*
- *Go To Delivery Spot Motivational Behavioral Codelet.*
- *Avoid Danger Motivational Behavioral Codelet.*
- *Random Movement Motivational Behavioral Codelet.*

Cada um destes Codelets tem a finalidade de fornecer comandos para um Codelet Motor específico do controlador. Alguns deles configuram os atuadores que controlam as pernas (*Legs Motor Codelet*) do agente, e outros configuram os que controlam as mãos (Hands Motor Codelet). Os Codelets que enviam comandos para o *Legs Motor Codelet* são: *Go To Jewel*, *Go To Food*, *Go To Delivery Spot*, *Avoid Danger* e *Random Movement*. O codelet *Go To Jewel*, por exemplo, comanda a criatura a executar o comportamento de movimentação do agente até a joia mais próxima pertencente às suas combinações da tabela de trocas. Em contrapartida, o *Go To Food* comanda o comportamento de movimentação da criatura na direção do alimento (ou alimento que possa estar enterrado) mais próximo. Para efetuar a movimentação de ir até o *Delivery Spot*, o controlador utiliza-se

do codelet *Go To Delivery Spot*. Já as movimentações aleatórias são comandadas pelo codelet *Random Movement*. E por fim, o *Avoid Danger Codelet* possui a finalidade de executar movimentos de desvios de obstáculos. Este Codelet é o único da arquitetura comandando comportamentos em ambos os Codelets Motores, pois comportamentos distintos podem ser necessários, dependendo da situação que a criatura enfrenta durante a simulação. Por exemplo, se a criatura detecta uma possível colisão com um obstaculo, o codelet parametriza comandos de desvio e os envia para o *Legs Motor Codelet*. Todavia, se a criatura detectar possível colisão com uma joia indesejável, o codelet envia comandos para os atuadores das mãos para que o agente enterre esta joia indesejável que está em seu caminho. Este último comportamento é de grande utilidade para a movimentação da criatura, pois uma vez que uma joia indesejável seja enterrada, essa joia não irá mais atrapalhar a movimentação da criatura durante a simulação. Caso este objeto não seja uma joia, e sim um alimento, o comportamento do codelet dependerá diretamente do estado interno da criatura. Se ela estiver com fome, o codelet enviará comandos de captura e uso da comida. Caso contrário, a criatura enterra a comida para que ela possa se alimentar posteriormente quando houver casos de urgência energética. Finalmente, o *Avoid Danger Codelet* pode também atuar nos *Hands Motor Codelet*, quando a criatura completa um ou mais de suas combinações da tabela de trocas. Nesta situação, quando a criatura estiver próximo de uma colisão com o Ponto de Troca, o *Avoid Danger Codelet* elabora um comportamento para criatura, de modo que a mesma possa trocar suas joias coletadas por pontos.

Como mencionado anteriormente, existem codelets Comportamentais que atuam somente sobre o *Hands Motor Codelets*: *Get Jewel Codelet* e *Eat Food Codelet*. Tendo em vista que a criatura necessita regular seu nível energético, o *Eat Food* é um codelet Comportamental que tem por objetivo enviar comandos para o Hands Motor Codelet para que a criatura capture a comida do ambiente, e em seguida, a coma. O *Get Jewel Codelet* é ligeiramente similar. Porém, este envia comandos para os atuadores da mão com o objetivo de capturar uma joia desejada, e logo depois, colocá-la em seu saco de joias. Os comportamentos *Go To Jewel* e *Get Jewel* são ativados pelo drive de Ambição. Em contrapartida, os comportamentos *Go To Food* e *Eat Food* são determinados pelo drive de Fome. Já o comportamento de *Avoid Danger* é disparado pelo drive de Prevenção de Danos. Por fim, comportamentos como *Random Movement* e *Go To Delivery Spot* são disparados pela ativação dos drives de Tédio e Honra respectivamente.

Um fator fundamental para compreender a operação do subsistema motivacional como um todo, é entendermos a interação do SMAN com o SMS por meio dos planos gerados e selecionados pelos codelets de Planejamento e Seleção de Planos. Quando o SMAN seleciona um plano para ser executado, esse plano é enviado para a entrada global dos codelets *Get Jewel* e *Go To Jewel*. A partir disso, esses codelets de Comportamento utilizam somente as instruções definidas no plano, que são consideradas prioritárias com

relação àquelas advindas das entradas locais. Dessa forma, os comandos de atuação são inseridos na variável X_i e a ativação do drive de Ambição não é mais utilizada para atribuir o valor de variável E_i, pois nesta situação, o plano é considerado prioritário sobre o drive de Ambição. Neste caso, o codelet Comportamental atribui o valor de nível de urgência, juntamente com a prioridade do drive de ambição para a variável E_i. Contudo, o fator mais relevante desta heurística é que apesar dos planos serem prioritários sobre os drives, eles podem não ser prioritários em comparação a outros drives presentes em outros comportamentos. Por exemplo, se a criatura está em nível de urgência no drive de Fome e ao mesmo tempo o agente apresenta um plano a ser executado, os comportamentos que terão mais prioridade na execução serão o *Go To Food* e o *Eat Food*. Isso acontece porque o drive de Fome é mais prioritário do que o drive de Ambição, e como neste cenário os comportamentos de *Go To Jewel* e *Get Jewel* utilizam o nível de urgência do drive de ambição para compor sua avaliação, os comportamentos selecionados e executados são o *Go To Food* e o *Eat Food*. As equações 4.5 e 4.6 abaixo definem as atribuições das variáveis E_i e X_i quando há planos selecionados e quando não há.

$$E_i = \begin{cases} 0.5 + p_i & \text{if } A_i > \theta_i \vee \zeta = 1, \\ (A_i + \delta_i)/2 & \text{if } A_i \leq \theta_i \wedge \zeta = 0 \end{cases} \tag{4.5}$$

$$X_i = \begin{cases} P_n & \text{if } \zeta = 1, \\ X_i & \text{if } \zeta = 0 \end{cases} \tag{4.6}$$

Na equação 4.5, E_i é a variável de avaliação do comportamento, p_i, A_i, θ_i e δ_i representam respectivamente as variáveis de prioridade, nível de ativação, limiar de urgência e distorção emocional do drive entrada do comportamento motivacional, e por fim, ζ representa a presença da seleção de um plano ocasionado pelo Codelet de Seleção de Planos. Quando ζ é igual a 1, significa que um plano foi selecionado e enviado para os Codelets de Comportamento Motivacional. Caso contrário, ζ igual a 0, significa que nenhum plano foi selecionado. Em contrapartida, na equação 4.6, X_i são os respectivos comandos gerados pelo Codelet de Comportamento Motivacional. Já P_n, são os comandos do plano selecionado pelo Codelet de Seleção de Planos.

A Figura 34 ilustra os Codelets de Comportamento Motivacional sendo alimentados pelos drives, a memória perceptual e também por planos gerados e selecionados pelos Codelets de Planejamento e Seleção de Planos. A Tabela 4 descreve as heurísticas utilizadas em cada Codelet Comportamental.

4.3.10 Codelets Motores

Como comentamos na seção anterior, na arquitetura do controlador motivacional existem dois tipos de Codelets Motores. Estes Codelets são responsáveis por executar

Tabela 4 – Parâmetros dos Codelets de Comportamento Motivacional

Codelet	Comportamento	Avaliação
Go To Jewel M. B.	Movimentação da criatura até a joia desejada mais próxima.	$E_1 = \begin{cases} 0.5 + p_a & \text{if } A_a > \theta_a \vee \zeta = 1, \\ (A_a + \delta_a)/2 & \text{if } A_a \leq \theta_a \wedge \zeta = 0. \end{cases}$
Get Jewel M. B.	Capturar a joia desejada.	$E_2 = \begin{cases} 0.5 + p_a & \text{if } A_a > \theta_a \vee \zeta = 1, \\ (A_a + \delta_a)/2 & \text{if } A_a \leq \theta_a \wedge \zeta = 0. \end{cases}$
Go To Food M. B.	Movimentação da criatura até a comida mais próxima.	$E_3 = \begin{cases} 0.5 + p_h & \text{if } A_h > \theta_h, \\ (A_h + \delta_h)/2 & \text{if } A_h \leq \theta_h. \end{cases}$
Eat Food M. B.	Comer alimentos presentes no ambiente.	$E_4 = \begin{cases} 0.5 + p_h & \text{if } A_h > \theta_h, \\ (A_h + \delta_h)/2 & \text{if } A_h \leq \theta_h. \end{cases}$
Avoid Danger M. B.	Evitar colisões com obstaculos, enterrar e desenterrar alimentos, enterrar joias indesejáveis e troca das joias coletadas por pontos no Ponto de Troca.	$E_5 = \begin{cases} 0.5 + p_d & \text{if } A_d > \theta_d, \\ (A_d + \delta_d)/2 & \text{if } A_d \leq \theta_d. \end{cases}$
Random Movement M. B.	Movimentação do agente para posições randômicas do ambiente.	$E_6 = \begin{cases} 0.5 + p_b & \text{if } A_b > \theta_b, \\ (A_b + \delta_b)/2 & \text{if } A_b \leq \theta_b. \end{cases}$
Go To Delivery Spot M. B.	Movimentação da criatura para o Ponto de Troca.	$E_7 = \begin{cases} 0.5 + p_o & \text{if } A_o > \theta_o, \\ (A_o + \delta_o)/2 & \text{if } A_o \leq \theta_o. \end{cases}$

onde A_a, A_h, A_d, A_b e A_o são as ativações dos respectivos drives: Ambição, Fome, Prevenção de Danos, Tédio e Honra, p_a, p_h, p_d,p_b e p_o são as prioridades de cada drive citados anteriormente, θ_a, θ_h, θ_d, θ_b e θ_o são os limiares de urgência, δ_a, δ_h, δ_d, δ_b e δ_o são as distorções emocionais (geradas pelos Codelets Emocionais), e por fim ζ representa a seleção ou não de um plano para ser executado. Um valor de ζ igual a 1 significa que um plano foi selecionado para ser executado. Se ao contrário, ζ for igual a zero, o Codelet de Seleção de Planos não apresenta nenhum plano factível para ser executado no momento.

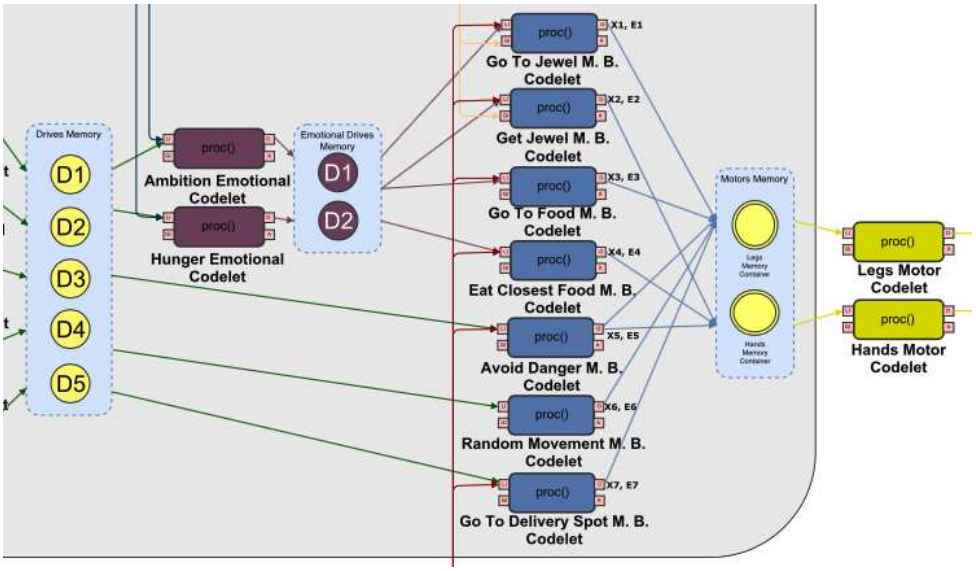

Figura 34 – Troca de mensagens entre os Codelets Perceptuais, Motivacionais, de Planejamento e Seleção de Planos.

os comportamentos parametrizados pelos Codelet de Comportamento Motivacional nos atuadores da criatura (através do uso da biblioteca *WS3D Proxy*). Esses codelets operam os atuadores das pernas e mãos do agente cognitivo. Os Codelet Motores presentes em nosso controlador motivacional são:

- *Legs Motor Codelet.*
- *Hands Motor Codelet.*

A Figura 35 ilustra os codelets motores *Legs Motor Codelet* e *Hands Motor Codelet* enviando comandos para a criatura utilizando o WS3DProxy.

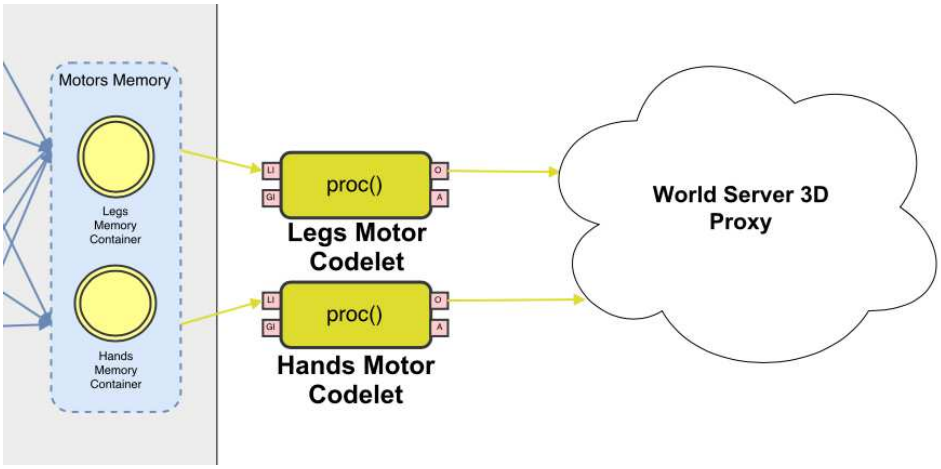

Figura 35 – Os Codelets Motores

Os *Memory Objetcs* gerados pelos Codelets de Comportamento Motivacional são armazenados na memória motora em *Memory Containers*. Cada Codelet Motor apresenta em sua memória de entrada um *Memory Container* próprio (*Legs Memory Container* e *Hands Memory Container*). Como dissemos anteriormente, *Memory Containers* são estruturas de dados que permitem o armazenamento de diversos *Memory Object* de um mesmo tipo, vindos de diferentes codelets e que permitem a seleção do *Memory Object* com a maior avaliação (maior valor E_i) dentre estes, para efeito de encadeamento com codelets subsequentes, implementando o mecanismo de subsunção dinâmica, conforme descrito na seção 3.1. Sendo assim, diferentes codelets comportamentais podem enviar seu comandos para um mesmo *Memory Container*, e quando o codelet motor for ler esse *Memory Container*, receberá dele o *Memory Object* com maior ativação, dentre todos os enviados pelos codelets comportamenais. Assim, o Codelet Motor simplesmente executa o comportamento parametrizado (de acordo com os valores de X_i) do Codelet de Comportamento Motivacional com maior valor E_i, dentre todos os enviados ao respectivo *Memory Container*. As seguintes ações no Legs Motor Codelet e no Hands Motor Codelet podem ser comandadas:

- Codelet Motor das Pernas (*Legs Motor Codelet*):
  - Rotação da Criatura.
  - Ir em direção a um Objeto (Joia, Alimento ou Ponto de Troca).
  - Movimentação aleatória.
  - Evitar Colisões com Obstáculos.
- Codelet Motor das Mãos (*Hands Motor Codelet*):
  - Capturar Joias.
  - Enterrar Jóias Indesejadas e Alimentos.
  - Desenterrar Alimentos.
  - Comer Alimentos.
  - Trocar Joias Coletadas por Pontos.

5 Discussão e Análise dos Resultados.

Este capítulo tem por objetivo apresentar os resultados obtidos nos experimentos executados, e além disso, promover uma discussão comparativa entre a eficiência e eficácia do controlador motivacional perante aos outros controladores desenvolvidos. As Figuras 36, 37, 38, 39 e 40 demonstram os resultados obtidos nos dez experimentos para os controladores: Reativo, JSOAR, CLARION, LIDA e Subsistema Motivacional do CST.

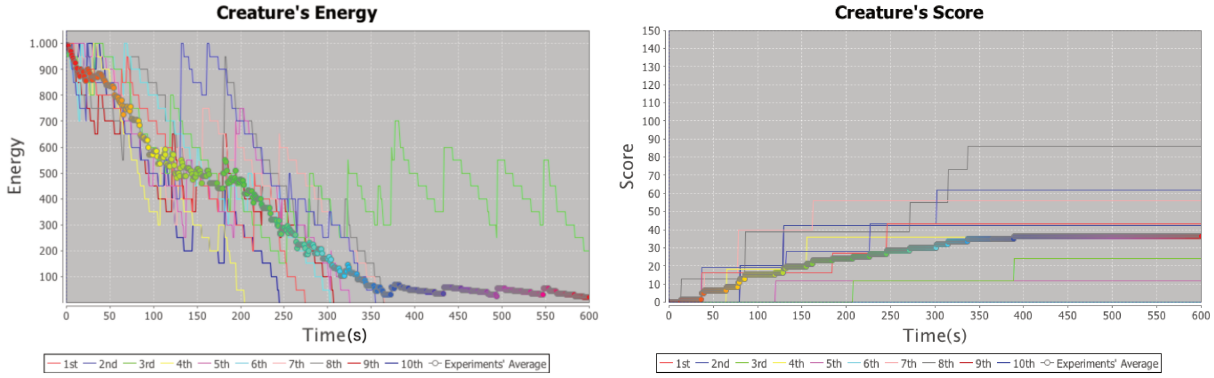

Figura 36 – Gráficos de Desempenho Energético e de Pontos Obtidos dos experimentos executados com o controlador desenvolvido com o controlador reativo.

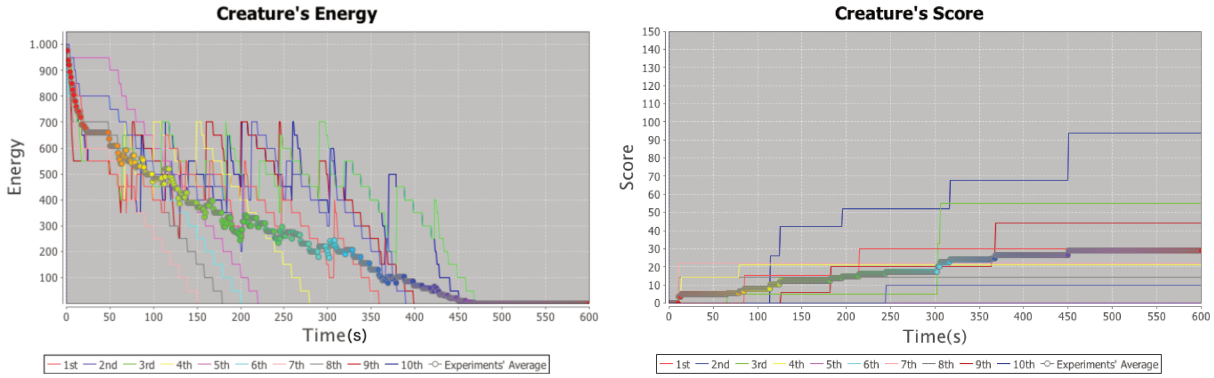

Figura 37 – Gráficos de Desempenho Energético e de Pontos Obtidos dos experimentos executados com o controlador desenvolvido com a arquitetura cognitiva JSOAR.

Os gráficos dessas figuras mostram o nível energético e os pontos obtidos pelas criatura ao longo do tempo de simulação. No gráfico de desempenho do nível energético, o eixo da ordenadas mede o nível de energia da criatura, registrada ao longo do tempo de simulação. Seu valor pode variar entre 0 e 1000. O eixo das abcissas apresenta o tempo de simulação que varia entre 0 e 600 segundos (10 minutos). Da mesma maneira que no gráfico de desempenho energético, o gráfico de pontos obtidos mostra no eixo das abcissas o tempo de simulação, e nas ordenadas o número de pontos obtidos, a partir das trocas de

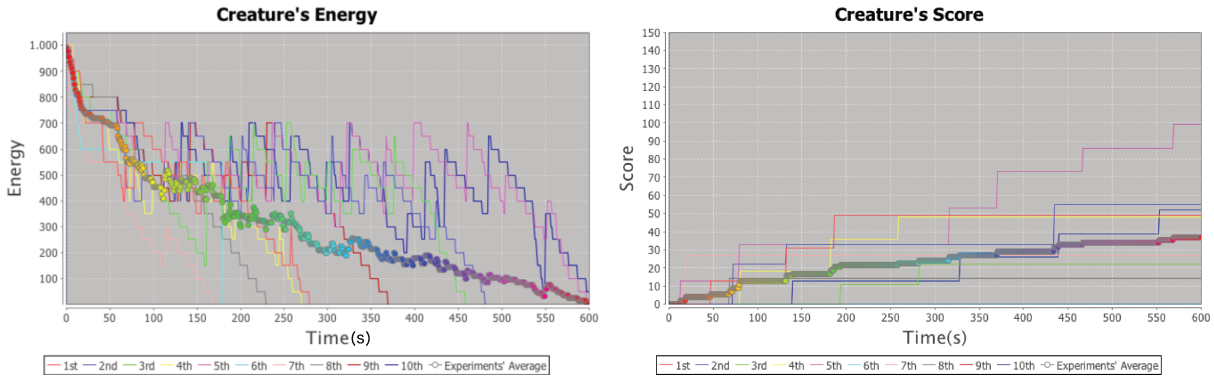

Figura 38 – Gráficos de Desempenho Energético e de Pontos Obtidos dos experimentos executados com o controlador desenvolvido com a arquitetura cognitiva CLARION.

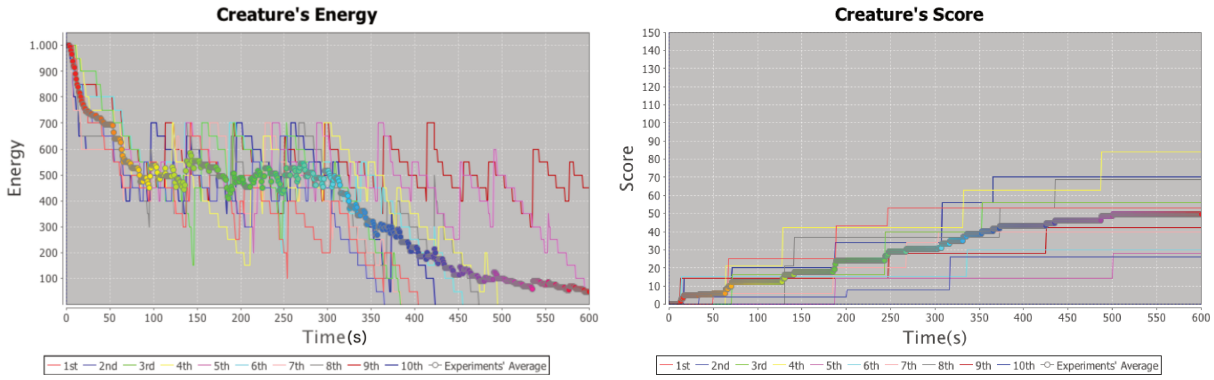

Figura 39 – Gráficos de Desempenho Energético e de Pontos Obtidos dos experimentos executados com o controlador desenvolvido com a arquitetura cognitiva LIDA.

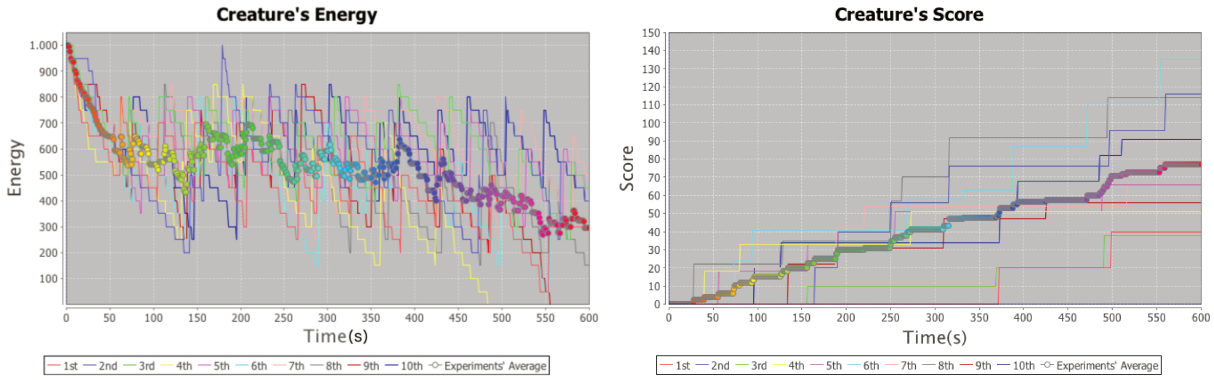

Figura 40 – Gráficos de desempenho Energético e de Pontos Obtidos dos experimentos executados com o controlador desenvolvido com o Subsistema Motivacional do CST.

joias por pontos, de acordo com a tabela de trocas da criatura. Os pontos obtidos podem variar entre 0 e o máximo de pontos alcançado pelo melhor experimento. Em ambos os gráficos são apresentados os resultados de todos os dez experimentos executados, representados por linhas finas coloridas, juntamente com o média calculada a cada segundo,

representada por uma linha com pequenos círculos (linha mais grossa). Com base nos dados fornecidos pelos gráficos de simulação é possível avaliar os controladores de cada experimento e também fazer um comparativo entre eles, tendo em vista a média obtida nas 10 simulações. As informações dos gráficos permitem computarmos a eficiência e a eficácia dos controladores. As Tabelas 5, 6, 7, 8 e 9 apresentam os resultados obtidos nos experimentos para todos os controladores de agentes inteligentes utilizando os seguintes parâmetros: Máxima Pontuação Obtida, Menor Nível Energético/Tempo de Simulação, Média, Mediana, Desvio Padrão e Variância do Nível Energético.

Esse dados estatísticos são importantes para análise dos resultados, pois com os valores de Média e Mediana do Nível Energético da criatura pode-se obter informações referentes à eficiência do controlador no que diz respeito ao controle energético da criatura, ou seja, informações que possibilitam entender se o controlador é ou não eficiente do ponto de vista do controle da energia da criatura. Por outro lado, dados como Desvio Padrão e Variância tem por objetivo fornecer informações sobre a estabilidade do nível energético do agente inteligente (através da análise de dispersão populacional), ou seja, entender se a criatura apresentou uma grande variabilidade energética durante o seu tempo de simulação. Por fim, dados como o Menor Nível Energético/Tempo de Simulação e Máxima Pontuação Obtida provêm informações sobre a eficácia do controlador em conseguir manter a energia do agente até o final da simulação e também entender se o controlador é ou não eficiente na busca de joias e na obtenção de pontos quando comparado com outros controladores.

Tabela 5 – Resultados Obtidos dos experimentos executados com o controlador Reativo.

Experimentos	Máx. Pontuação Obtida	Menor N. Ener./Tempo de Sim.(s)	Média do N. Ener.	Mediana do N. Ener.	Desvio Padrão do N. Ener.	Variância do N. Ener.
1º Exp.	43.0	0/274s	249.08	0.00	324.54	105153.07
2º Exp.	62.0	0/355s	343.84	300.00	349.42	121892.22
3º Exp.	24.0	150.00/244s	500.00	450.00	210.80	44364.69
4º Exp.	36.0	0/205s	179.53	0.00	300.09	89905.61
5º Exp.	12.0	0/325s	283.03	150.00	325.29	105635.42
6º Exp.	0.0	0/303s	291.85	50.00	353.12	124484.28
7º Exp.	56.0	0/365s	324.71	350.00	316.35	99909.44
8º Exp.	86.0	0/364s	340.52	300.00	331.84	109935.01
9º Exp.	0.0	0/307s	243.93	100.00	279.09	77762.62
10º Exp.	42.0	0/244s	199.58	0.00	305.85	93390.01
Média	36.1	-/-	295.60	170.00	309.64	97243.24

Do ponto de vista do nível energético, pode-se perceber que o controlador moti-

Tabela 6 – Resultados Obtidos dos experimentos executados com o controlador com a arquitetura JSOAR.

Experimentos	Máx. Pontuação Obtida	Menor N. Ener./Tempo de Sim.(s)	Média do N. Ener.	Mediana do N. Ener.	Desvio Padrão do N. Ener.	Variância do N. Ener.
1º Exp.	30.0	0/359s	242.43	200.00	243.58	59231.37
2º Exp.	10.0	0/390s	317.39	350.00	286.34	81856.57
3º Exp.	55.0	0/469s	370.80	450.00	242.64	58777.07
4º Exp.	21.0	0/279s	219.13	0.00	270.57	73086.44
5º Exp.	0.0	0/220s	207.65	0.00	329.77	108568.71
6º Exp.	0.0	0/200s	166.89	0.00	279.26	77855.38
7º Exp.	22.0	0/150s	101.41	0.00	209.15	43671.04
8º Exp.	14.0	0/179s	134.69	0.00	243.03	58967.00
9º Exp.	44.0	0/399s	318.30	400.00	263.47	69298.95
10º Exp.	94.0	0/451s	355.66	450.00	244.63	59743.37
Média	29.0	-/-	243.43	185.00	261.24	69105.59

Tabela 7 – Resultados Obtidos dos experimentos executados com o controlador com a arquitetura CLARION.

Experimentos	Máx. Pontuação Obtida	Menor N. Ener./Tempo de Sim.(s)	Média do N. Ener.	Mediana do N. Ener.	Desvio Padrão do N. Ener.	Variância do N. Ener.
1º Exp.	49.0	0/279s	221.30	0.00	277.14	76676.19
2º Exp.	55.0	0/481s	389.02	450.00	248.65	61726.32
3º Exp.	22.0	0/458s	372.71	450.00	261.92	68489.98
4º Exp.	48.0	0/270s	204.58	0.00	266.35	70823.49
5º Exp.	99.0	0/597s	512.65	500.00	169.41	28650.41
6º Exp.	0.0	0/180s	175.62	0.00	272.75	74268.54
7º Exp.	27.0	0/168s	101.16	0.00	198.51	39341.41
8º Exp.	14.0	0/229s	193.43	0.00	293.19	85817.87
9º Exp.	0.0	50.00/369s	321.55	350.00	302.53	91370.15
10º Exp.	52.0	0/547s	498.84	500.00	182.14	33118.44
Média	36.6	-/-	299.08.	225.00	247.26	63028.32

vacional conseguiu manter a energia do agente inteligente até o fim da simulação em oito dos dez experimentos. Portanto, os dois únicos experimentos onde o agente ficou inativo foram o quarto (com 0 no instante 484 segundos) e o nono (com 0 no instante 555 segundos). Além disso, se levarmos em conta dados estatísticos como Média, Mediana, Desvio Padrão e Variância, nota-se que o controlador motivacional obteve os melhores resultados

Tabela 8 – Resultados Obtidos dos experimentos executados com o controlador com a arquitetura LIDA.

Experimentos	Máx. Pontuação Obtida	Menor N. Ener./Tempo de Sim.(s)	Média do N. Ener.	Mediana do N. Ener.	Desvio Padrão do N. Ener.	Variância do N. Ener.
1º Exp.	53.0	0/404s	277.29	250.00	257.40	66143.89
2º Exp.	26.0	0/365s	284.36	300.00	271.35	73507.45
3º Exp.	56.0	0/384s	341.26	400.00	304.39	92498.57
4º Exp.	84.0	0/495s	381.03	450.00	257.42	66154.34
5º Exp.	28.0	50/595s	488.85	500.00	171.20	29260.08
6º Exp.	30.0	0/455s	382.70	450.00	276.90	76547.48
7º Exp.	40.0	0/385s	302.25	350.00	264.70	69949.20
8º Exp.	69.0	0/474s	354.24	400.00	244.75	59803.13
9º Exp.	42.0	300/533s	553.16	550.00	128.96	16603.98
10º Exp.	70.0	0/424s	342.35	400.00	265.50	70374.03
Média	49.8	-/-	370.45	405.00	244.26	62084.21

Tabela 9 – Resultados Obtidos dos experimentos executados com o controlador com o Subsistema Motivacional.

Experimentos	Máx. Pontuação Obtida	Menor N. Ener./Tempo de Sim.(s)	Média do N. Ener.	Mediana do N. Ener.	Desvio Padrão do N. Ener.	Variância do N. Ener.
1º Exp.	40.0	50.00/544s	459.32	450.00	167.05	27858.27
2º Exp.	66.0	250.00/285s	599.42	600.00	147.46	21709.31
3º Exp.	38.0	250.00/574s	606.57	600.00	143.56	20576.60
4º Exp.	51.0	0/484s	388.27	400.00	270.14	72853.25
5º Exp.	66.0	300.00/525s	591.26	600.00	131.10	17157.47
6º Exp.	135.0	50.00/548s	476.54	500.00	189.03	35672.55
7º Exp.	116.0	200.00/135s	569.88	550.00	182.30	33177.86
8º Exp.	114.0	150.00/475s	510.40	500.00	188.93	35636.45
9º Exp.	56.0	0/555s	489.35	500.00	225.73	50868.30
10º Exp.	91.0	250.00/144s	606.16	600.00	133.56	17807.36
Média	77.3	-/-	535.36	530.00	177.88	33331.17

em comparação com os demais controladores. Sendo assim, o controlador motivacional obteve o maior valor das média do nível energético de todos os experimentos, com o valor "535.36", e também com a maior mediana energética, com o valor "530.00". Com este valores, pode-se afirmar que o subsistema motivacional foi a tecnologia que obteve a maior eficiência energética, dentre todas as estudadas. Além disso, com o valor da média dos

desvios padrão, de “177.88”, e variância de “33331.17”, também pode-se afirmar que o controlador motivacional foi o controlador que manteve a energia da criatura mais estável, tendo em vista que é o menor valor de desvio padrão e variância dentre todos os controladores aqui desenvolvidos e experimentados. Do ponto de vista da eficácia do controlador motivacional em manter a criatura ativa, pode-se dizer que, em comparação aos outros controladores, o controlador motivacional foi o que obteve a melhor acurácia. de “80%”, ou seja, em oito de dez experimentos o controlador conseguiu manter o nível energético acima de zero até o fim das simulações.

Seguindo essa linha de análise, se quisermos elaborar um ranking dos demais controladores pode-se dizer que o segundo melhor controlador é o controlador LIDA. Este controlador, apesar de somente conseguir manter o nível energético da criatura até o fim da simulação em duas oportunidades (no quinto e no nono experimento, obtendo a acurácia de “20%”), ele foi o segundo melhor controlador, com relação ao controle do nível energético, obtendo um valor para as médias do nível energético igual a “370.45” e a média da mediana com um valor “405.00”. Também pode-se afirmar que o controlador LIDA foi o segundo melhor controlador que manteve o nível energético estável, com os valores de média dos desvios padrão igual a “224.26” e variância de “62187.69”.

Em terceiro lugar, dentro da perspectiva de eficiência energética, está o controlador CLARION, com a média do nível energético igual a “299.08” e a média da mediana com valor de “225.00”. O controlador CLARION também se mantém em terceiro lugar no que diz respeito ao controle da estabilidade energética. com um desvio padrão médio igual a “247.26” e variância média igual a “62084.21”. Entretanto, o controlador CLARION somente conseguiu terminar um único experimento (nono experimento), sendo assim, este controlador empata com o controlador Reativo no quesito de eficácia energética obtendo uma acurácia de “10%”. Já em quarto lugar, no quesito de eficiência energética, está o controlador JSOAR com a média do nível energético igual a “243.43” e mediana média com valor de “185.00”. Por outro lado, o controlador JSOAR está em quinto lugar na questão da eficácia energética, pois ele não conseguiu concluir nenhum experimento com a energia da criatura acima de zero. E por fim, o controlador Reativo ficou em último lugar no que diz respeito ao controle energético e em quarto lugar no quesito de eficácia, concluindo o terceiro experimento com “150.00” no instante “244” segundos.

Embora o controle energético se mostre um critério importante para a avaliação de desempenho dos controladores, é necessário igualmente avaliarmos a Máxima Pontuação Obtida. O controlador que obteve a maior média de máxima pontuação foi o controlador Motivacional com “77.3” pontos, em seguida, está o controlador LIDA com “49.8” pontos, em terceiro lugar o controlador CLARION com “36.6”, em quarto lugar o controlador Reativo com “36.1” e, por fim, em último lugar ficou o controlador JSOAR com uma média de pontuação máxima de “29.0”. As Tabelas 10a, 10b e 10c fazem um resumo

dos rankings de melhores controladores levando em consideração a eficiência e eficácia energética, juntamente com a máxima pontuação obtida.

Tabela 10 – Rankings de eficiência (a) e eficácia (b) energética, juntamente com o ranking de pontuação obtida (c) dos controladores Motivacional, LIDA, CLARION, JSOAR e Reativo.

(a)

Posição	Controlador
1º	Motivacional
2º	LIDA
3º	CLARION
4º	JSOAR
5º	Reativo

(b)

Posição	Controlador
1º	Motivacional
2º	LIDA
3º	*CLARION
3º	*Reativo
4º	JSOAR

(c)

Posição	Controlador
1º	Motivacional
2º	LIDA
3º	CLARION
4º	Reativo
5º	JSOAR

5.1 Análise das Estruturas Internas do Controlador Motivacional.

Nosso modelo de subsistema motivacional é bastante sofisticado, incluindo sub-estruturas internas que, dentre outras coisas, representam motivações, apreciações primárias, goals, emoções e planos do agente inteligente, resultando em um comportamento que incorpora todas essas variáveis. Nessa seção, apresentamos como esses diferentes parâmetros variaram ao longo da simulação.

Como discutimos nos capítulos 3 e 4, o subsistema motivacional pode ser dividido em duas partes o Subsistema Motivacional de Alto Nível (SMAN) e o Subsistema Motivacional de Subsunção (SMS). Essa divisão foi inspirada na teoria dos processos duais (EVANS; STANOVICH, 2013; KAHNEMAN, 2011), que preconizam a mente humana como sendo constituída por dois sub-sistemas distintos, um deles rápido, paralelo, reativo e limitado, destinado a tratar situações *default* (muitas vezes associado com o comportamento inconsciente) - chamado de modo livre de *system1* e o outro mais lento, serial, deliberativo e sofisticado (muitas vezes associado com o comportamento consciente do ser humano), destinado a tratar situações de exceção, que podem demandar um processamento mais elaborado, ou que exigem algum tipo de aprendizado, chamado também de modo livre de *system2*. Dessa forma o SMS estaria associado ao que os proponentes da teoria dos processos duais chamariam de *system1* e o SMAN ao *system2*. Dessa forma, o principal objetivo do SMAN seria realizar o processamento de "Exceções", ou seja, situações em que o SMS não provê uma solução adequada na interação do agente inteligente com o ambiente. SMAM e SMS devem portanto interagir entre si, sendo que nenhum deles deve predominar em todas as situações. Espera-se que em uma grande maioria de casos, o SMS seja suficiente para tomar uma decisão adequada. Entretanto, em diversas situações excepcionais onde um processamento mais elaborado pode ser necessário, o SMAN

deve preponderar, e assumir o controle do agente. Para exemplificar como as estruturas do SMAN e do SMS interferem no modo de operação do Sistema Motivacional como um todo, mostramos na Figura 41 as ativações das subestruturas de: Apreciação Primária, Humor, Distorções Cognitivas dos Drives Emocionais e Ativações dos Drives que foram obtidas na execução do sexto experimento do controlador motivacional.

Como o SMAN é destinado a realizar processamentos de exceção, em nosso caso ele foi desenhado para resolver o desafio da busca de joias e obtenção de pontos da criatura, tendo em vista a obtenção da máxima pontuação possível (através da troca de joias por pontos no "Delivery Spot") a partir da geração e execução de planos. A Tabela 11 informa a quantidade de planos gerados e executados pelo SMAN em todos os experimentos.

Com base nos dados fornecidos pela Tabela 11 pode-se observar que em 50% dos casos, o sistema de planejamento do controlador motivacional gerou e executou planos. Dois dos experimentos foram os mais efetivos na obtenção de pontos, o sexto e o oitavo experimento. Entre todos os experimentos, o sexto apresentou a maior pontuação. Já o oitavo, foi o terceiro melhor experimento na obtenção de pontos. Não somente para esses casos, mas de uma forma geral é possível perceber que os experimentos que obtiveram as pontuações mais elevadas, em sua maioria, foram aqueles que geraram e executaram planos. A Tabela 11 demonstra que quanto mais planos foram executados pelo controlador

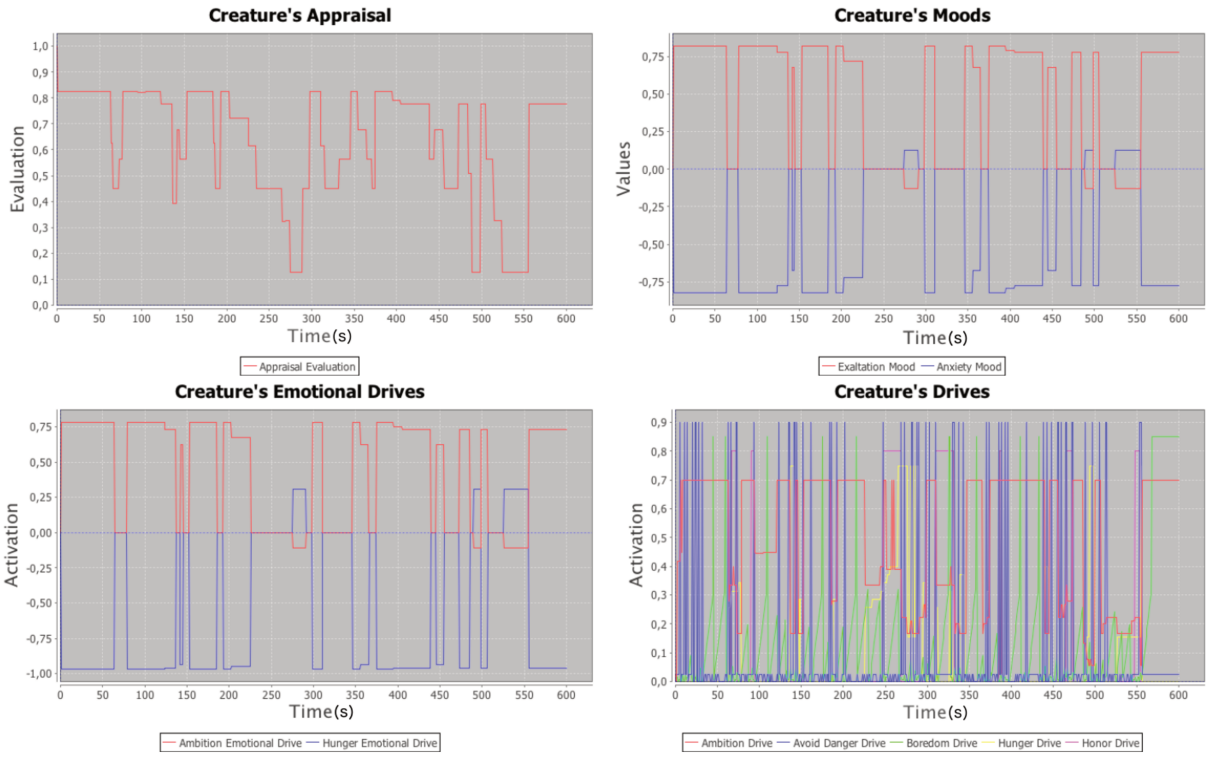

Figura 41 – Níveis de Ativação das seguintes variáveis do agente inteligente durante a execução do sexto experimento do controlador motivacional: Apreciação Primária (SMAN), Humores, Distorções Cognitivas das Emoções e Drives gerais (SMS).

Tabela 11 – Número de Planos gerados e executados no experimentos do controlador motivacional.

Posição	Experimento	Máximo de Pontos Obtidos	Número de Planos Gerados e Executados
1º	6º Exp.	135.0	3
2º	7º Exp.	116.0	0
3º	8º Exp.	114.0	3
4º	10º Exp.	91.0	2
5º*	2º Exp.	66.0	1
5º*	5º Exp.	66.0	0
6º	9º Exp.	56.0	1
7º	4º Exp.	51.0	0
8º	1º Exp.	40.0	0
9º	3º Exp.	38.0	0

motivacional, mais Leaflets foram completados e também mais pontos foram adquiridos.

Em complemento aos experimentos propostos na seção 4.1, realizamos um experimento adicional para tentar mensurar a influência particular de SMAN ou SMS na tomada de decisão do controlador motivacional. Este experimento consiste no mesmo experimento anteriormente executado, inclusive com as mesmas condições de tempo de simulação, número de joias/comidas geradas por minuto e números de obstáculos. Entretanto, ao invés de medir a eficiência/eficácia do controlador motivacional em capturar joias e manter seu nível energético, neste caso estávamos interessados em analisar em que situações SMS ganhava o controle do sistema e em que situações SMAN o fazia. A Figura 42 ilustra as porcentagens da interferência de cada um dos subsistemas motivacionais durante o tempo de simulação, em um dos casos simulados.

A Figura demonstra claramente a diferença da influência dos subsistemas. Por exemplo, SMS tomou as decisões da criatura em “87%” do tempo de simulação, ou seja, durante “522” segundos foi a subsunção motivacional (através dos drives) que determinou as decisões do agente cognitivo. Em contrapartida, em “13%” (ou em “78” segundos) do tempo de simulação, o comportamento da criatura foi orientado e comandado pelos planos gerados e selecionados pelos Codelets de Planejamento e Seleção de Planos. Neste experimento, o SMAN gerou e executou apenas três planos. Portanto, o tempo de influência de cada subsistema depende diretamente dos planos gerados no subsistema de alto nível. Isto é, quanto mais planos forem gerados mais o SMAN influenciará nas ações da criatura e menos o SMS atuará no comportamento. Isso acontece justamente porque o SMAN é prioritário no que diz respeito ao controle da criatura. Em outras palavras quando um plano é gerado, o mesmo domina o controle da criatura até o final da execução do plano,

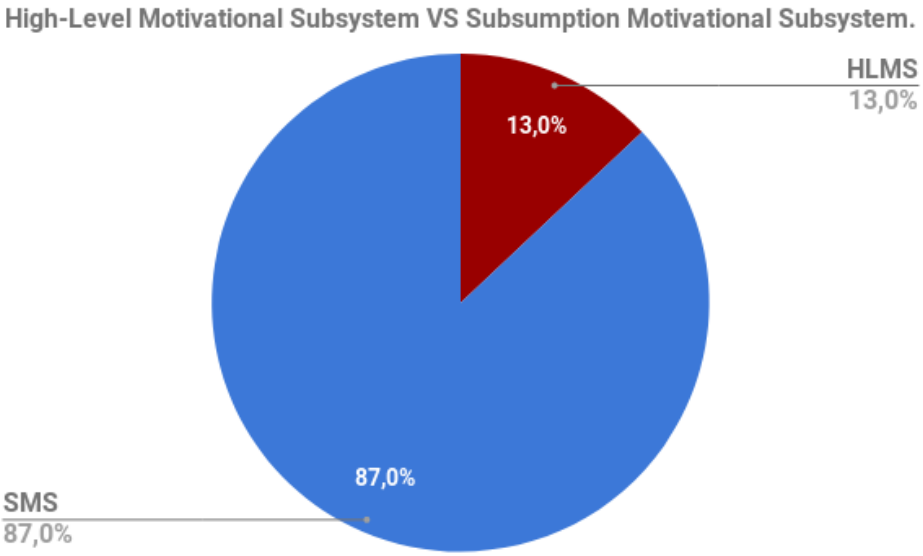

Figura 42 – HLMS vs SMS.

ou seja, quando seu objetivo é alcançado.

6 Conclusão

O presente trabalho envolveu o estudo de sistemas motivacionais em arquiteturas cognitivas e a proposição e desenvolvimento de um subsistema motivacional, com contribuições aos mecanismos estudados, de forma que os mesmos pudessem ser disponibilizados no *toolkit* CST. Dessa forma, desenvolvemos tanto um modelo conceitual de subsistema motivacional, quanto uma implementação computacional desse modelo. Para validar nosso modelo e implementação, o subsistema motivacional foi testado em um conjunto de experimentos, que logrou resultados que consideramos satisfatórios, superando um controlador puramente reativo e também controladores implementados com as arquiteturas cognitivas SOAR, CLARION e LIDA, arquiteturas muito populares na comunidade de arquiteturas cognitivas. O desenvolvimento dos controladores baseados nas ACs aqui utilizadas foi realizado em conformidade com as melhores práticas empregadas nos tutoriais e manuais de cada AC. Links dos manuais e tutoriais das ACs estão disponibilizados no capítulo 7 (Anexo).

Apesar dos bons resultados obtidos a partir da execução dos experimentos, seria por demais pretensioso generalizarmos e afirmar que o nosso modelo de subsistema motivacional é superior aos modelos disponibilizados pelas demais ACs. Tal conclusão seria desmesurada, pois nosso modelo de subsistema motivacional não foi testado em outras aplicações para comprovar tal superioridade. Além disso, apesar de nossos melhores esforços em construir controladores utilizando as demais ACs que explorassem adequadamente suas potencialidades, não podemos garantir que nossa versão de controlador utilizando cada AC é aquele que poderia proporcionar o melhor desempenho, diante das potencialidades de cada AC. Dessa forma, o que podemos afirmar é que, de acordo com o contexto de simulação do WS3D proposto por este trabalho, e as implementações que realizamos de controladores utilizando as demais ACs, visando explorar as melhores potencialidades de cada uma delas, mas sujeitas a uma possível inabilidade de nossa parte em explorar adequadamente esse potencial, o controlador baseado em nosso modelo motivacional leva vantagens sobre os demais controladores testados no que diz respeito aos quesitos de eficiência/eficácia energética, e também na máxima pontuação obtida.

Vale a pena ressaltar que nosso modelo motivacional, agora inserido no CST é bastante poderoso, incorporando múltiplas facetas relacionadas ao fenômeno motivacional em seres vivos, e aproveitando diversas ideias apresentadas em outras ACs de maneira integrada e configurável. Dessa forma, pode ser utilizado em diferentes tipos de controlador, permitindo ao usuário explorar diferentes perspectivas do fenômeno motivacional, podendo ser utilizado em diversas área que necessitem de um sistema de controle, como por exemplo: Controle de tráfego urbano, Controladores de personagens em jogos digi-

tais, Controladores para robôs baseado em comportamento humano/animal, Aplicações voltadas para a área de Cidades Inteligentes e Internet das Coisas, etc.

Nosso modelo de Subsistema Motivacional foi inspirado tanto no módulo MS (Motivational System) da arquitetura CLARION, como em particularidades da implementação de motivações e emoções na arquitetura LIDA. Entretanto, não nos limitamos a reproduzir o mecanismo motivacional desenvolvido nessas arquiteturas, mas trouxemos contribuições próprias no mecanismo que concebemos para integrar o CST. Dentre os aperfeiçoamentos que desenvolvemos, está nosso mecanismo de urgência, que em nossa opinião é mais flexível que o mecanismo implementado na arquitetura CLARION. Embora o sistema motivacional da arquitetura CLARION seja capaz de diferenciar drives primários de baixo e alto nível, permitindo que um Drive seja prioritário sobre outros, o SMS através dos parâmetros de prioridade e limite de urgência, proporcionou um mecanismo melhor, estendendo essa compartimentalização entre 2 níveis para toda uma hierarquia possível, permitindo, em tese, que toda uma pirâmide de necessidades (como, por exemplo, na teoria de Maslow) possa ser modelada (enquanto em CLARION apenas 2 níveis são possíveis). Além disso, nosso Subsistema Motivacional também permite que a distorção cognitiva emocional seja aplicada através da avaliação primária juntamente com os estados de humor que uma criatura possa apresentar. Em nossos experimentos, com essas estruturas, a criatura conseguiu manter o seu nível energético até o fim, em oito dos dez experimentos executados. Consequentemente, essas estruturas deram suporte na captura de joias no ambiente, enquanto o sistema de planejamento planejava e executava sequências de ações visando atingir os objetivos de pontuação da criatura. De modo geral, pode-se observar que conforme os Codelets de Planejamento e Seleção de Planos geram e executam planos, a criatura consegue atingir a meta de uma maneira mais eficiente, que neste caso, fez com o agente inteligente obtivesse as melhores pontuações quando comparados com os demais experimentos. Portanto, concluímos que o modelo de Subsistema Motivacional foi satisfatoriamente validado e pode ser utilizado em aplicações gerais envolvendo agentes inteligentes.

6.1 Trabalhos Futuros

Dentre os trabalhos futuros a serem realizados estão:

- Desenvolver o Subsistema de Aprendizado e Imaginação integrado com o Subsistema Motivacional.
- Desenvolver o Subsistema de Memórias Episódicas.
- Desenvolver o Subsistema de Metacognição.
- Desenvolver o Subsistema de Consciência integrado com o Sistema de Planejamento.

Com estes subsistemas sugeridos como novos módulos a serem inseridos no CST, novas habilidades poderão ser adicionadas aos agentes cognitivos, que deverão complementar as funcionalidades do subsistema motivacional proposto por este trabalho. Por exemplo, com Subsistema de Aprendizado e Imaginação, acredita-se que a criatura possa ganhar a habilidade de aprender e regular os parâmetros de limite de urgência e prioridade de seus drives, fazendo com que eles se tornem dinâmicos conforme a criatura experiencia o ambiente à sua volta. Já com o Subsistema de Memórias Episódicas, o agente cognitivo poderá adquirir a habilidade de recordar quais foram as suas ultimas decisões (ou planos) executadas e onde o mesmo obteve os melhores resultados. Por outro lado, com o mecanismo do Subsistema de Metacognição, a criatura poderá analisar e reavaliar suas decisões momentâneas, tendo em vista alcançar a otimização de todo o sistema. E por fim, o desenvolvimento de um Subsistema de Consciência (no sentido de *system 2* na teoria do processos duais) permitirá que o agente cognitivo integre e oriente todos os outros subsistemas através desse mecanismo, ou seja, este mecanismo organizará e comandará os processos dos subsistemas adjacentes, agindo com se fosse um "maestro" monitorando e corrigindo um grupo de músicos em uma orquestra.

Referências

ANDERSON, J.; LEBIERE, C. The atomic components of thought lawrence erlbaum. *Mathway, NJ*, 1998. Citado na página 51.

ANDERSON, J. R. The architecture of cognition. Citeseer, 1983. Citado na página 17.

ANDERSON, J. R. A spreading activation theory of memory. *Journal of verbal learning and verbal behavior*, Elsevier, v. 22, n. 3, p. 261–295, 1983. Citado na página 17.

ANDERSON, J. R. Rules of the mind. ERIC, 1993. Citado na página 24.

ANDERSON, J. R.; BOWER, G. H. Human associative memory. VH Winston & Sons, 1973. Citado na página 17.

ARKIN, R. C. *Behavior-based robotics*. [S.l.]: MIT press, 1998. Citado 2 vezes nas páginas 39 e 40.

ARRABALES, R.; LEDEZMA, A.; SANCHIS, A. Cera-cranium: A test bed for machine consciousness research. 2009. Citado na página 26.

ARRABALES, R.; LEDEZMA, A.; SANCHIS, A. A cognitive approach to multimodal attention. Red de Agentes Fisicos, 2009. Citado na página 26.

ARRABALES, R.; LEDEZMA, A.; SANCHIS, A. Towards conscious-like behavior in computer game characters. In: IEEE. *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*. [S.l.], 2009. p. 217–224. Citado na página 26.

ARRABALES, R.; LEDEZMA, A.; SANCHIS, A. Simulating visual qualia in the cera-cranium cognitive architecture. In: *From Brains to Systems*. [S.l.]: Springer, 2011. p. 223–238. Citado na página 26.

BAARS, B. J. In the theatre of consciousness. global workspace theory, a rigorous scientific theory of consciousness. *Journal of Consciousness Studies*, Imprint Academic, v. 4, n. 4, p. 292–309, 1997. Citado na página 27.

BACH, J. The micropsi agent architecture. In: CITESEER. *Proceedings of ICCM-5, international conference on cognitive modeling, Bamberg, Germany*. [S.l.], 2003. p. 15–20. Citado na página 27.

BACH, J. Representations for a complex world: Combining distributed and localist representations for learning and planning. In: *Biomimetic Neural Learning for Intelligent Robots*. [S.l.]: Springer, 2005. p. 265–280. Citado na página 27.

BACH, J. Micropsi 2: the next generation of the micropsi framework. In: SPRINGER. *International Conference on Artificial General Intelligence*. [S.l.], 2012. p. 11–20. Citado na página 27.

BACH, J. Modeling motivation in micropsi 2. In: SPRINGER. *International Conference on Artificial General Intelligence*. [S.l.], 2015. p. 3–13. Citado na página 27.

BACH, J.; VUINE, R. Designing agents with micropsi node nets. In: SPRINGER. *Annual Conference on Artificial Intelligence*. [S.l.], 2003. p. 164–178. Citado na página 27.

BATES, J. *et al.* The role of emotion in believable agents. *Communications of the ACM*, Citeseer, v. 37, n. 7, p. 122–125, 1994. Citado na página 26.

BERRIDGE, K. C.; ALDRIDGE, J. W. Special review: Decision utility, the brain, and pursuit of hedonic goals. *Social cognition*, Guilford Press, v. 26, n. 5, p. 621–646, 2008. Citado na página 32.

BREAZEAL, C. *et al.* A motivational system for regulating human-robot interaction. In: *Aaai/iaai*. [S.l.: s.n.], 1998. p. 54–61. Citado na página 20.

BROOKS, R. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, IEEE, v. 2, n. 1, p. 14–23, 1986. Citado na página 39.

BUCK, R. *Human motivation and emotion*. [S.l.]: John Wiley & Sons, 1988. Citado na página 25.

BUDAKOVA, D.; DAKOVSKI, L. Computer model of emotional agents. *Lecture Notes in Computer Science*, Springer-Verlag, v. 4133, p. 450, 2006. Citado na página 26.

CASTRO, E. C.; GUDWIN, R. R. A scene-based episodic memory system for a simulated autonomous creature. *International Journal of Synthetic Emotions*, IGI Global, v. 4, n. 1, p. 32–64, jan 2013. Disponível em: <https://doi.org/10.4018%2Fjse.2013010102>. Citado na página 56.

CAñAMERO, D. A hormonal model of emotions for behavior control. *VUB AI-Lab Memo*, Citeseer, v. 2006, 1997. Citado 4 vezes nas páginas 20, 26, 44 e 74.

CAñAMERO, D. Issues in the design of emotional agents. In: *Emotional and Intelligent: The Tangled Knot of Cognition. Papers from the 1998 AAAI Fall Symposium*. [S.l.: s.n.], 1998. p. 49–54. Citado na página 26.

CLEEREMANS, A.; DESTREBECQZ, A.; BOYER, M. Implicit learning: News from the front. *Trends in cognitive sciences*, Elsevier, v. 2, n. 10, p. 406–416, 1998. Citado na página 29.

CONNELL, J. H. A behavior-based arm controller. *IEEE Transactions on Robotics and Automation*, IEEE, v. 5, n. 6, p. 784–791, 1989. Citado na página 39.

DAMASIO, A. R. Descartes' error: Emotion, rationality and the human brain. 1994. Citado na página 26.

DAMASIO, A. R. *The feeling of what happens: Body and emotion in the making of consciousness*. [S.l.]: Houghton Mifflin Harcourt, 1999. Citado na página 26.

EPSTEIN, A. N. Instinct and motivation as explanations for complex behavior. In: *The physiological mechanisms of motivation*. [S.l.]: Springer, 1982. p. 25–58. Citado na página 52.

EVANS, J. S. B.; STANOVICH, K. E. Dual-process theories of higher cognition advancing the debate. *Perspectives on psychological science*, Sage Publications, v. 8, n. 3, p. 223–241, 2013. Citado na página 99.

FRANKLIN, S. *Artificial minds*. [S.l.]: MIT press, 1997. Citado na página 31.

FRANKLIN, S.; KELEMEN, A.; MCCAULEY, L. Ida: A cognitive agent architecture. In: IEEE. *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*. [S.l.], 1998. v. 3, p. 2646–2651. Citado 2 vezes nas páginas 27 e 38.

FRANKLIN, S.; MADL, T.; D'MELLO, S.; SNAIDER, J. Lida: A systems-level architecture for cognition, emotion, and learning. *IEEE Transactions on Autonomous Mental Development*, IEEE, v. 6, n. 1, p. 19–41, 2014. Citado 2 vezes nas páginas 27 e 38.

FRANKLIN, S.; MADL, T.; STRAIN, S.; FAGHIHI, U.; DONG, D.; KUGELE, S.; SNAIDER, J.; AGRAWAL, P.; CHEN, S. A lida cognitive model tutorial. *Biologically Inspired Cognitive Architectures*, Elsevier, v. 16, p. 105–130, 2016. Citado 3 vezes nas páginas 21, 27 e 38.

FRANKLIN, S.; Patterson Jr., F. The lida architecture: Adding new modes of learning to an intelligent, autonomous, software agent. *pat*, v. 703, p. 764–1004, 2006. Citado na página 27.

FRóES, E. de M. *Building a Motivational Subsystem For Cognitive System Toolkit. XIII Simpósio Brasileiro de Automação Inteligente (SBAI-2017)*. 2017. Citado 4 vezes nas páginas , 41, 42 e 57.

GUDWIN, R.; PARAENSE, A.; PAULA, S. M. de; FRóES, E.; GIBAUT, W.; CASTRO, E.; FIGUEIREDO, V.; RAIZER, K. The multipurpose enhanced cognitive architecture (meca). *Biologically Inspired Cognitive Architectures*, v. 22, n. Supplement C, p. 20 – 34, 2017. ISSN 2212-683X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2212683X17301068>. Citado 2 vezes nas páginas e 77.

HAMADI, Y.; JABBOUR, S.; SAÏS, L. Learning for dynamic subsumption. *International Journal on Artificial Intelligence Tools*, World Scientific, v. 19, n. 04, p. 511–529, 2010. Citado na página 40.

HECKEL, F. W.; YOUNGBLOOD, G. M. Multi-agent coordination using dynamic behavior-based subsumption. In: *AIIDE*. [S.l.: s.n.], 2010. Citado na página 40.

HOFSTADTER, D. R.; MITCHELL, M. *et al.* The copycat project: A model of mental fluidity and analogy-making. *Advances in connectionist and neural computation theory*, v. 2, n. 31-112, p. 29–30, 1994. Citado 2 vezes nas páginas 27 e 38.

HULL, C. L. Principles of behavior: An introduction to behavior theory. Appleton-Century, 1943. Citado 5 vezes nas páginas 19, 24, 25, 41 e 51.

KAHNEMAN, D. *Thinking, fast and slow*. [S.l.]: Farrar, Strauss and Giroux, New York / Macmillan, 2011. ISBN 1846140552. Citado na página 99.

LAIRD, J. E. Extending the soar cognitive architecture. *Frontiers in Artificial Intelligence and Applications*, IOS Press, v. 171, p. 224, 2008. Citado 3 vezes nas páginas , 28 e 37.

LAIRD, J. E. *The Soar cognitive architecture*. [S.l.]: MIT Press, 2012. Citado 2 vezes nas páginas 28 e 47.

LAIRD, J. E.; CONGDON, C. B. The soar user's manual, version 9.4.0. *Ann Arbor, MI: Electrical Engineering and Computer Science Department, U. of Michigan*, 2014. Citado 2 vezes nas páginas e 34.

LAIRD, J. E.; KINKADE, K. R.; MOHAN, S.; XU, J. Z. Cognitive robotics using the soar cognitive architecture. *Cognitive Robotics AAAI Technical Report, WS-12*, v. 6, 2012. Citado na página 28.

LAIRD, J. E.; NEWELL, A.; ROSENBLOOM, P. S. Soar: An architecture for general intelligence. *Artificial intelligence*, Elsevier, v. 33, n. 1, p. 1–64, 1987. Citado 3 vezes nas páginas 17, 21 e 28.

LAIRD, J. E.; ROSENBLOOM, P. S.; NEWELL, A. Towards chunking as a general learning mechanism. In: AAAI PRESS. *Proceedings of the Fourth AAAI Conference on Artificial Intelligence*. [S.l.], 1984. p. 188–192. Citado 3 vezes nas páginas 17, 21 e 28.

LANGLEY, P.; LAIRD, J. E.; ROGERS, S. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, Elsevier, v. 10, n. 2, p. 141–160, 2009. Citado na página 17.

LAZARUS, R. S. *Emotion and adaptation*. [S.l.]: Oxford University Press on Demand, 1991. Citado 5 vezes nas páginas 31, 33, 48, 66 e 74.

LUCENTINI, D. F. *Uma Comparação entre Arquiteturas Cognitivas: Análise Teórica e Prática*. [S.l.]: Dissertação de Mestrado, DCA-FEEC-UNICAMP, 2017. Citado 5 vezes nas páginas , 34, 37, 56 e 58.

MASLOW, A. H. A theory of human motivation. *Psychological review*, American Psychological Association, v. 50, n. 4, p. 370, 1943. Citado na página 24.

MASLOW, A. H. *Motivation and personality*. [S.l.: s.n.], 1970. v. 2. Citado na página 25.

MCCALL, R. J. *Fundamental motivation and perception for a systems-level cognitive architecture*. Tese (Doutorado) — THE UNIVERSITY OF MEMPHIS, 2014. Citado 6 vezes nas páginas , 27, 31, 32, 43 e 48.

MCCLELLAND, D. *Personality. New York: William Sloane Associates*. [S.l.]: Dryden Press, 1951. Citado na página 25.

MCDOUGALL, W. *An introduction to social psychology*. [S.l.]: Psychology Press, 2015. Citado na página 25.

MEYER, J.-A.; WILSON, S. W. From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior (complex adaptive systems). 1990. Citado na página 17.

MEYER, J.-J. Reasoning about emotional agents. *Artificial Intelligence Preprint Series*, v. 44, 2008. Citado na página 26.

MURRAY, H. A. Explorations in personality. Oxford Univ. Press, 1938. Citado na página 25.

NAKASHIMA, H.; NODA, I. Dynamic subsumption architecture for programming intelligent agents. In: CITESEER. *icmas*. [S.l.], 1998. p. 190–197. Citado na página 40.

O'REILLY, R. C. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural computation*, MIT Press, v. 8, n. 5, p. 895–938, 1996. Citado na página 27.

ORTONY, A.; CLORE, G. L.; COLLINS, A. *The cognitive structure of emotions*. [S.l.]: Cambridge university press, 1990. Citado na página 26.

O'REILLY, R. C.; HAZY, T. E.; HERD, S. A. *The leabra cognitive architecture: how to play 20 principles with nature and win!* [S.l.]: Oxford University Press, 2012. Citado na página 27.

PARAENSE, A. L.; RAIZER, K.; PAULA, S. M. de; ROHMER, E.; GUDWIN, R. R. The cognitive systems toolkit and the cst reference cognitive architecture. *Biologically Inspired Cognitive Architectures*, Elsevier, v. 17, p. 32–48, 2016. Citado 2 vezes nas páginas 38 e 42.

PICARD, R. W.; PICARD, R. *Affective computing*. [S.l.]: MIT press Cambridge, 1997. v. 252. Citado na página 26.

RAIZER, K.; PARAENSE, A. L.; GUDWIN, R. R. A cognitive architecture with incremental levels of machine consciousness inspired by cognitive neuroscience. *International Journal of Machine Consciousness*, World Scientific, v. 4, n. 02, p. 335–352, 2012. Citado na página 38.

REBER, A. S. Implicit learning and tacit knowledge. *Journal of experimental psychology: General*, American Psychological Association, v. 118, n. 3, p. 219, 1989. Citado na página 29.

REILLY, W. S. *Believable Social and Emotional Agents*. [S.l.], 1996. Citado na página 26.

REISS, S. Multifaceted nature of intrinsic motivation: The theory of 16 basic desires. *Review of General Psychology*, Educational Publishing Foundation, v. 8, n. 3, p. 179, 2004. Citado na página 25.

ROSEMAN, I. J.; SMITH, C. A. Appraisal theory. *Appraisal processes in emotion: Theory, methods, research*, Oxford University Press Oxford,, UK, p. 3–19, 2001. Citado na página 48.

SAMSONOVICH, A. V. Toward a unified catalog of implemented cognitive architectures. *BICA*, v. 221, p. 195–244, 2010. Citado na página 17.

SEPTSEAULT, C.; NÉDÉLEC, A. A model of an embodied emotional agent. In: SPRINGER. *International Workshop on Intelligent Virtual Agents*. [S.l.], 2005. p. 498–498. Citado na página 26.

SLOMAN, A. Motives mechanisms and emotions'. emotion and cognition, 1 (3): 21 7-2 3 4, 1 9 8 7. reprinted in m. a. boden (ed), the philosophy of artificial intelligence,'. *Oxford Readings in Philosophy's Series, Oxford University P ress*, v. 2, n. 3, p. 1–24, 1987. Citado na página 25.

SLOMAN, A. Damasio, descartes, alarms and meta-management. In: IEEE. *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*. [S.l.], 1998. v. 3, p. 2652–2657. Citado na página 26.

SLOMAN, A. Architecture-based conceptions of mind. In: *In the Scope of Logic, Methodology and Philosophy of Science*. [S.l.]: Springer, 2002. p. 403–427. Citado na página 17.

SLOMAN, A. *et al.* Beyond shallow models of emotion. *Cognitive Processing*, v. 2, n. 1, p. 177–198, 2001. Citado na página 26.

SUN, R. *Duality of the mind: A bottom-up approach toward cognition*. [S.l.]: Psychology Press, 2001. Citado 2 vezes nas páginas 29 e 51.

SUN, R. A tutorial on clarion 5.0. *Cognitive Science Department, Rensselaer Polytechnic Institute*, 2003. Citado 4 vezes nas páginas 21, 24, 26 e 30.

SUN, R. Desiderata for cognitive architectures. *Philosophical Psychology*, Taylor & Francis, v. 17, n. 3, p. 341–373, 2004. Citado 2 vezes nas páginas 17 e 26.

SUN, R. Cognitive architectures and multi-agent social simulation. In: SPRINGER. *Pacific Rim International Workshop on Multi-Agents*. [S.l.], 2005. p. 7–21. Citado 3 vezes nas páginas , 28 e 30.

SUN, R. The importance of cognitive architectures: An analysis based on clarion. *Journal of Experimental & Theoretical Artificial Intelligence*, Taylor & Francis, v. 19, n. 2, p. 159–193, 2007. Citado 2 vezes nas páginas 17 e 26.

SUN, R. The motivational and metacognitive control in clarion. *Modeling integrated cognitive systems*, p. 63–75, 2007. Citado na página 29.

SUN, R. Motivational representations within a computational cognitive architecture. *Cognitive Computation*, Springer, v. 1, n. 1, p. 91–103, 2009. Citado 10 vezes nas páginas 20, 24, 25, 26, 28, 29, 30, 43, 51 e 52.

SUN, R.; PETERSON, T.; MERRILL, E. Bottom-up skill learning in reactive sequential decision tasks. In: LAWRENCE ERLBAUM ASSOCIATES INC HILLSDALE, NJ. *Proceedings of 18th cognitive science society conference*. [S.l.], 1996. p. 684–690. Citado na página 26.

TOATES, F. M. *Motivational systems*. [S.l.]: CUP Archive, 1986. Citado 2 vezes nas páginas 20 e 25.

TYRRELL, T. The use of hierarchies for action selection. *Adaptive Behavior*, Sage Publications, v. 1, n. 4, p. 387–420, 1993. Citado na página 25.

7 Anexo

O código-fonte da versão mais recente do projeto CST, já incluindo a implementação do Subsistema Motivacional desenvolvido neste trabalho, pode ser obtido no repositório disponível em:

- https://github.com/CST-Group/cst.git.

Detalhes sobre o projeto CST podem ser encontrados em:

- http://cst.fee.unicamp.br/.

Todos os controladores aqui desenvolvidos juntamente com os experimentos executados podem ser encontrados nos links:

- https://github.com/CST-Group/MotivationalSystemWithWorldServer3D.git.
- https://github.com/CST-Group/ReactiveSystemWithWorldServer3D.git.
- https://github.com/eduardofroes/LidaWithWorldServer3D.git.
- https://github.com/eduardofroes/JSoarWithWorldServer3D.git.
- https://github.com/eduardofroes/ClarionControllerWithWorldServer3D.git.

Informações e Tutoriais sobre as Arquiteturas Cognitivas JSOAR/SOAR, CLARION e LIDA podem ser encontradas nos links:

- http://soartech.github.io/jsoar/.
- https://soar.eecs.umich.edu/articles/downloads/soar-suite/228-soar-tutorial-9-6-0.
- https://sites.google.com/site/clarioncognitivearchitecture/publications.
- http://ccrg.cs.memphis.edu/framework.html.
- http://ccrg.cs.memphis.edu/assets/framework/The-LIDA-Tutorial.pdf.

Este trabalho gerou até agora dois artigos publicados, o primeiro deles em um evento nacional e o segundo em um periódico importante da área de arquiteturas cognitivas:

- Building a Motivacional Subsystem For The Cognitive System Toolkit. - Artigo apresentado no XIII Simpósio Brasileiro de Automação Inteligente, Porto Alegre – RS, 1º – 4 de Outubro de 2017.

- The Multipurpose Enhanced Cognitive Architecture (MECA) - Artigo publicado no *Biologically Inspired Cognitive Architectures* - BICA Journal, 2017.