

Leonardo Vailatti Eichstaedt

Problema de embasamento de símbolos em um sistema multi-contexto

Florianópolis

2019

Leonardo Vailatti Eichstaedt

Problema de embasamento de símbolos em um sistema multi-contexto

Trabalho de Conclusão do Curso de Graduação em Ciências da Computação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do Título de Bacharel em Ciências da Computação. Orientador: Prof. Dr. Elder Rizzon Santos.

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CIÊNCIAS DA COMPUTAÇÃO

Florianópolis

2019

“The limits of my language means the limits of my world.”
- Ludwig Wittgenstein

*“The poet is the man of metaphor: while the philosopher
is interested only in the truth of meaning,
beyond even signs and names,
and the sophist manipulates empty signs.
The poet plays on the multiplicity of signifieds.”*
- Jacques Derrida

Resumo

O problema de embasamento de símbolos (symbol grounding problem) está relacionado ao problema de como símbolos ganham seus significados. Conseqüentemente, este problema está relacionado com o problema de como a consciência é capaz de criar uma relação entre um sistema de símbolos formais e seus referentes. Como símbolos sem sentido, manipulados unicamente com base em suas formas, podem ser embasados em algo além do que outros símbolos sem sentido. Pode-se questionar se um agente artificial poderia desenvolver uma capacidade semântica autônoma, tal qual a de seres humanos, para conectar seus símbolos com o ambiente no qual o agente artificial está situado. Este trabalho avalia algumas estratégias que foram propostas para solucionar este problema, as quais estão organizadas em três abordagens principais: representacionalista, semi-representacionalista e não representacionalista. Após uma introdução ao problema filosófico e às soluções propostas, este trabalho aplica uma estratégia sugerida por Harnad, baseada em um modelo híbrido que implementa uma mistura de características características de sistemas simbólicos e conexionistas, a um agente BDI visto como um sistema multi-contexto, definido pela linguagem Sigon. O trabalho mostra como o embasamento de símbolos de um agente às percepções do ambiente pode ser uma aplicação das estratégias para a resolução do problema de embasamento de símbolos em um contexto de criação de agentes multi-contexto. Dessa forma, apesar de ser de uma maneira limitada, estaremos simulando o comportamento de um agente que consegue formar conexões entre percepções não-simbólicas do ambiente com equivalentes simbólicos internos do agente. Porém, dentro dessas limitações da abordagem utilizada está o núcleo do problema filosófico de aquisição de significados que continua em aberto para ser explorado por trabalhos futuros.

Palavras-chaves: Agentes, inteligência artificial, Symbol Grounding Problem, sistemas multi-contexto.

Abstract

The symbol grounding problem is related to the problem of how symbols gain their meanings. Consequently, this problem is related to the problem of how consciousness is capable of creating a relationship between a system of formal symbols and their referents. How meaningless symbols, manipulated solely on the basis of their shapes, can be grounded on something other than (other) meaningless symbols. One may wonder whether an artificial agent could develop an autonomous semantic capacity, such as that of human beings, to connect their symbols with the environment in which the artificial agent is situated. This paper evaluates some strategies that have been proposed to solve this problem, which are organized in three main approaches: representational, semi-representational and non-representational. After an introduction to the philosophical problem and the proposed solutions, this work applies a strategy suggested by Harnad, based on a hybrid model that implements a mixture of characteristic features of symbolic and connectionist systems, to a BDI agent seen as a multi-context system, defined with the Sigon language. The work shows how the basing of symbols of an agent to the perceptions of the environment can be an application of the strategies for solving the problem of the symbol grounding problem in a context of creation of multi-context agents. Thus, in spite of being in a limited way, we are simulating the behavior of an agent that manages to form connections between non-symbolic perceptions of the environment with internal symbolic equivalents of the agent. But within these limitations of the approach used lies the core of the philosophical problem of meaning acquisition that remains open to be explored by future works.

Key-words: Agentes, artificial inteligência, Symbol Grounding Problem, multi-context.

Lista de ilustrações

Figura 1 – Representação do teste de Turing.	18
Figura 2 – Representação de uma rede neural, mostrando as unidades e as camadas formadas por elas.	21
Figura 3 – Representação de um agente.	23
Figura 4 – Agente BDI	28
Figura 5 – Esquema do argumento do quarto chinês.	31
Figura 6 – Representação e representante.	33
Figura 7 – Esquema de Peirce para significação simbólico.	35
Figura 8 – Estrutura da M1: E é o ambiente, S_1 é o estado interno de M1, LoA_1 é o nível de abstração no qual a M1 interage com E, $f(e)$ é a função que identifica S_1 , onde (e) é uma dada interação entre o agente e o ambiente.	49
Figura 9 – Estrutura da M2: E é o ambiente, M2 não interage com o ambiente, mas interage com M1; o ambiente age em M2 indiretamente, por meio de um processo evolucionário. Sym_1 é o símbolo elaborado por M2. LoA_2 é o nível de abstração no qual a M2 interage com E. (S_1, Sym_1) é a associação entre símbolo e o estado interno da M1, a saída da elaboração de M2.	50
Figura 10 – Arquitetura de um agente inteligente de duas máquinas. Um agente de duas máquinas recebe/envia alguma ação/percepção (e) do/para o ambiente E. E interage com a M1 e age na M2, modificando-a de acordo com o processo evolucionário. Qualquer ação é relacionado com o estado interno (S_1) da M1 em um nível de abstração específico, LoA_1 . M1 comunica seus estado internos para M2. O estado interno de M1 é transduzido para uma entrada de M2, que associa uma entrada a um símbolo (Sym_1) . M2 guarda o estado e o símbolo relacionado em sua memória. Para qualquer entrada, M2 segue o procedimento definido pela regra de performance. Cada símbolo é selecionado pela M2 é uma função (g) do estado interno, S_1 . Já que S_n também é um resultado da função - $f(e)$ - a saída de M2 é uma função de uma função, $g(f(e))$	51
Figura 11 – Esquema de cognição simbólica.	54
Figura 12 – Esquema de cognição segundo Brooks.	54
Figura 13 – Exemplo de arquitetura de subsunção proposta por Brooks.	56
Figura 14 – Na arquitetura de subsunção, as máquinas de estados finitos são ligadas em camadas de controle. Cada camada é construída em cima de outras camadas já existentes.	58
Figura 15 – Imagem do robô KISMET.	60

Figura 16	– Processo de reconhecimento de estímulo sensorial visual em um ser humano.	72
Figura 17	– Processo de reconhecimento de estímulo sensorial visual em um agente artificial.	73
Figura 18	– Coleção de imagens que compõem o conjunto C de dados não-simbólicos.	73
Figura 19	– Função que transforma uma percepção não-simbólica ao seu equivalente simbólico.	74
Figura 20	– Relação entre conjunto de dados não-simbólicos e conjunto de símbolos.	75
Figura 21	– Estrutura do agente BDI em Sigon.	76
Figura 22	– Processo de reconhecimento de estímulo sensorial visual em um agente artificial que consegue relacionar a percepção visual a um símbolo.	78
Figura 23	– Modelo de um agente que lida com a percepção de forma a transformá-la em um símbolo.	79
Figura 24	– Exemplo de imagens do MNIST.	83
Figura 25	– Estrutura da rede neural	85
Figura 26	– Canvas para usuário desenhar número.	87
Figura 27	– Canvas com os números desenhados pelo usuário.	87
Figura 28	– Resultado da execução da rede neural.	88
Figura 29	– Resultado da execução da rede neural com o desenho de um número “2” como entrada.	89
Figura 30	– Resultado da execução da rede neural com o desenho de um número “1” como entrada.	92
Figura 31	– Dados do agente antes de ser detectado uma percepção pelos sensores.	93
Figura 32	– Percepção do agente.	93
Figura 33	– Dados do agente antes de ser detectado uma percepção pelos sensores.	93
Figura 34	– Função que transforma uma percepção não-simbólica vinda do Canvas ao seu equivalente simbólico.	94

Lista de tabelas

Tabela 1 – Classificação inteligência artificial	17
Tabela 2 – Esquema de classificação das soluções para resolução do problema do embasamento simbólico	37
Tabela 3 – Correspondência entre comportamento não-verbal e funções proto-linguísticas	61

Sumário

1	INTRODUÇÃO	13
1.1	OBJETIVOS	14
1.1.1	Objetivo Geral	14
1.1.2	Objetivo Específicos	15
1.1.3	Metodologia	15
1.2	ORGANIZAÇÃO	16
1.3	JUSTIFICATIVA	16
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	INTELIGÊNCIA ARTIFICIAL	17
2.1.1	IA Simbólica	20
2.1.2	IA Conexionista	21
2.2	AGENTES	23
2.2.1	Categorias de Agentes	25
2.2.2	Agentes BDI	26
2.3	SISTEMAS MULTI-CONTEXTOS	27
2.4	ARGUMENTO DO QUARTO CHINÊS	30
2.5	PROBLEMA DO EMBASAMENTO SIMBÓLICO	32
2.6	SOLUÇÕES PARA O PROBLEMA DO EMBASAMENTO SIMBÓLICO	36
2.6.1	Representacionalista	37
2.6.1.1	Modelo Híbrido	38
2.6.1.2	Hipótese do Roubo Simbólico	40
2.6.2	Semi-representacionalista	42
2.6.2.1	Modelo epistemológico	42
2.6.2.2	Problema físico de embasamento de símbolos	44
2.6.2.3	Semântica Baseada em Ação	47
2.6.3	Não Representacionalista	53
2.6.3.1	Arquitetura de Subsunção	53
2.6.3.2	Modelo baseado em comportamento	59
2.7	CONSIDERAÇÕES FINAIS	62
3	TRABALHOS RELACIONADOS	63
3.1	SIGON: UMA LINGUAGEM PARA AGENTES COMO SISTEMAS MULTI-CONTEXTOS	63
3.2	IMPLEMENTAÇÃO DE UM FRAMEWORK PARA O DESENVOLVIMENTO DE AGENTES COMO SISTEMA MULTI-CONTEXTOS	66

3.3	CONSIDERAÇÕES FINAIS	70
4	DESENVOLVIMENTO	72
4.1	PROPOSTA	72
4.2	TEORIA	78
4.3	IMPLEMENTAÇÃO	82
4.4	RESULTADOS	92
	Conclusão	95
4.5	Trabalhos Futuros	97
	Referências	100
	Apêndices	104
	APÊNDICE A ARTIGO	105
	APÊNDICE B CÓDIGO FONTE	125
B.1	Reconhecedor de Dígitos	125
B.2	Agente	134

1 INTRODUÇÃO

As capacidades cognitivas dos seres humanos são tão importantes para o nosso dia a dia e para nosso entendimento de nós mesmos que o próprio nome científico de nossa espécie se refere as tais propriedades: homo sapiens, ou seja, “homem sábio”. O campo da Inteligência Artificial tenta não apenas compreender, mas também construir entidades inteligentes (NORVIG; RUSSELL, 2013).

É evidente, portanto, que para que a construção de entidades inteligentes seja viável, é necessário um entendimento de como funciona o pensamento no ser humano. Sendo assim, a inteligência artificial é uma disciplina que está em constante comunicação com outras áreas de estudo como as Ciências Cognitivas, Filosofia, Psicologia etc.

Os desenvolvimentos iniciais de Allen Newell e Herbert A. Simon marcaram a “era de ouro” no desenvolvimento de programas de inteligência artificial e criaram uma atmosfera de otimismo em relação às perspectivas futuras do desenvolvimento de entidades inteligentes.

O sucesso de seus programas e modelos de cognição levaram a Newell e Simon a formularem, em 1976, uma hipótese chamada de “sistema de símbolos físicos” (NEWELL; SIMON, 1976). Essa hipótese afirma que “um sistema de símbolos físicos têm os meios necessários e suficientes para uma ação inteligente geral”(NEWELL; SIMON, 1976). O que essa hipótese implica é que computadores, quando lhes fornecermos o programa apropriado para processamento de símbolos, serão capazes de realizar ações inteligentes.

Essa teoria é relacionada às teorias computacionais da mente proposta, em sua forma moderna, por Hilary Putnam (1967) e Jerry Fodor (1975). As teorias computacionais da mente englobam diferentes visões que se desenvolveram do fundamento de que a mente humana é um sistema de processamento de informações (RESCORLA, 2017). Essas visões da mente como um computador, mais especificamente como um computador processando símbolos de acordo com regras, é diretamente relacionada aos primeiros esforços no desenvolvimento de uma inteligência artificial no campo da ciência da computação.

Em 1980, o filósofo John Searle publicou um artigo que apresentou um experimento mental que veio a se tornar um dos mais conhecidos na filosofia e ciência da computação. Seu artigo propôs um argumento contra a teoria computacional da mente e a noção de que o que constitui o pensamento é apenas uma manipulação de símbolos (SEARLE, 1980).

O argumento do quarto chinês, como venho a ser chamado, critica a falta de “intencionalidade” (ou “entendimento”) da máquina em relação com o que ela está fazendo, portanto ela não poderia ser caracterizada como pensante, mas no máximo como “si-

mulando” o pensamento. O erro fatal em se tratar o pensamento como processamento simbólico, Searle quer mostrar, é que essa manipulação de símbolos é feita com base na “forma” do símbolo ao invés de seu significado. Os computadores e a nossa mente se diferenciariam justamente nessa capacidade de atribuir um significado para símbolos, sendo essa capacidade presente em nossas mentes e ausente nos computadores. Portanto, cognição não pode ser apenas a manipulação simbólica.

Em 1990, Stevan Harnad formula o “*Symbol Grounding Problem*” (daqui em diante chamado de problema do embasamento simbólico) para questionar como é feita a transposição desse abismo entre forma e significado de um símbolo em nossas mentes e como podemos reproduzir tal capacidade mental em um computador, ou seja, como embasar (*to ground*) um símbolo a um significado que não seja apenas outro símbolo sem significado (HARNAD, 1990). Podemos entender então o problema do embasamento simbólico como um problema relacionado à como palavras (símbolos) adquirem significado.

O presente trabalho não busca apenas investigar o problema do embasamento simbólico, seus antecedentes e desenvolvimentos, mas busca também demonstrar que tal problema não é apenas de caráter filosófico e abstrato mas que possui consequências práticas no desenvolvimento de inteligências artificiais.

O desenvolvimento de inteligências artificiais neste trabalho se dará por meio de agentes inteligentes. Estes correspondem a uma visão de desenvolvimento de inteligências artificiais, muito associada a Peter Norvig e Stuart Russell (2013), que podem ser caracterizados por sua ênfase na capacidade de interação com o ambiente em que estão e sua autonomia para tomar a melhor decisão esperada visto a sua base de conhecimento.

Para demonstrar tal aplicabilidade, será proposto e avaliado um protótipo de um modelo para embasamento simbólico que será utilizado em um *framework* de sistema multi-contexto para agentes inteligentes, ou seja, agentes cujas capacidades são expressas em linguagens lógicas diferentes. Espera-se que por meio de símbolos embasados seja possível utilizá-los como componente de regras ponte, permitindo que módulos do agente especificados em diferentes lógicas possam se comunicar mais facilmente por meio destes símbolos que possuem significados definidos.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Propor um modelo que lide com *Symbol Grounding* integrado a um *framework* de Sistemas Multi-Contextos.

1.1.2 Objetivo Específicos

- Analisar o problema de *Symbol Grounding* em seus aspectos filosóficos e suas consequências para a ciência da computação, focando prioritariamente na área de inteligência computacional.
- Analisar a fundamentação e o estado da arte de sistemas multi-contexto.
- Analisar o estado da arte nas possíveis respostas para o problema de *Symbol Grounding*.
- Propor um modelo computacional com base em tentativas de responder o problema de *Symbol Grounding*.
- Incorporar a modelagem de uma possível solução do *Symbol Grounding* em um ambiente de agentes em Sistemas Multi-Contextos, buscando enriquecer a comunicação entre contextos.
- Analisar os resultados obtidos, de modo que possa ser avaliado a viabilidade da modelagem proposta para solução do *Symbol Grounding* no contexto de Sistemas Multi-Contextos para agentes inteligentes.

1.1.3 Metodologia

- Estudar o problema de *Symbol Grounding* na bibliografia clássica e artigos relacionados, de modo que se tenha uma maior compreensão do problema, ou seja, entender claramente seu enunciado e suas implicações filosóficas e computacionais para a área da inteligência artificial.
- Buscar na literatura multidisciplinar respostas que foram propostas ao problema de *Symbol Grounding*.
- Selecionar uma ou mais dessas propostas para utilizá-la(s) como base na construção de um modelo computacional que busque efetuar a fundamentação de símbolos.
- Utilizar o modelo construído em um ambiente de agentes inteligentes baseados em um Sistemas Multi-Contextos, a fim de analisar seu comportamento.
- Avaliar os resultados obtidos do modelo implementado, de modo que se possa dar algum respaldo com relação à viabilidade da solução ao problema de *Symbol Grounding* utilizado como inspiração na construção do modelo.

1.2 ORGANIZAÇÃO

No capítulo 2 são apresentados os principais conceitos que fundamentam esta pesquisa. No capítulo 3 são mostrados os trabalhos relacionados com esta pesquisa. A apresentação do modelo e a implementação do agente feita no capítulo 4. A conclusão mostra as considerações finais e os trabalhos futuros

1.3 JUSTIFICATIVA

A inteligência artificial tem conexões próximas com a filosofia, porque estas compartilham muitos conceitos entre si, por exemplo, ação, consciência, epistemologia (o que é sensato dizer sobre o mundo) etc.

A inteligência artificial aplicada progrediu massivamente e agora está presente em muitos aspectos do nosso ambiente. Apesar desse desenvolvimento, os problemas básicos da inteligência artificial permanecem, e ignorá-los “porque nossos sistemas estão melhorando de qualquer maneira” é uma estratégia arriscada. A maneira de seguir em frente parece voltar ao básico, e isso inclui os problemas filosóficos mais fundamentais.

A visão clássica “computacionalista” era que a cognição é computação sobre representações, que pode, assim, ocorrer em qualquer sistema computacional, natural ou artificial. No entanto, a característica definidora dessa abordagem está agora sob ameaça pelo problema que é conhecido como problema do embasamento de símbolos: um sistema de processamento de símbolos não pode ser o substrato correto para a cognição, porque os constituintes mais básicos do sistema não têm significado intrínseco, então como tais estados internos podem representar uma categoria de coisas no mundo externo?

Uma quantidade desconcertante de perguntas vêm à mente: devemos reparar a inteligência artificial clássica, já que a inteligência ainda é processamento de informações de entrada e saída? Abandonar a pretensão de inteligência geral e continuar com os sucessos da inteligência artificial em termos práticos? Abraçar a cognição incorporada, o desenvolvimento cognitivo do ponto de vista da enação ou a mente estendida? Reviver as redes neurais em uma nova forma? Substituir inteligência artificial por “sistemas cognitivos”? Procurar por sistemas alternativos, dinâmicos, inspirados no cérebro?

Esse trabalho serve para que, antes de respondermos essas questões de maneira precipitada, possamos rever os aspectos mais fundamentais desse problema que assola os sistemas simbólicos, explorar algumas das soluções que foram propostas e demonstrar como essas propostas podem ser úteis no estado da arte da pesquisa sobre agentes inteligentes.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 INTELIGÊNCIA ARTIFICIAL

O conceito de Inteligência Artificial (IA) é nebuloso e podemos encontrar múltiplas definições na literatura. Essa multiplicidade de significados acontece pelo motivo de que o próprio conceito de inteligência não é simples de definir. Em nosso cotidiano podemos classificar intuitivamente casos isolados de comportamento como “inteligentes”, mas encontrar uma definição única que abranja todas as possibilidades de uso do termo é uma tarefa muito mais difícil. Podemos tentar encapsular algumas das propostas para tentar responder “O que é Inteligência Artificial?” por meio de uma análise de um conjunto de respostas que foram dadas para essa questão. Estas respostas assumem que IA deve ser definida em termos de seus objetivos (BRINGSJORD; GOVINDARAJULU, 2018).

Em linhas gerais podemos ter definições que relacionam Inteligência Artificial aos processos de pensamento e raciocínio ou ao comportamento. Até mesmo a medida utilizada para mensurar o sucesso de uma inteligência artificial não é consensualmente definida. Podemos utilizar como medida o desempenho humano ou ideal de inteligência (NORVIG; RUSSELL, 2013).

Tabela 1 – Classificação inteligência artificial

	Desempenho Humano	Racionalidade Ideal
Baseado no raciocínio	Sistemas que pensam como humanos	Sistemas que pensam racionalmente
Baseado no comportamento	Sistemas que agem como humanos	Sistemas que agem racionalmente

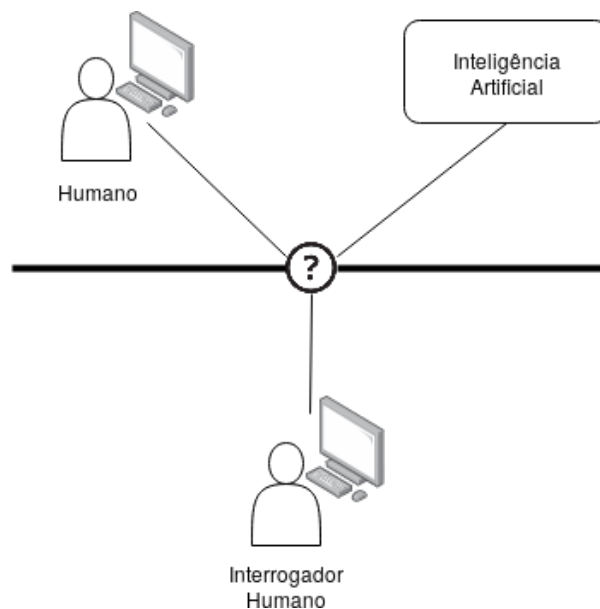
Fonte: (BRINGSJORD; GOVINDARAJULU, 2018)

Em seu clássico artigo, *Computing Machinery and Intelligence* (1950), Alan Turing faz a pergunta: “As máquinas podem pensar?”. Como o processo de pensar é de difícil definição, Turing substituiu essa pergunta por um novo problema, descrito em termos do “jogo da imitação” (TURING, 1950). Nesse novo aspecto, a pergunta a ser feita seria: “Há como imaginar um computador digital que faria bem o jogo da imitação?”.

O teste de Turing funciona da seguinte maneira: existem dois humanos e um sistema de inteligência artificial situados em um mesmo ambiente. Um dos humanos é um interrogador que está separado (por uma barreira) do outro humano e do sistema de IA. Este interrogador entra em uma conversa em linguagem natural (via teclado) com o outro humano e também com a máquina, e caso ele não consiga distinguir se está conversando

com a máquina ou com o ser humano é um indicativo de que o sistema é inteligente e passou no Teste de Turing.

Figura 1 – Representação do teste de Turing.



Fonte: o autor(2019)

Dessa maneira, Turing claramente adota a posição de que basta com que um programa se comporte como uma pessoa para que possamos definir esse programa como inteligente.

Outros autores não acreditam que apenas imitar o comportamento humano é suficiente para caracterizar inteligência e preferem adotar a posição de que para um programa ser inteligente, deve pensar como um ser humano. Essa posição é muitas vezes associada a uma abordagem interdisciplinar com o campo da ciência cognitiva, pois para dizer que um programa pensa como um ser humano, devemos poder determinar como os seres humanos pensam. Logo, uma vez que se tenha um modelo mental preciso, seria possível implementar um modelo equivalente em um programa computacional(NORVIG; RUSSELL, 2013).

McCarthy e Hayes (1969) criticam ambas abordagens que olham para o ser humano como parâmetro de sucesso. Segundo McCarthy e Hayes (1969):

“Houveram várias tentativas de projetar inteligência com o mesmo tipo de flexibilidade que a de um humano. Isso significou coisas diferentes para diferentes pesquisadores, mas nenhum obteve muito sucesso, mesmo no sentido de inteligência geral usada pelo investigador em questão.”

Essas abordagens são criticadas por McCarthy pois, segundo ele, se distraem com certos aspectos de pensamento/ação superficiais, restritos aos humanos.

Para ele, a construção de um programa inteligente estaria melhor relacionada com manipuladores de fatos (MCCARTHY; HAYES, 1969), ou seja, com as leis do pensamento. Entidades inteligentes devem estar equipadas com uma representação do mundo e com base nessa representação devem ser capazes de responder questões feitas.

Nessa concepção, a ênfase é dada em inferências corretas, ou seja, seguir as leis do pensamento corretamente, com base em uma base de fatos.

Peter Norvig e Stuart Russell também adotam uma abordagem que busca ir além do ser humano como padrão de sucesso para uma IA.

Segundo Russell (1997):

"Minha própria motivação para estudar a IA é criar e entender a inteligência como uma propriedade geral dos sistemas, e não como um atributo específico dos seres humanos."

Porém, Norvig e Russell (2013) adotam a abordagem de agentes racionais. Dessa forma, como o próprio nome indica, a ênfase da abordagem é a ação. Nesse caso, a ênfase não está nas inferências corretas pois estas são apenas parte daquilo que caracteriza um agente racional. A inferência correta não representa toda racionalidade (NORVIG; RUSSELL, 2013). Existem situações em que o agir racionalmente não envolve inferência e em outros casos, não existe uma ação comprovadamente correta a ser feita, mas de qualquer forma o agente deve ainda ser capaz de autonomamente tomar alguma decisão.

Outra distinção que pode ser feita entre definições de IA é o conceito de inteligência artificial “fraca” e “forte”.

A IA “forte” busca criar um programa que possua todas as capacidades mentais que seres humanos possuem. A IA “fraca”, por outro lado, são restritas na realização de certa tarefa e são feitas de forma com que apenas pareçam inteligentes (SEARLE, 1980).

As mais variadas hipóteses surgiram de como uma inteligência pode ser criada artificialmente, isso implicou no surgimento de diferentes tipos de técnicas práticas para criação dessas inteligências. O campo da inteligência artificial se desenvolveu tanto por meio de autores incrementando técnicas já existentes, quanto pela rejeição total de paradigmas passados e estabelecimento de novos. Desta forma, dentro do campo geral da inteligência artificial as mais diversas teses e tipos de implementação convivem e dialogam entre si. Neste trabalho, iremos dividir os paradigmas utilizados para especificação de Inteligências Artificiais em duas grandes categorias: Inteligência Artificial Simbólica e Inteligência Artificial Conexionista.

O campo da Inteligência Artificial nasceu simbólico e lógico. Os pioneiros da Inteligência Artificial formalizaram muitas teorias, hipóteses e aplicações elegantes. A partir dos anos 80, o pêndulo virou-se para o conexionismo, um paradigma inspirado nas conexões

neurais do cérebro. Com a crescente quantidade de dados acessíveis e poder computacional cada vez mais forte, os modelos conexionistas ganham um impulso considerável nos últimos anos (BRINGSJORD; GOVINDARAJULU, 2018).

2.1.1 IA Simbólica

A Inteligência Artificial simbólica foi concebida em tentativas de representar o conhecimento humano de forma explícita, por meio de fatos, regras, e outras formas simbólicas declarativas. Essa categoria de técnicas é muitas vezes referida como GOFAI (*Good Old Fashioned AI*). Essa é abordagem clássica de codificar um modelo do problema e esperar o sistema processar os dados de entrada de acordo com esse modelo para gerar uma solução.

Newell e Simon (1976) propuseram a *physical symbol system hypothesis*, argumentando que símbolos são as raízes da ação inteligente e, portanto, o tópico primário da inteligência artificial.

De acordo com a hipótese por eles proposta, um *physical symbol system* é um dispositivo computacional para manipulação de símbolos. Símbolos, por sua vez, formam expressões, ou estruturas simbólicas por meio de algum tipo de conexão. Essas estruturas funcionariam como representações internas de ambientes. Para Newell e Simon, computadores e mentes humanas são *physical symbol systems*, apesar de operarem em hardwares diferentes. Em outras palavras, essa hipótese implica na existência de computação no nível simbólico em um sistema independente do substrato físico no qual ele opera (NEWELL; SIMON, 1976).

Abordagens simbólicas para representar conhecimento são altamente estruturadas, o que pode ser traçado para suas origens na teoria lógica antes mesmo do surgimento do campo da inteligência artificial. Desde Aristóteles logicistas tentam desenvolver sistemas baseados em regras para expressão de conhecimento e inferência (NORVIG; RUSSELL, 2013). A unidade básica para a representação simbólicas são átomos simbólicos, por exemplo, palavras ou conceitos específicos.

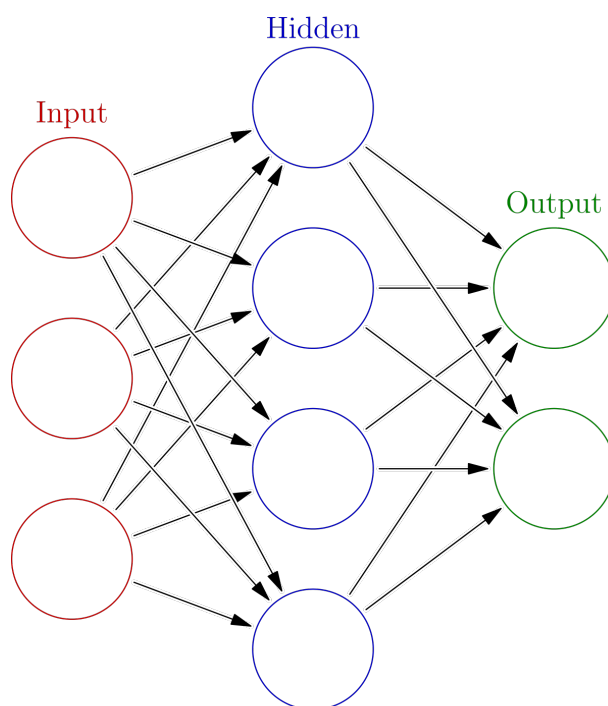
Os sistemas que se enquadram nessa categoria geralmente envolvem raciocínio dedutivo, inferências lógicas e algum tipo de algoritmo de busca que encontra as soluções dentro das restrições do modelo especificado (NORVIG; RUSSELL, 2013).

Algoritmos simbólicos eliminam opções que violam o modelo especificado e podem ser verificadas para sempre produzir uma solução que satisfaça todas as restrições muito mais facilmente do que suas contrapartes conexionistas. Como normalmente há pouco ou nenhum treinamento algorítmico envolvido, o modelo pode ser dinâmico e mudar tão rapidamente quanto necessário.

2.1.2 IA Conexionista

O conexionismo pode fornecer uma nova forma, alternativa à teoria clássica da mente, para entender a natureza da mente e sua relação com o cérebro. Esta abordagem busca explicar as habilidades cognitivas por meio de modelos simplificados do cérebro, as redes neurais. Essas redes neurais são compostas de um grande número de unidades, análogas aos neurônios, junto com pesos, análogos às conexões (sinapses) cerebrais, que determinam a medida da força de conexões entre as unidades (GARSON, 2018).

Figura 2 – Representação de uma rede neural, mostrando as unidades e as camadas formadas por elas.



Fonte: Wikimedia

Uma rede neural consiste de um grande número de unidades unidas em um padrão de conexões. Unidades em uma rede são geralmente segregadas em três classes: unidades de entrada, que recebem informações a serem processadas, unidades de saída onde os resultados do processamento são encontrados e unidades intermediárias, chamadas ocultas. Se uma rede neural modelasse todo o sistema nervoso humano, as unidades de entrada seriam análogas aos neurônios sensoriais, às unidades de saída aos neurônios motores e às unidades ocultas a todos os outros neurônios (GARSON, 2018).

O padrão de ativação configurado por uma rede é determinado pelos pesos, ou seja, pela força das conexões entre as unidades. Encontrar o conjunto correto de pesos para realizar uma determinada tarefa é o objetivo central da pesquisa conexionista.

Foram criados algoritmos de aprendizagem que podem calcular os pesos certos para realizar muitas tarefas. [Hinton \(1992\)](#) faz uma revisão desses algoritmos, classificando-os em duas grandes categorias de algoritmos para aprendizagem de pesos: aprendizagem supervisionada e não supervisionada.

A abordagem de aprendizado supervisionado consiste no conjunto de técnicas em que existe um “supervisor” que sabe previamente qual é a resposta esperada de um sistema dado uma entrada. Para efetuar o aprendizado, o algoritmo compara sua saída com a deste supervisor, para que assim, se ajuste em busca de minimizar o erro de sua resposta ([BISHOP, 2006](#)). Os algoritmos utilizam um conjunto de treinamento composto de muitos exemplos de entradas, já classificados, para induzir um modelo que seja capaz de classificar novas instâncias de forma precisa, com base no aprendizado obtido com o treinamento com os dados de treinamento. Este conjunto externo de exemplos “supervisiona” o processo de treinamento. Para usar este método, é necessário um conjunto de treinamento composto de muitos exemplos de entradas e suas saídas desejadas para uma determinada tarefa. O aspecto chave dessa categoria de técnicas é que o usuário não especifica as regras do domínio que está sendo modelado. A rede descobre as regras dos dados de treinamento. O usuário fornece dados de entrada e dados de saída de amostra (quanto maior e mais diversificado for o conjunto de dados, melhor) ([BISHOP, 2006](#))([SATHYA; ABRAHAM, 2013](#)).

Na abordagem de aprendizado não-supervisionado o algoritmo atua sem um supervisor. O conjunto de dados utilizado na abordagem de aprendizado não-supervisionado estão classificados, ou seja, o conjunto de dados não possui uma saída esperada para cada uma de suas instâncias. Nestes algoritmos não existe influência humana direta e o conhecimento se dá a partir do agrupamento dos dados de entradas, analisados e agrupados conforme a proximidade dos seus valores ([HAN; KAMBER; PEI, 2006](#)).

Esses algoritmos não precisam de um modelo do mundo. Eles só precisam de dados de amostra suficientes a partir dos quais o modelo do mundo pode ser inferido estatisticamente. Esta é uma característica muito poderosa, mas também uma fraqueza. Os recursos de entrada devem ser selecionados com muito cuidado. Eles também precisam ser normalizados ou escalonados, para evitar que um recurso ultrapasse os outros e pré-processados para serem mais significativos para a classificação.

Os algoritmos orientados por dados pressupõem implicitamente que o modelo do mundo que eles estão capturando é relativamente estável. Isso os torna muito eficazes para problemas em que as regras do jogo não estão mudando significativamente, ou mudando a uma taxa que é lenta o suficiente para permitir que novas amostras de dados suficientes sejam coletadas para reciclagem e adaptação à nova realidade.

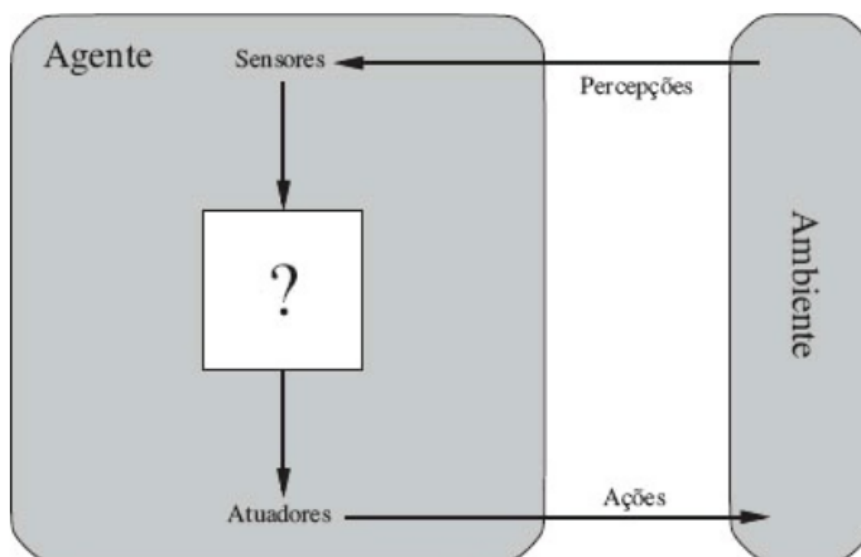
2.2 AGENTES

Existem diversas definições para agentes. Uma explicação para a divergência de opiniões surge do fato de existirem diversos tipos de agentes, utilizados em diferentes domínios de aplicações. O que se percebe é que os conceitos são elaborados de acordo com o objetivo que se quer atingir com o desenvolvimento de um agente.

A definição de agentes proposta por [Norvig e Russell \(2013\)](#) parte de uma noção geral de agente, definido como entidade autônoma que é capaz de agir sobre o ambiente em que se encontra, com base nas informações que percebe através de sensores. Portanto, o elemento essencial que esses autores destacam para definir um agente é a capacidade de agir.

É evidente que, seguindo à risca essa definição, é abrangido uma grande quantidade de entidades. Até mesmo um termostato se encaixa nessa definição. Os tipos de agentes inteligentes autônomos de interesse para o campo da inteligência artificial não são os tipos de coisas que encontramos na natureza. Pelo contrário, são os artificialmente criados. Assim, um agente artificial, inteligente e autônomo é qualquer coisa que criamos capaz de ações baseadas na informação que percebe, em suas próprias experiências e em suas próprias decisões sobre qual ação ela realiza.

Figura 3 – Representação de um agente.



Fonte: ([NORVIG; RUSSELL, 2013](#))

[Franklin e Graesser \(1996\)](#) buscam propriedades compartilhadas por agentes para poder identificar o que constitui a essência de um agente. Para eles, essa essência é estar situado e fazer parte de algum ambiente. Cada agente sente seu ambiente e age de forma autônoma sobre ele. Cada agente atua em busca de sua própria agenda, seja satisfazendo os impulsos evoluídos, como em humanos e animais, ou perseguindo objetivos projetados,

como acontece em agentes de software. Dessa forma, um agente autônomo é um sistema situado dentro e parte de um ambiente que percebe o ambiente e age sobre ele, ao longo do tempo, em busca de sua própria agenda e de modo a afetar o que ele percebe no futuro.

Ao apontar a relação entre um agente e seu ambiente como uma característica fundamental de sua definição, Franklin e Graesser (1996) revelam um fato importante sobre o grau de complexidade de um agente: à medida que aumentamos a complexidade do ambiente e a variedade de problemas que devem ser resolvidos para que o agente atinja seus objetivos, é necessária mais flexibilidade e sofisticação por parte do agente (FRANKLIN; GRAESSER, 1996).

Tendo outro enfoque, Shoham (1993) afirma que um agente é uma entidade à qual se atribuem estados, denominados de estados mentais. Os estados mentais usuais são: crenças, decisões, capacidades, objetivos, intenções, compromissos e expectativas, conceitos análogos ou similares aos humanos. É em função destes estados que o agente age e toma suas decisões.

Hayes-Roth (1995) destaca três funções que são continuamente realizadas por agentes inteligentes: percepção de condições dinâmicas no ambiente; ação para afetar as condições do meio ambiente; e raciocínio para interpretar percepções, resolver problemas, extrair inferências e determinar ações.

Feita uma revisão de diversas definições para agentes inteligentes, podemos destacar algumas propriedades que são comumente aceitas como pertencentes à agentes:

- **Autonomia:** um agente deve controlar suas próprias ações de acordo com estados internos, sem a intervenção humana ou de outros agentes. Para ter autonomia, o agente deve ter um certo grau de inteligência. Essa, por sua vez, deve agir sobre seu conhecimento embutido, sua experiência, bem como em suas percepções do mundo. Isso implica que, ao ser criado, um agente já é provido de um certo grau de conhecimento, bem como, em alguns casos, capacidade de aprendizagem;
- **Domínio:** um agente sempre está situado em algum ambiente. O ambiente é o domínio ou o mundo do agente. É no ambiente em que se encontra que o agente irá receber estímulos e agir. O domínio do agente é um dos principais determinantes do quão complexo um agente deverá ser para poder cumprir seus objetivos;
- **Reatividade:** um agente deve, por meio de recursos de percepção, captar e reagir de acordo com a mudanças no ambiente em que estiver inserido. Portanto, a capacidade de percepção está relacionada à reatividade. Agentes podem empregar sensores como um teclado ou uma câmera de vídeo para coletar informações sobre o mundo em que se encontra;

- Proatividade: agentes, além de reagirem às alterações no ambiente, apresentam um comportamento orientado a objetivos, tomando iniciativas quando julgarem apropriado;
- Adaptabilidade: agentes devem ser capazes de se adaptar ao meio externo, examinar os processos efetuados anteriormente e adaptar suas ações futuras;
- Habilidade social: agentes podem, por impossibilidade de resolução de certos problemas ou por algum tipo de conveniência, ter a capacidade de interagir com outros agentes com o objetivo de obter informações, conseguir ajuda para atingir suas metas, ou ainda para auxiliarem outros agentes.

2.2.1 Categorias de Agentes

Um ponto essencial na criação de agentes se encontra na especificação de habilidades que permitam com que os agente ajam de maneira autônoma em seu ambiente. Com base em [Wooldridge e Jennings \(1995\)](#), podemos categorizar os agentes com base em sua arquitetura:

- Reativos: Os agentes reativos são estruturalmente simples, pois não possuem uma representação do seu ambiente e não possuem capacidade de realizar raciocínios lógicos complexos. Esses agentes selecionam suas ações com base no seu estado atual. Possuem inteligência limitada, não tendo capacidade de raciocinar sobre suas intenções, reagindo apenas sobre planos pré-definidos. Esse tipo de agente possui uma lista interna de regras de ações a serem executadas baseadas na situação do agente ([NORVIG; RUSSELL, 2013](#)). [Wooldridge e Jennings \(1995\)](#) propõem uma definição que contrasta este tipo de agente com um agente deliberativo. Para eles, um agente reativo é um agente que não inclui nenhum tipo de modelo simbólico central sobre o mundo e não usa um raciocínio simbólico complexo;
- Deliberativos: Um agente deliberativo é definido como um agente que contém um modelo simbólico explicitamente representado do mundo real e no qual as decisões são tomadas por meio do raciocínio lógico baseado na correspondência de padrões e na manipulação simbólica ([WOOLDRIDGE; JENNINGS, 1995](#)). Esses agentes têm a habilidade de raciocínio sobre suas intenções e crenças, e podem, por um processo explícito, criar e escolher ações e planos a serem realizados. Portanto, agentes deliberativos possuem uma imagem inerente de um mundo externo e metas a serem atingidas. Assim, são capazes de produzir uma lista de ações para atingir o objetivo. Sua deliberação não se limita ao estado atual do mundo em que está situado, mas constrói hipóteses sobre possíveis estados futuros e possui informações sobre o passado;

- Híbridos: Muitos pesquisadores sugerem que nem uma abordagem completamente deliberativa nem completamente reativa é adequada para construção de agentes. Eles argumentaram o caso de sistemas híbridos, que tentam se combinar ambas as abordagens (WOOLDRIDGE; JENNINGS, 1995). Os agentes híbridos utilizam planejamento de alto nível de abstração em uma fase de pré-processamento, enquanto decisões sobre alternativas de refinamento menos significativas são tratadas por sistemas reativos.

2.2.2 Agentes BDI

O estudo de arquiteturas de agentes deliberativos baseados em estados mentais surgiu da necessidade do uso desses agentes no desenvolvimento de aplicações de tempo real em ambientes dinâmicos complexos. A estrutura do agente é definida sob uma perspectiva psicológica, que possui componentes (estados) mentais. Baseado na arquitetura de estados mentais, o modelo BDI foi proposto por Bratman (1987) como uma teoria filosófica sobre o raciocínio prático, onde o comportamento humano considerando apenas três estados mentais para a descrição do processamento interno de um agente: crença, desejo e intenção (*belief, desire, intention*).

Rao e Georgeff (2000) adotaram o modelo BDI para agentes de software e desenvolveram a arquitetura BDI, apresentando uma teoria formal e um interpretador BDI. A seguir são detalhadas as três atitudes mentais que compõem o modelo BDI:

- Crenças (*Beliefs*): As crenças representam a visão fundamental do agente em relação ao seu ambiente (CARVALHO, 2004). Um agente pode ter crenças sobre o mundo habitado por ele, sobre outros agentes, sobre interações com outros agentes e crenças sobre as suas próprias crenças;
- Desejos (*Desires*): Os desejos representam os estados desejáveis a serem atingidos (CARVALHO, 2004). Os desejos motivam o agente a realizar as tarefas para as quais ele foi projetado a executar. O agente escolhe os desejos que são possíveis de acordo com algum critério, incentivando a realizar as tarefas para as quais ele foi projetado. Um desejo causa uma intenção de agir (ação) de modo a cumprir uma determinada meta ou conjunto de metas;
- Intenções (*Intentions*): As intenções representam o atual plano de ação escolhido, correspondendo aos estados do mundo que o agente quer efetivamente provocar. Podem ser consideradas um subconjunto dos desejos, mas devem ser consistentes. Se um agente decide seguir uma meta específica, então esta meta torna-se uma intenção. As intenções determinam o processo de raciocínio prático, pois determinam as ações a serem realizadas.

O projeto de um Agente BDI envolve a determinação, prévia, dos aspectos comportamentais do agente, ou seja: deve haver algum conhecimento sobre como o agente deverá executar as suas tarefas (CARVALHO, 2004). No seu projeto são especificados as suas crenças e desejos; no entanto, a escolha das intenções é determinada pelo próprio agente, de acordo com as suas opções (desejos) disponíveis.

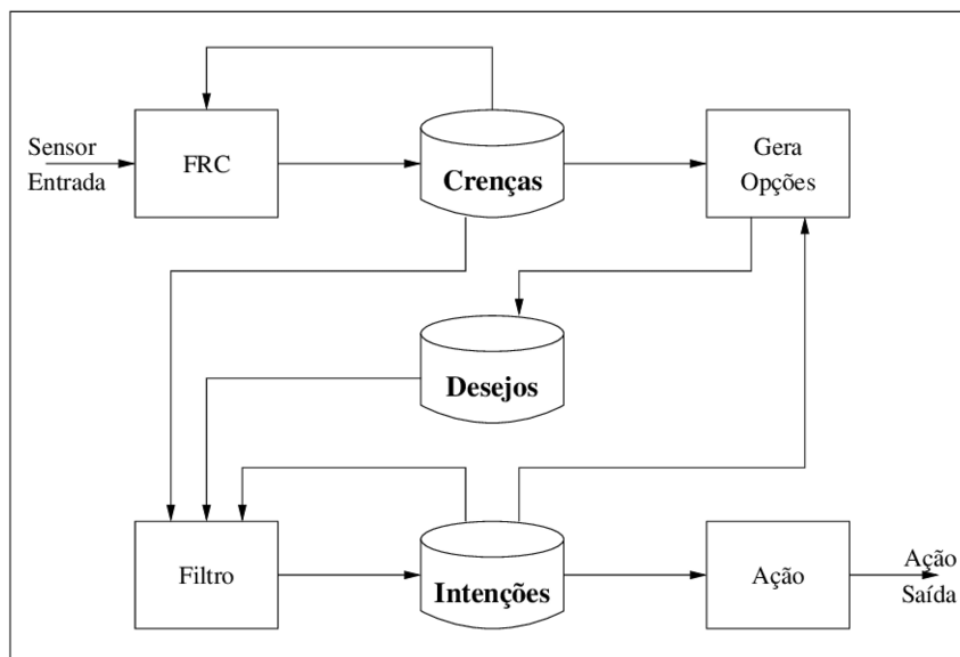
Em Weiss (2000) são definidos os seguintes componentes importantes de uma arquitetura BDI:

- Conjunto de crenças (*Beliefs*) representando as informações que o agente tem sobre seu ambiente;
- Função de revisão de crenças (*Belief Revision Function*), determina um novo conjunto de crenças a partir das entradas dos dados coletados do meio e das crenças atuais do agente;
- Conjunto de desejos (*Desires*) representando os desejos atuais do agente;
- Função geradora de opções (*Option Generation Function*), que determina as opções disponíveis para o agente, ou seja, seus desejos, tendo como base suas crenças atuais sobre seu ambiente e suas intenções atuais;
- Conjunto de intenções (*Intentions*) atuais, representando os estados que o agente pretende alcançar;
- Função filtro (Filter), representa o processo de deliberação que determina as intenções dos agentes, tendo como base suas crenças, desejos e intenções atuais;
- Função de seleção de ação (*Action Selection Function*), responsável por determinar a ação a ser executada pelo agente baseada no conjunto de intenções atuais.

2.3 SISTEMAS MULTI-CONTEXTOS

Sistemas Multi-contexto foram propostos em Inteligência Artificial e Representação do Conhecimento para modelar o intercâmbio de informações entre fontes heterogêneas (GONÇALVES; KNORR; LEITE, 2014). Tais sistemas consistem em um conjunto de contextos, cada um dos quais pode ser informalmente considerado como uma lógica e um conjunto de fórmulas escritas nessa lógica, e um conjunto de regras de ponte para transferir informações entre contextos. Assim, diferentes contextos podem ser usados para representar diferentes componentes da arquitetura e as interações entre esses componentes podem ser especificadas por meio das regras de ponte entre os contextos (BREWKA; EITER, 2007).

Figura 4 – Agente BDI



Fonte: (WEISS, 2000)

Formalmente, um sistema multi-contexto é definido como :

Seja I um conjunto finito de índices, $\{L_i\}_{i \in I}$ um conjunto de linguagens, $\{\Omega_i\}_{i \in I}$ um conjunto de conjuntos de axiomas, $\{\delta_i\}_{i \in I}$ um conjunto de regras de inferências e, $\{C_i = \langle L_i, \Omega_i, \delta_i \rangle\}_{i \in I}$ um conjunto de contextos.

Um Sistema Multi-contexto M é um par $\langle \{C_i\}_{i \in I}, \Delta_{br} \rangle$, onde $\{C_i\}_{i \in I}$ é o conjunto de contextos e Δ_{br} é o conjunto de regras de ponte sobre M (GIUNCHIGLIA; SERAFINI, 1994).

Em muitos domínios de aplicação, a adoção dos Sistemas Multi-contexto pode trazer avanços reais. Quando diferentes tipos de sistemas de informação heterogêneos estão envolvidos e uma formalização rigorosa deve ser adotada, também em vista da confiabilidade e verificabilidade dos sistemas resultantes, os Sistemas Multi-contexto são uma boa escolha (GONÇALVES; KNORR; LEITE, 2014).

A partir de uma perspectiva de engenharia de software, os sistemas multi-contexto suportam o desenvolvimento de arquiteturas modulares. Cada componente arquitetural, seja um componente funcional (responsável por avaliar a situação atual do agente, por exemplo) ou um componente de estrutura de dados (as crenças do agente, por exemplo), pode ser representado como um contexto separado. Os *links* entre os componentes podem ser explicitados escrevendo regras de ponte para ligar os contextos. Essa capacidade de suportar diretamente a decomposição de componentes oferece um caminho limpo, desde a especificação de alto nível da arquitetura até seu *design* detalhado (PARSONS et al.,

2002).

Em uma perspectiva de modelagem lógica, há quatro vantagens principais de adotar uma abordagem multi-contexto. A primeira é uma extensão das vantagens da engenharia de software que se aplica especificamente a sistemas lógicos. Ao quebrar a descrição lógica de um agente em um conjunto de contextos, cada um deles contém um conjunto de fórmulas relacionadas, obtemos efetivamente uma forma de lógica de várias ordenações (todas as fórmulas em uma única classificação de área de contexto) com as vantagens concomitantes de escalabilidade e eficiência. A segunda vantagem segue a partir disso. O uso de sistemas com vários contextos possibilita a criação de agentes que usam várias lógicas diferentes de uma maneira que mantém as lógicas separadas (todas as fórmulas de uma lógica são reunidas em um contexto). Isso permite aumentar o poder de representação dos agentes lógicos (em comparação com aqueles que usam uma única lógica) ou simplificar os agentes conceitualmente (em comparação com aqueles que usam várias lógicas em um contexto global) (PARSONS et al., 2002).

Ambas as vantagens acima se aplicam a qualquer agente lógico construído usando sistemas multi-contexto. As duas vantagens restantes se aplicam a tipos específicos de agente lógico - aqueles que raciocinam sobre suas crenças e as de outros agentes. Assim, o uso de sistemas multi-contexto facilita a execução direta das especificações do agente, onde essas especificações lidam com noções modais. A vantagem final está relacionada a isso. Agentes que raciocinam sobre crenças são frequentemente confrontados com o problema de modelar as crenças de outros agentes, e isso pode ser difícil, especialmente quando esses outros agentes raciocinam sobre as crenças de uma maneira diferente (porque, por exemplo, eles usam uma lógica diferente). Sistemas multi-contexto fornecem uma solução perfeita para este problema (PARSONS et al., 2002).

Usando uma abordagem multi-contexto, uma arquitetura de agente consiste em quatro tipos básicos de componentes (PARSONS et al., 2002):

- Unidades: entidades estruturais representando os principais componentes da arquitetura;
- Linguagens declarativas, cada uma com um conjunto de axiomas e um número de regras de inferência. Cada unidade tem uma lógica única associada a ela;
- Teorias: conjuntos de fórmulas escritas na lógica associada a uma unidade;
- Regras de ponte: Regras de inferência que relacionam fórmulas em diferentes unidades.

Unidades representam os vários componentes da arquitetura. Eles contêm a maior parte do conhecimento de solução de problemas de um agente, e esse conhecimento é co-

dificado na teoria específica que a unidade encapsula. Em geral, a natureza das unidades varia entre as arquiteturas. Por exemplo, um agente BDI pode ter unidades que representam teorias de crenças, desejos e intenções, enquanto uma arquitetura baseada em uma separação funcional de interesses pode ter unidades que codificam teorias de cooperação, avaliação de situação e execução de planos. Em ambos os casos, cada unidade possui uma lógica adequada associada a ela. Assim, a unidade de crença de um agente BDI tem uma lógica de crença associada a ele, e a unidade de intenção tem uma lógica de intenção. A lógica associada a cada unidade fornece a linguagem na qual as informações nesta unidade são codificadas e as regras de ponte fornecem o mecanismo pelo qual as informações são transferidas entre as unidades. As regras de ponte podem ser entendidas como regras de inferência com premissas e conclusões em diferentes unidades.

2.4 ARGUMENTO DO QUARTO CHINÊS

O argumento e experimento mental conhecido como “*Chinese Room Argument*” foi publicado pela primeira vez em um artigo em 1980 pelo filósofo americano John Searle. Ele foi elaborado da seguinte forma:

“Suponha que eu esteja trancado em um sala e receba um grande quantidade de escrita chinesa. Suponha, além disso, que eu não saiba chinês, seja escrito ou falado, e que eu não sou nem mesmo confiante de que eu poderia reconhecer escrita chinesa como escrita chinesa distinta de, digamos, escrita japonesa ou rabiscos sem sentido. Para mim, a escrita chinesa é igualmente numerosos rabiscos sem sentido.

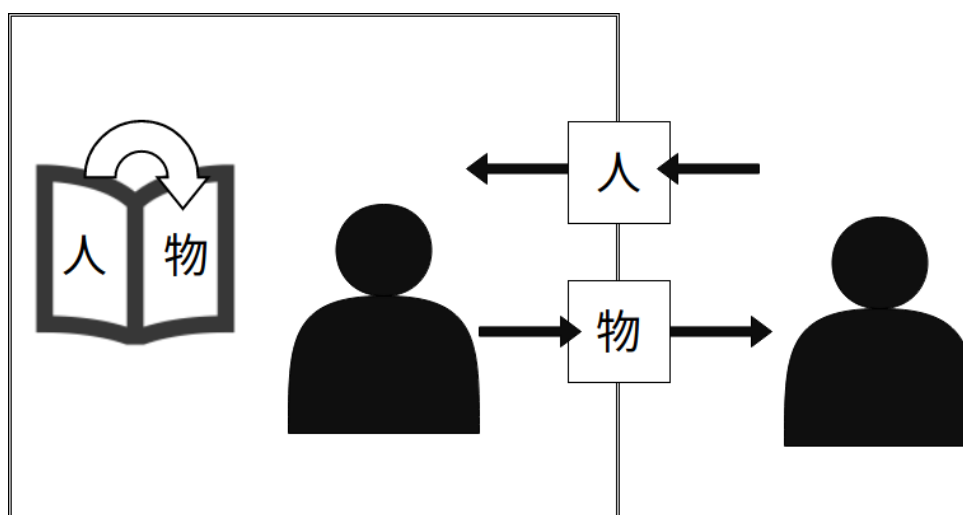
Agora suponha ainda que após este primeiro lote de escrita chinesa eu receba um segundo lote de escrita chinesa juntamente com um conjunto de regras para correlacionar o segundo lote com o primeiro lote. As regras estão em inglês e eu entendo essas regras, tão bem quanto qualquer outro nativo orador de inglês. Elas me permitem correlacionar um conjunto de símbolos formais com outro conjunto de símbolos formais, e tudo o que ‘formal’ significa aqui é que eu posso identificar símbolos inteiramente por suas formas. Agora suponha também que eu receba um terceiro lote de símbolos chineses junto com algumas instruções, novamente em inglês, que me permitam correlacionar elementos deste terceiro lote com os dois primeiros lotes, e essas regras me instruem como devolver certos símbolos chineses com certos tipos de formas em resposta a certos tipos de formas me dado no terceiro lote. Desconhecido para mim, as pessoas que estão me dando todos esses símbolos chamam o primeiro lote de ‘script’, chamam o segundo lote de ‘história’ e chamam o terceiro lote ‘perguntas’. Além disso, eles chamam os símbolos que eu lhes dou de volta em resposta ao terceiro lote ‘respostas às perguntas’ e o conjunto de regras em inglês que eles me deram, eles chamam de ‘o programa’.” (SEARLE, 1980)

O que Searle tenta demonstrar com esse experimento é que em relação aos símbolos em Chinês, ele se comporta como um computador, ou seja, ele produz as repostas apenas

por meio de operações computacionais (manipulação de símbolos), sem entender nada de chinês.

Para um observador externo, as respostas dele seriam indistinguíveis das de um falante chinês nativo, porém claramente existe uma diferença no processo mental dos dois. Searle apenas manipularia símbolos, enquanto o falante nativo teria compreensão do sentido dos símbolos manipulados. Similarmente, mesmo que um computador conseguisse produzir respostas que um ser humano para um observador externo, o processo que ocorre dentro da mente humana para produzir tais respostas seria diferente do processo de um computador.

Figura 5 – Esquema do argumento do quarto chinês.



Fonte: o autor

Searle argumenta que os programas implementados por computadores são apenas sintáticos. As operações de computador são “formais”, pois respondem apenas à forma física das sequências de símbolos, não ao significado dos símbolos. As mentes, por outro lado, têm estados com significado, conteúdos mentais (SEARLE, 1980).

Para Searle, este é o ponto-chave: a sintaxe não é por si só suficiente, nem constitutiva da semântica. Portanto, embora os computadores possam manipular a sintaxe para produzir respostas apropriadas à entrada de linguagem natural, eles não entendem as sentenças que recebem ou produzem, pois não podem associar significados às palavras:

“Contanto que o programa é definido em termos de operações computacionais em elementos formalmente definidos, o que o exemplo sugere é que estes, por eles mesmos, não têm nenhuma conexão interessante com compreensão. Eles certamente não são condições suficientes, e não há a menor razão em supor que eles são condições necessárias ou mesmo que eles fazem uma contribuição significativa para o entendimento.” (SEARLE, 1980)

Searle (1986) apresenta um argumento de que, como a sintaxe não é suficiente para a semântica, os programas não podem produzir mentes:

1. Os programas são puramente formais (sintáticos).
2. Mentes humanas têm conteúdos mentais (semântica).
3. A sintaxe, por si só, não é constitutiva nem suficiente do conteúdo semântico.
4. Portanto, os programas por si só não são constitutivos nem suficientes para as mentes.

Originalmente o alvo das críticas de Searle eram apenas as IAs “fortes”, ou seja, apenas aquelas IAs com pretensões de possuir as mesmas capacidades que uma mente humana possui. Mas como o paradigma convencional das IAs era o de sistemas simbólicos baseados em regras para manipulação desses símbolos, o argumento da sala chinês levou muitos a repensar a melhor maneira de tentar construir uma IA.

O argumento possui ainda uma conclusão mais ampla, relacionado à teoria da mente. Ele é uma crítica a teoria de que mentes humanas são computacionais ou sistemas computacionais de processamento de informação. Em vez disso, as mentes devem resultar de processos biológicos; os computadores podem, na melhor das hipóteses, simular esses processos biológicos (COLE, 2019).

2.5 PROBLEMA DO EMBASAMENTO SIMBÓLICO

O problema de embasamento (*grounding*) simbólico está relacionado ao problema de como símbolos ganham seus significados. Este problema está relacionado, consequentemente, com o problema da consciência, ou melhor, o problema de como a consciência é capaz de criar uma relação entre um sistema de símbolos formais e seus referentes (HARNAD, 2007).

Em Harnad (1990) define este problema é colocado na forma da seguinte pergunta:

“Como a interpretação semântica de um sistema formal de símbolos pode ser intrínseca ao sistema, em vez de apenas parasitar os significados em nossas cabeças? Como os significados dos símbolos sem sentido, manipulados unicamente com base em suas formas (arbitrárias), podem ser fundamentados em algo além de outros símbolos sem sentido?”

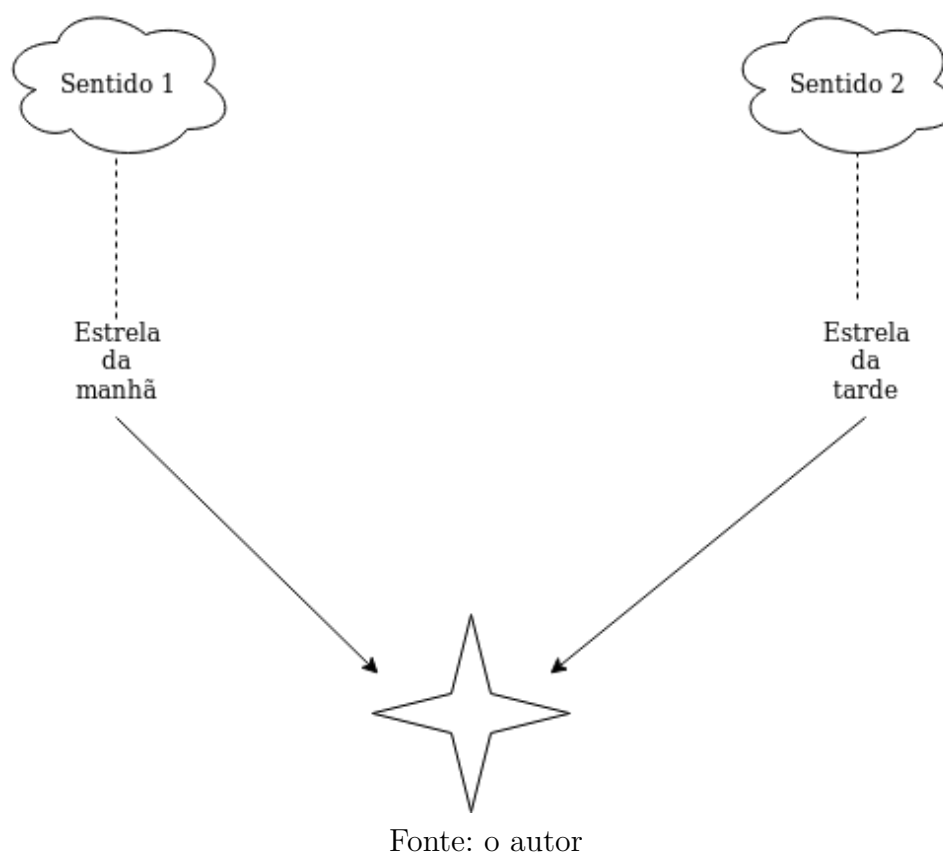
Harnad (1994) define “símbolo” como um objeto pertencente a um sistema de símbolos (a noção de símbolo isolado não é útil, segundo ele). Um sistema de símbolos é um conjunto de símbolos e regras sintáticas para manipulação destes com base em sua forma (não seu significado). Os símbolos são tratados como primitivos e não-definidos, no que se

refere às regras. Ainda assim, os símbolos e suas combinações úteis são interpretáveis de maneira significativa (HARNAD, 1994). Embora os símbolos façam sentido, esse sentido está em nossas cabeças e não no sistema de símbolos.

Os símbolos são interpretados como tendo um significado e um referente. O filósofo Gottlob Frege é responsável pela conceitualização de significado e referente. Ele propõe uma distinção entre a coisa a qual uma palavra se refere (o seu referente) e o seu significado (ou sentido) por meio da formulação de um enigma que ficou conhecido como “Enigma de Frege”.

As frases “a=a” e “a=b” possuem valores cognitivos nitidamente diferentes. A primeira é uma formulação apriorística, já a segunda possui uma ampliação de nosso conhecimento e nem sempre pode ser justificada de forma a priori. Se “a” possuísse o valor de “a estrela da manhã” e “b” o valor de “a estrela da tarde”, ambos teriam o mesmo referencial pois tanto “a estrela da manhã” quanto “a estrela da tarde” se referem à Vênus. Portanto, se quiséssemos ver a identidade como uma relação entre aquilo que se referem os símbolos, pareceria que “a=b” não poderia ser diferenciado de “a=a”, pois ambos se referem a mesma coisa (FREGE, 1997).

Figura 6 – Representação e representante.



Como esse poderia ser o caso se já foi mostrado que estas duas frases possuem valores cognitivos diferentes? A resposta está justamente na distinção que Frege faz entre

dois componentes presentes na expressão linguística: o sentido e a referência (FREGE, 1948).

A referência é como a palavra se engancha ao mundo, ou seja, como a palavra se relaciona com o mundo. O sentido de uma expressão é o modo como a referência se apresenta. A referência de “a estrela da manhã” e “a estrela da tarde” seria a mesma, mas não o sentido.

Existem muitos símbolos que não são sobre o mundo real, mas sobre abstrações de vários tipos ou sobre significados culturais e portanto não possuem referência a significados, como a palavra “água benta”, e assim eles nunca serão embasados por objetos meramente perceptivos. Iremos focar em símbolos que podem ser embasados por processos perceptivos.

Tomando o significado de uma palavra como o meio de escolher o seu referente, então, nesse caso é necessário que a entidade que contém essa palavra tenha a capacidade de seguir as regras para atribuir um referente ao símbolo (HARNAD, 1990). No caso do ser humano, a nossa mente possui tal capacidade, apesar de não sabermos exatamente como ela faz isso.

Portanto, essa dinâmica de atribuição de significado é constituído por uma relação que pode ser colocado como no modelo sugerido pelo semioticista Charles Saunders Peirce (1994) sugere para que se tenha significado:

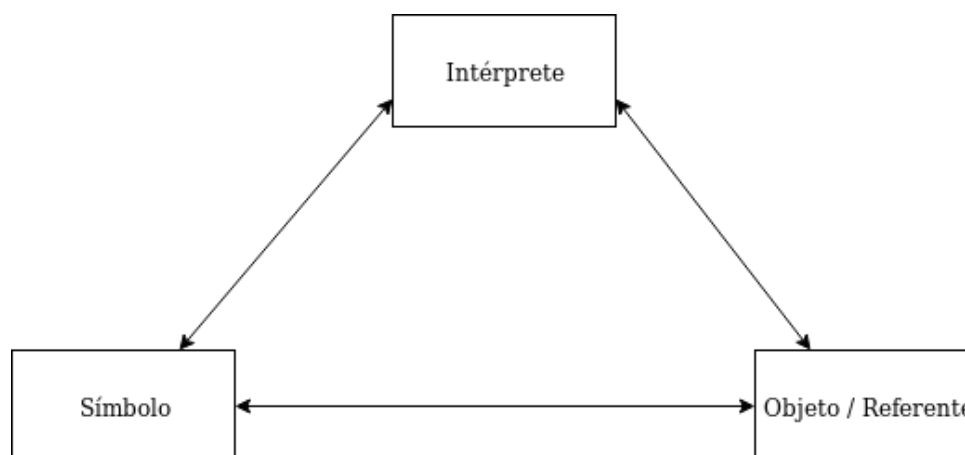
- Uma mente (intérprete)
- Um símbolo
- Um objeto (referente) externo
- Um “processo” responsável por criar uma relação entre a palavra e seu referente

Porém, e se a entidade em que a palavra está contida for um pedaço de papel?

Não existe uma conexão entre a forma simbólica de uma palavra em um pedaço de papel e o referente dessa palavra caso não haja uma mente mediando essa interação. Um pedaço de papel apenas armazena símbolos e, por si só, não compartilha da mesma propriedade presente em uma mente humana que a torna capaz de escolher referentes aos símbolos (HARNAD, 1990).

Portanto, os significados dos símbolos em um pedaço de papel não estão “embasados”, enquanto os símbolos em uma mente estão “embasados” pois existe algo (a mente) que faz o papel de mediação (por meio do significado) das formas simbólicas e seus referentes.

Figura 7 – Esquema de Peirce para significação simbólico.



Fonte: o autor

O que entendemos por “embasamento” (*grounding*), então, é a capacidade que uma entidade possui de escolher referentes para seus símbolos (HARNAD, 1990). Embasamento é uma função de entrada / saída. O embasamento conecta as entradas sensoriais de objetos externos a símbolos e estados internos que ocorrem dentro de um sistema sensório-motor autônomo, orientando o processamento e a saída resultante do sistema.

Seria um computador como um pedaço de papel, apenas armazenando símbolos, ou se assemelha a uma mente, com a capacidade de associar símbolos aos seus referentes?

Certas escolas de teoria da mente, como as teorias computacionais da mente (RESCORLA, 2017), acreditam que a mente humana funciona como um computador, ou seja, como uma máquina de manipulação de símbolos. De acordo com essas teorias, um conjunto de símbolos e de regras de manipulação desses seria suficiente para descrever o funcionamento de como a mente humana é capaz de escolher o referente de uma palavra. Essencialmente, o funcionamento da mente humana se daria na forma de um algoritmo computacional: um conjunto de regras para manipulação de símbolos.

Caso tomemos as teorias computacionais da mente como corretas, segue que, como um computador é teoricamente capaz de realizar qualquer computação, com o algoritmo apropriado, seria possível que um programa de computador funcionasse como uma mente humana.

Sistemas simbólicos, como proposto por Newell e Simon (1976), são caracterizados como um sistema de cognição de alto nível no qual os símbolos são vistos como “[situados] na raiz da ação inteligente” (NEWELL; SIMON, 1976). Eles defenderam a Hipótese dos Sistemas de Símbolo Físico, fazendo uma declaração que se alinha às teorias computacionais da mente: “um sistema de símbolo físico possui os meios necessários e suficientes para a ação inteligente geral”(NEWELL; SIMON, 1976).

Essa hipótese, portanto, define a equivalência entre os sistemas de símbolos e a ação inteligente, de tal maneira que toda ação inteligente seria originada em um sistema de símbolos. O sistema de símbolos descrito por [Newell e Simon \(1976\)](#) é visto como um programa de computador capaz de manipular entidades chamadas símbolos, “padrões físicos” combinados em expressões, que podem ser criados, modificados ou destruídos por processos sintáticos. A questão era, e muitas das críticas sobre os sistemas de símbolos vieram disso, como esses sistemas, construídos e manipulando apenas símbolos, poderiam designar algo fora de seu domínio.

O argumento do quarto chinês (discutido na seção 2.4) de [Searle \(1980\)](#) foi formulado para contestar ideias deste tipo. Tomando como verdadeira a conclusão de Searle de que um sistema simbólico, por si só, não possui a capacidade de selecionar referentes para um símbolo, conclui-se que a capacidade de escolher referentes para símbolos não é puramente computacional, portanto está além da capacidade das manipulações meramente simbólicas de um computador.

Aqui dois aspectos são importantes: primeiro, os símbolos não representam nada para um sistema, pelo menos não o que eles dizem “designar”. Somente alguém operando o sistema pode reconhecer esses símbolos como referindo-se a entidades fora do sistema. Segundo, o sistema de símbolos não pode conter seu fechamento ao relacionar símbolos apenas com outros símbolos; alguma outra coisa deve ser necessária para estabelecer uma conexão entre os símbolos e o que eles representam.

A busca por processos de embasamento de símbolos diz respeito à compreensão de processos que possibilitem a conexão dessas representações puramente simbólicas com o que elas representam de fato, o que poderia ser diretamente, ou por meio de outras representações fundamentadas.

2.6 SOLUÇÕES PARA O PROBLEMA DO EMBASAMENTO SIMBÓLICO

A seguir serão discutidos os principais trabalhos que compõem o estado da arte em termos de propostas para resolver o problema de embasamento de símbolos.

Os trabalhos discutidos foram selecionados por seu papel influente em diversas linhas de pesquisa e/ou por sua natureza representativa a uma classe de soluções.

As soluções foram classificadas em três abordagens, uma por seção: representacionalismo, semi-representacionalismo e não-representacionalismo. Todas as abordagens procuram embasar os símbolos através das capacidades sensório-motoras do agente artificial envolvido. As estratégias diferem nos métodos utilizados para elaborar os dados obtidos a partir das experiências sensório-motoras, e no papel atribuído à elaboração das

representações de dados, no processo de geração da semântica para os símbolos.

Tabela 2 – Esquema de classificação das soluções para resolução do problema do embasamento simbólico

Abordagem	Nome	Descrição	Artigo	Seção
Representacionalista	Modelo Híbrido	Modelo híbrido, que implementa uma mistura de características de um sistema simbólico e de um sistema conexionista.	(HARNAD, 1990)	2.6.1.1
	Hipótese do Roubo Simbólico	Novas categorias aprendidas por meio combinações de símbolos que descrevem a nova categoria.	(CANGELOSI; GRECO; HARNAD, 2000)	2.6.1.2
Semi-representacionalista	Modelo Epistemológico	O conceito deve ser uma descrição completa do objeto a qual se refere, e portanto deve coletar diferentes tipos de representações.	(DAVIDSSON, 1994)	2.6.2.1
	Problem Físico do Embasamento Simbólico	Embasar um sistema simbólico de um agente nas atividades sensório-motoras e transformar o problema de embasamento de símbolos no “problema físico de embasamento de símbolos”.	(VOGT, 2001)	2.6.2.2
	Semântica Baseada em Ação	Nova teoria de significado - semântica baseada em ação - e um novo tipo de agente - os agentes de duas máquinas.	(FLORIDI, 2011)	2.6.2.3
Não representacionalista	Arquitetura de Subsunção	Arquitetura de subsunção acopla a informação sensorial à seleção da ação, evitando qualquer elaboração de representações explícitas.	(BROOKS, 1986)	2.6.3.1
	Modelo Baseado em Comportamento	Desenvolvimento de capacidades semânticas em um agente com base no desenvolvimento das capacidades linguísticas de crianças, como uma ferramenta com a qual interagem com seu ambiente e com outros agentes.	(VARSHAVSKAYA, 2002)	2.6.3.2

Fonte: o autor

2.6.1 Representacionalista

Em linhas gerais, as abordagens representacionalistas buscam resolver o problema do embasamento de símbolos ancorando os símbolos às representações decorrentes das manipulações dos dados perceptivos de agentes. Essas abordagens consideram as representações conceituais e categóricas, elaboradas por um agente, como o significado dos símbolos.

Segundo Floridi (2011), para abordagens representacionalistas, um agente deveria ser capaz de:

1. Capturar algumas características salientes compartilhadas por conjuntos de dados perceptivos
2. Abstraí-las dos conjuntos de dados

3. Identificar as abstrações como conteúdo de representações categóricas e conceituais
4. Usar essas representações para embasar seus símbolos

2.6.1.1 Modelo Híbrido

No mesmo artigo em que Harnad define o problema do embasamento de símbolos, ele também oferece uma possível solução para o problema. Essa proposta estabeleceu o padrão para todas as estratégias posteriores.

A estratégia sugerida por Harnad (1990) é baseada em um modelo híbrido, que implementa uma mistura de características de um sistema simbólico e de um sistema conexionista. Com isso, Harnad tenta superar os limites típicos encontrados pelos sistemas simbólicos e conexionistas puros.

Por um lado, em um modelo puramente simbólico faltam as conexões entre os símbolos e seus referentes. Um sistema simbólico, embora passível de interpretação sintática sistemática, não está embasado.

Por outro lado, embora sistemas conexionistas tornem possível conectar símbolos e referentes usando os dados perceptivos e os recursos das categorias representacionais, eles não podem manipular símbolos (como os sistemas de símbolos podem fazer facilmente) para produzir uma interpretação intrínseca, sistemática e finita dos mesmos.

Isso justifica a solução proposta por Harnad, que, devido à sua natureza semi-simbólica pretende oferecer o melhor dos dois mundos.

Além dessa justificativa computacional, Harnad baseia sua estratégia em uma teoria psicológica que assume a habilidade de construir categorias do mundo como base para a linguagem e cognição. Segundo ele:

“Nós já sabemos o que os seres humanos são capazes de fazer. Eles podem (1) discriminar, (2) manipular, (3) identificar e (4) descrever os objetos, eventos e estados de coisas no mundo em que vivem, e eles também podem (5) ‘produzir descrições’ e (6) ‘responder a descrições’ desses objetos, eventos e estados de coisas.” (HARNAD, 1990)

De acordo com a proposta de Harnad, os símbolos manipulados por um agente podem ser embasados conectando-os aos dados perceptivos que eles denotam. A conexão é estabelecida por um processo de categorização dos sinais sensorio-motores. Com base nas capacidades humanas já mencionadas, Harnad propõe que os símbolos devem ser fundamentados em três etapas:

1. Iconização: o processo de transformação de padrões de dados sensoriais em representações icônicas (equivalentes análogos internos das projeções de objetos sobre as superfícies sensoriais do agente)

2. Discriminação: o processo de julgar se duas entradas são as mesmas ou, se são diferentes, o quanto elas diferem
3. Identificação: o processo de atribuir uma resposta única - isto é, um nome - a uma classe de *inputs*, tratando-os como equivalentes ou invariantes em algum aspecto.

Os dois primeiros estágios produzem representações sub-simbólicas, enquanto o terceiro estágio embasa os símbolos.

As representações icônicas em (1) são obtidas a partir do conjunto de todas as experiências relacionadas às percepções do mesmo tipo de objeto. As representações categóricas são então obtidas através do processo de discriminação em (2). Essa delimitação de categorias seria feita por meio de uma análise de grau de congruência entre as representações icônicas. Uma vez elaboradas, as representações categóricas são associadas em (3) com classes de símbolos (os nomes), provendo assim estes com referências apropriadas que os fundamentam.

Iconização e discriminação são subprocessos, realizados por meio de redes neurais. Elas tornam possível a associação de um nome com uma classe de entrada e posteriormente, a nomeação dos referentes. No entanto, por si só, as redes neurais são incapazes de produzir representações simbólicas, então elas ainda não podem permitir que o agente desenvolva capacidades simbólicas. A fim de contornar essa falha, Harnad fornece ao seu modelo híbrido um sistema simbólico, que pode manipular símbolos sintaticamente e, graças às categorias produzidas pelo sistema conexionista, finalmente consegue atingir um embasamento semântico para seus símbolos.

O problema dessa abordagem para Floridi (2011) diz respeito ao modo como o sistema híbrido deve encontrar as características invariantes de suas projeções sensoriais que permitem categorizar e identificar objetos corretamente.

Para ilustrar a deficiência da solução apresentada por Harnad, Floridi (2011) propõe que imaginemos, por exemplo, um agente que implemente o modelo híbrido, chamado PERC. Inicialmente, PERC não possui conteúdo ou recursos semânticos, portanto, não possui compromisso semântico. Assumindo que PERC está equipado com uma maneira de adquirir alguma entrada de dados, uma câmera de vídeo, através da qual observa seu ambiente externo. Seguindo Harnad, suponha que, por meio de sua câmera e redes neurais, PERC é capaz de produzir algumas representações icônicas a partir dos dados perceptivos que recolhe do ambiente. PERC deveria então desenvolver representações categóricas destes dados perceptivos, considerando apenas as características invariantes das representações icônicas (FLORIDI, 2011).

Em seguida, deveriam ser organizadas as representações categóricas em categorias conceituais como, por exemplo, “animal quadrúpede”. A pergunta a ser feita é: de onde

as categorias conceituais, como “animal quadrúpede”, se originam?

Redes neurais podem ser usadas para encontrar estruturas (se existirem) nos dados, como padrões de dados. No entanto, se forem supervisionadas então elas são treinadas por meio de um conjunto de treinamento pré-selecionado e feedback repetido, então qualquer embasamento que eles possam fornecer é inteiramente extrínseco.

Se elas são treinadas sem supervisão, as redes implementam algoritmos de treinamento que não usam o dados de saída, mas dependem apenas dos dados de entrada para tentar encontrar estruturas no espaço de dados de entrada. Unidades na mesma camada competem entre si para serem ativadas. No entanto, elas ainda precisam ter vieses integrados e detectores de recursos internos para alcançar a saída desejada. Tais recursos são necessariamente codificados por um supervisor, de acordo com critérios estabelecidos (FLORIDI, 2011).

Além disso, uma vez que redes não supervisionadas foram treinados, ainda precisam ter sua saída verificada para ver se os resultados fazem algum sentido em relação ao espaço de dados de entrada. Este processo de validação é realizada externamente por um supervisor. Então, neste caso também, seja qual for o embasamento que eles podem fornecer ainda é totalmente extrínseco (FLORIDI, 2011).

No exemplo, “animal quadrúpede”, como categoria não é o resultado do embasamento intrínseco do PERC porque este já deve ter tido ajuda semântica para alcançar essa conclusão. A estratégia posta em Harnad (1990), na verdade, pressupõe a disponibilidade de recursos semânticos que se espera que o agente seja capaz de desenvolver do zero, através de suas interações com o ambiente e outro agentes e das elaborações de seus dados perceptivos.

2.6.1.2 Hipótese do Roubo Simbólico

Com base em uma mesma teoria psicológica que vê a habilidade de construir categorias do mundo como uma base para a cognição, Cangelosi, Greco e Harnad (2000) expandem a possibilidade do desenvolvimento de novas categorias por meio da noção de linguagem como algo que não é isolado na capacidades de um indivíduo único, mas que é intrinsecamente relacionado com as habilidades sociais de um agente.

A explicação da origem e evolução da linguagem para esta abordagem é baseado no híbrido das capacidades simbólico/sensorimotor implementadas pelo sistema. Inicialmente, organismos desenvolvem a habilidade de construir categorias do mundo por meio direto das suas capacidades perceptivas. Então, alguns organismos experimentam combinações proposicionais dos nomes de algumas dessas categorias e descobrem a vantagem dessa nova forma de aprender categorias.

Em Cangelosi, Greco e Harnad (2000) é diferenciado duas maneiras de adquirir

novas categorias:

A primeira abordagem, como já vista em [Harnad \(1990\)](#), é por meio da interação de agentes, por meio de seus sensores, com o seu ambiente e na transformação de percepções de um agente em uma categoria e subsequentemente em símbolos embasados. Essa abordagem foi denominada como abordagem de trabalho sensorimotor (*sensorimotor toil*).

A segunda abordagem foi chamada de roubo simbólico (*symbolic toil*). Nela, novas categorias são aprendidas por meio de proposições - combinações de símbolos que descrevem a nova categoria. Essa nova abordagem ainda necessita do trabalho sensório-motor para embasar uma certa quantidade de categorias iniciais. Uma vez que tenham sido embasadas uma quantidade inicial de categorias, estas podem então ser combinadas para criação de novas categorias sem a necessidade da interação do agente com o ambiente por meio de suas atividades sensório-motoras. A nova categoria é então meramente descrita pelas proposições compostas dos símbolos já embasados.

Categorias embasadas diretamente pelo trabalho sensório-motor possuem representações icônicas e categóricas, enquanto categorias embasadas indiretamente pelo roubo simbólico possuem representações simbólicas consistindo de descrições proposicionais.

Qualquer combinação de símbolos irá herdar o embasamento semântico de seus símbolos mais elementares que compõem a proposição. Esse tipo de aprendizado, segundo [Cangelosi, Greco e Harnad \(2000\)](#), emula o aprendizado de novos conceitos que ocorre em humanos por meio de definições puramente linguísticas. Considerando, por exemplo, o caso que não saibamos o que uma “zebra” seja, mas que estamos familiarizados com o que é um “cavalo” e um “padrão listrado”, pois já os vimos realmente (trabalho sensório-motor). Nesse caso, temos dois nomes que representam sua categoria: “cavalo” representa a categoria dos cavalos e “listras” representa a categoria dos padrões listrados. Suponha que somos introduzidos a definição linguística de zebra como: “zebra” = “cavalo” + “listras”. Assim sendo, podemos imediatamente entender que o símbolo “zebra” deve corresponder a uma combinação de (a representação categórica de) cavalos com (a representação categórica de) listras. Além disso, quando virmos uma zebra individual, seremos capaz de identificar que ela faz parte da categoria zebra que nos foi introduzida linguisticamente. Esse exemplo mostra como podemos aprender novas categorias e nomes embasados por meio da combinação de categorias básicas e nomes já embasados e é por meio desse princípio que funciona o roubo simbólico.

Conseguir categorias por roubo simbólico, então, é consegui-las na base do conhecimento fundamentado dos outros, transferidos através de proposições simbólicas cujos termos - todos menos um - já estão fundamentados para nós também. Essa nova maneira de adquirir categorias poupa o agente de muito trabalho sensório-motor.

Para Cangelosi, Greco e Harnad (2002), a vantagem da utilização do método de roubo simbólico é que este dá aos agentes a vantagem adaptativa das habilidades de linguagem natural. Isto é infinitamente superior ao método puramente sensorimotor, apesar de ainda fundamentado e dependente dele (CANGELOSI; GRECO; HARNAD, 2002).

O problema, para Floridi (2011), é o ainda crucial problema de como organismo conseguem semantizar os dados resultados de suas atividades sensorio-motoras, e portanto o problema do embasamento de símbolos continua aberto.

2.6.2 Semi-representacionista

As abordagens semi-representacionistas, como a de Davidsson (1994) e (VOGT, 2001), ainda são representacionista em sua natureza mas se diferenciam desta por lidar com o uso das representações do agente com base em princípios utilizados em robôs baseados em comportamento (*behavior-based robots*).

2.6.2.1 Modelo epistemológico

Davidsson (1994) concorda com (HARNAD, 1990) que um sistema puramente simbólico não é suficiente para resolver o problema do embasamento de símbolos, mas não concorda que redes conexionistas são suficientes para servir a função de aprendizagem de características invariantes das representações categóricas e de conectar nomes aos ícones das entidades a qual pertencem.

Para Davidsson, o que a proposta de Harnad deixa não respondido é: “que tipo de aprendizagem é permitida pelas redes neurais?” (DAVIDSSON, 1994).

O tipo de aprendizagem de um agente possui restrições impostas pelo mundo real. Davidsson (1994) defende que mesmo que os modelos de alguns objetos possam ser pré-programados, um agente interagindo com o mundo real provavelmente irá encontrar objetos que não se encaixam ao seu modelo. Além disso, o agente provavelmente não irá encontrar todas as instâncias de uma categoria de uma só vez, ele irá encontrar uma instância de vez em quando e deve ser capaz de incorporá-lo ao conjunto de conhecimento de conceitos. Portanto, conceitos devem ser aprendidos de maneira gradual, por meio de repetidas interações com o ambiente ao longo do tempo.

Redes neurais fornecem uma aprendizagem discriminativa que não é voltada para uma fácil adaptação incremental de seus conteúdos, devido a estrutura fixa das redes neurais. Segue que, para Davidsson (1994), a maioria das redes neurais não são apropriadas para o tipo de aprendizagem requerida por um agente que queira lidar com o problema de embasamento de símbolos.

Para Davidsson (1994), o problema de embasamento de símbolos deve ser visto em termos de representações de conceitos gerais e aprendizagem de máquinas. A ideia

principal é que o conceito deve ser uma descrição completa do objeto a qual se refere, e portanto deve coletar diferentes tipos de representações. Para ele, devem haver diferentes tipos de representações para diferentes tipos de propósitos, uma única descrição não é suficiente para prover todas as funcionalidades desejadas.

Segundo Davidsson uma descrição é composta por três partes:

- Designador: é o nome (símbolo) utilizado para se referir à categoria;
- Representação epistemológica: é usado para reconhecer instâncias de uma categoria;
- Representação inferencial: é uma coleção de conhecimento “enciclopédico” sobre a categoria e seus membros, uma coleção que pode ser usada para inferir informações não-perceptíveis e fazer previsões.

Por exemplo, para a categoria “cadeira”, o designador é nome “cadeira” na língua portuguesa e sua representação epistemológica é algum tipo de objeto modelo em 3D de uma cadeira típica. Além dessas, teríamos o conhecimento “enciclopédico”, a representação inferencial, que pode conter informações como: cadeiras podem ser usadas para sentar, geralmente feitas de madeira etc

Um objeto que é percebido pelo sistema de visão do agente é combinado a uma representação epistemológica. Isso ativa uma estrutura de conhecimento maior, a representação composta do conceito, da qual a representação epistemológica faz parte. Essa estrutura contém, dentre outras coisas, o designador.

Davidsson (1994) reconhece que as representações que fundamentam os símbolos devem ser não pré-programado, mas sim aprendido pelo agente a partir de sua própria “experiência”. Então ele sugere o uso de dois paradigmas típicos de aprendizagem de máquinas: aprendendo por observação e aprendendo com exemplos.

Aprendizagem por observação é um mecanismo de aprendizagem não supervisionado, que permite o sistema gerar descrições de categóricas. Exemplos não são pré-classificados e o sistema deve ser capaz de formar categorias de maneira autônoma. Porém, o programador ainda provê ao sistema um número específico de descrição de entidades bem selecionadas, que permite ao agente agrupar as entidades em categoriais

Claramente, as descrições significantes primeiro selecionadas e, em seguida, fornecido pelo treinador humano para a aprendizagem artificial é uma condição essencial para qualquer outra categorização das entidades tratadas pelo agente. São também uma condição *sine qua non* para a solução do problema de embasamento de símbolos. Como tais descrições são fornecidas antes que o agente desenvolva suas capacidades semânticas e antes que comece a elaborar qualquer tipo de descrição de forma autônoma, eles são

totalmente externos ao agente e representam um recurso semântico dado ao agente pelo programador.

A mesma objeção se aplica à aprendizagem por meio de exemplos. Nesse caso, a presença de critérios externos é ainda mais óbvio, dado que esse tipo de aprendizagem pré-supõe um conjunto de exemplos de categorias explicitamente pré-classificadas. Portanto, a estratégia de Davidsson é semanticamente comprometida e para Floridi (2011) também falha como possível solução para o problema e embasamento de símbolos.

2.6.2.2 Problema físico de embasamento de símbolos

Vogt (2001) conecta a solução proposta por Harnad (1990) com robótica situada (BROOKS, 1990; BROOKS, 1991) e com a definição semiótica de símbolos proposta por Peirce (1994).

Sua estratégia consiste em abordar o problema do embasamento de símbolos por meio do conceito de cognição incorporada (*embodied cognition*). Ele busca embasar o sistema simbólico de um agente nas atividades sensório-motoras e transformar o problema de embasamento de símbolos no “problema físico de embasamento de símbolos”, e então resolver este por meio de duas ferramentas conceituais: o símbolo semiótico e o jogo da adivinhação.

Tradicionalmente, cientistas cognitivos descrevem a cognição em termos de sistemas simbólicos (NEWELL; SIMON, 1976). Eles assumem que os agentes cognitivos manipulam símbolos quando eles pensam, raciocinam ou usam linguagens. Porém, ao explicar os processos subjacentes à essas funções cognitivas em termos de manipulação de símbolos enfrentamos o problema do embasamento de símbolos. Esse problema aparece pois símbolos são definidos como representações internas, que deveriam se relacionar com entidades no mundo real.

Uma abordagem alternativa das ciências cognitivas tenta superar esse problema descrevendo cognição em termos das dinâmicas das interações de um agente com o mundo. Essa abordagem foi chamada de de ciência cognitiva incorporada (PFEIFER, 1999). Nessa abordagem, a inteligência pode ser descrita em termos das experiências corpóreas do agente, que são adquiridas pelas suas interações com o ambiente. Dessa forma, o problema do embasamento de símbolos seria evitado até certo nível pois não seria mais necessário o uso de representações simbólicas para implementar comportamento inteligente.

Para Vogt (2001), apesar de que muito pode ser feito sem o uso de descrições simbólicas, formas mais sofisticadas de cognição ainda são melhores descritas em termos de sistemas simbólicos. No entanto, para superar o problema de embasamento de símbolos, o sistema simbólico deve ser incorporado e situado.

Deve ser incorporado para poder ter experiências do mundo e situado para que

possa adquirir seu conhecimento por meio de interações com o mundo. A questão fundamental levantada por Vogt (2001) é: seria possível definir e desenvolver um sistema simbólico incorporado e situado?

Para Vogt (2001), isso é possível adotando a definição triádica de símbolos proposta por Peirce. Esses seriam os chamados símbolos semióticos, compostos por:

- Forma (*representamen*): a forma tomada pelo signo
- Significado (*interpretant*): o conteúdo semântico do signo
- Referente (*object*): o objeto ao qual o signo refere

De acordo com Peirce (1994), o significado do símbolo semiótico surge em sua interpretação. Como tal, o significado surge do processo de semiose, que é a interação entre forma, significado e referente. Isso quer dizer que o significado depende de como o símbolo semiótico é construído e com qual função. Então, o significado de um símbolo semiótico pode ser visto como uma relação entre a forma e um referente. Essa relação é baseada na experiência e interação corporal do agente com o referente.

A experiência de um agente é baseado no histórico de suas interações. Cada interação entre um agente e um referente pode ativar suas experiências passadas criando novas experiências. A maneira que as experiências são memorizadas e representadas formam a representação interna do significado. A interação entre um agente e o referente define a relação funcional.

Usando a definição de Peirce de símbolos, Vogt (2001) tenta mostrar que os símbolos que constituem um sistema de símbolos semióticos de um agente já estão semanticamente embasados devido a sua natureza intrínseca. Já que tanto o significado quanto o referente já estão incorporados à definição do símbolo. A fundamentação de todo sistema de símbolos semióticos é então deixado para tipos especiais de agentes que podem embasar os significados de seus símbolos em suas atividades sensório-motoras, resolvendo assim o problema físico de embasamento de símbolos.

A solução para o problema físico de embasamento de símbolos é baseado em um jogo de adivinhação, uma técnica usado para estudar o desenvolvimento de uma linguagem comum. De acordo com Vogt (2001), o jogo da adivinhação torna explícito os significados dos símbolos e permite que eles sejam embasados pelas percepções e interações dos agentes. Se o jogo da adivinhação termina com sucesso, então o problema físico do embasamento de símbolos estaria resolvido.

O jogo da adivinhação envolve dois robôs, situados em um ambiente comum. Cada robô possui um papel: o falador nomeia os objetos que percebe, o ouvinte tem a tarefa

de encontrar os objetos nomeados por meio de tentativa e erro. Durante o jogo, os participantes desenvolvem um sistema comum de símbolo semióticos por meio das interações comunicativas. O jogo termina com sucesso se os dois robôs desenvolverem um léxico compartilhado, embasado nas interações entre eles e seu ambiente.

O jogo possui quatro estágios. As primeiras duas fases consistem em começar as atividades perceptivas dos robôs no ambiente e a seleção de qual parte do ambiente os robôs irão operar. As duas últimas fases fazem parte da formação de significado. Mais especificamente, elas constituem o jogo da discriminação, por meio da qual categorias são elaboradas, e o jogo da nomeação, pelo meio da qual categorias são nomeadas. Esses dois estágios permitem que os robôs encontrem um referente para seus símbolos.

Para embasar seus símbolos, os agentes envolvidos no jogo da adivinhação tem que categorizar os dados obtidos da percepção de um objeto, de maneira que eles possam posteriormente distinguir essa categoria de objetos das outras. De acordo com [Vogt \(2001\)](#), o processo de formação de significado é feito pelo jogo da discriminação. Durante esse processo, o agente associa dados perceptivos similares para elaborar suas representações categóricas. Uma vez que o agente tenha elaborado uma categoria para cada objeto percebido, o jogo da nomeação começa. Durante esta última fase, os agentes se comunicam para indicar o objeto que eles categorizar. O falante faz uma enunciação que funciona como o nome de uma das categorias que ele elaborou. O ouvinte tenta interpretar o enunciado e o associar com uma das categorias que ele próprio elaborou. O objetivo é identificar a mesma categoria nomeada pelo falante. Se o ouvinte encontra a interpretação correta para o enunciado do falante, então os dois agentes são capazes de se comunicar e o jogo da adivinhação foi um sucesso.

[Floridi \(2011\)](#) apresenta duas críticas a essa abordagem. Uma delas está relacionada com a abordagem semiótica, a outra com o que o jogo da adivinhação realmente prova.

Símbolos satisfazem a definição semiótica - possuem forma, significado e referente - apenas se eles são interpretados por um agente que já possua semântica. Essa era inclusive a visão de Peirce. Signos são apenas símbolos com significados na mente de um intérprete. Não se pode assumir que o agente qualifica como um intérprete sem cometer o erro de petição de princípio. Por isso, para [Floridi \(2011\)](#), o problema do embasamento simbólico não pode ser reduzido para o problema físico do embasamento simbólico: o agente não possui uma semântica intrínseca, autonomamente elaborada, portanto não se pode fazer o próximo passo de ancorar a semântica dos símbolos semióticos ao ambiente, pois não há nada para ser ancorado.

O jogo da adivinhação também não pode conectar os símbolos aos seus referentes externos por meio de interações dos robôs com o ambiente, pois, como é admitido pelo próprio [Vogt](#), o jogo da adivinhação não tem a função de embasar símbolos. O jogo da

adivinhação assume que os agentes manipulam símbolos previamente embasados. Seu objetivo é mostrar como dois agentes podem vir a compartilhar um mesmo vocabulário embasado por meio de um processo de comunicação.

2.6.2.3 Semântica Baseada em Ação

Floridi (2011) propôs uma nova solução para o problema do embasamento simbólico que ele chama de práxica. A abordagem práxica é baseada em dois componentes principais: uma nova teoria de significado - chamada de semântica baseada em ação (*action-based semantics*) - e um novo tipo de agente - os agentes de duas máquinas - que implementa a semântica baseada em ação graças à sua arquitetura. Os dois componentes permitem que os agentes desenvolvam habilidades semânticas suficientes para satisfazer os requisitos para superar o problema de embasamento simbólico. Floridi (2011) também defende o uso de um processo de evolução evolutiva para que o agente possa desenvolver habilidades semânticas complexas.

A distinção entre símbolos e significado é uma diferença crucial entre a solução práxica e as abordagens revisadas anteriormente. Outras tentativas para resolver o problema do embasamento de símbolos considera o significado e o símbolo como dois aspectos dos mesmos dados. Assim, um agente deveria elaborar um conjunto de dados perceptivos para obter uma representação que é tanto o significado quanto o símbolo que é usado para nomear essa representação. A solução práxica, por outro lado, trata significado e símbolo como dois tipos de dados independentes: o primeiro é dado diretamente toda vez que um agente de duas máquinas interage com o ambiente, enquanto o segundo é produzido por uma das máquinas específicas desse agente.

Floridi (2011) aponta que a semântica elaborada por um agente de duas máquinas é significativamente simples, porém, Floridi (2011) defende que por meio de processos evolucionários e processos sociais onde os agentes possuam habilidades de comunicação, a semântica pode escalar seu grau de complexidade.

A solução práxica de Floridi (2011) pode ser dada em dois passos:

O primeiro passo consiste em delinear a abordagem apropriada envolvida no processo de geração de novos significados. Isso é definido como semântica baseada em ação. A semântica baseada em ação requer uma explicação do processo específico que permite o acoplamento de símbolos aos significados.

O segundo passo, consiste em descrever um agente de duas máquinas que implementa a semântica baseada em ação. Um agente de duas máquinas atribui significados a símbolos sem ainda elaborar qualquer tipo de representação categórica. O segundo estágio do processo semântica concerne como um agente de duas máquinas gera representações. Estas não são nem categóricas nem conceituais, ao contrário de Harnad (1990), e ainda

permitem o desenvolvimento de uma semântica na qual os símbolos podem ser nomes de classes de significados.

A ideia básica de uma semântica baseada em ação é simples: no início, o significados dos símbolos gerados por um agente são os estados internos daquele agente, que por sua vez estão diretamente correlacionados com as ações realizadas pelo mesmo agente.

Considere um agente comum, como um robô capaz de se mover em um laboratório. Floridi (2011) chama esse suposto agente de FOTOC. Sempre que o FOTOC executa um movimento, como “virar à esquerda”, ele entra em um estado interno específico e deve ser capaz de aproveitar esse estado interno como um significado a ser associado a um símbolo. Assim, dizendo que as ações executadas são os significados dos símbolos, Floridi quer dizer que o agente relaciona seus símbolos aos estados em que é colocado pelas ações que ele executa e que os símbolos são considerados os nomes das ações por meio dos estados internos correspondentes.

Por meio da semântica baseada em ação, um agente pode gerar significados sem seus dados perceptivos (a detecção da FOTOC de sua localização no escritório do laboratório) causando algum tipo de representação, um processo que é sempre baseado em critérios semânticos. Os estados internos do agente são excelentes candidatos para o papel de recursos não semânticos, mas que induzem a semântica.

Seguindo a semântica baseada em ação, evita-se o uso de qualquer tipo de assistência externa. É certo que, na maioria das vezes, um agente executa uma ação para atingir algum objetivo, mas essa forma de comportamento teleológico não é o que está envolvido na semântica baseada em ação. A semântica baseada em ação pressupõe que a ação executada - e não a meta a ser alcançada - por um agente vai basear seus símbolos semanticamente.

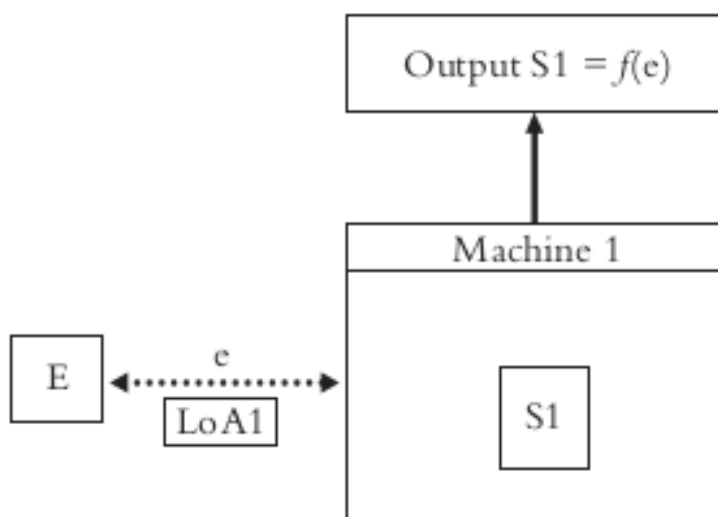
Na semântica baseada em ação, o significado é o estado interno do agente, um estado acionado pela ação correspondente. Neste estágio, nenhuma interação semântica com outros agentes ainda está em exibição.

De acordo com a semântica baseada em ação, a associação inicial de símbolos e significados é uma relação direta de entrada-saída, que decorre apenas do desempenho das ações. Um agente individual associa automaticamente um significado a um símbolo por meio do desempenho de uma ação, sem considerar ainda o quadro no qual ele executou essa ação e sem levar em consideração ainda a associação realizada por outros agentes. O componente social surge somente após a associação ter ocorrido. Em outras palavras: de acordo com a semântica baseada em ação, a semântica tem suas raízes iniciais nos comportamentos individuais do agente, não na comunidade.

O tipo de agente que Floridi (2011) chama de agente de duas máquinas é constituído por máquinas - M1 (fig. 8) e M2 (fig. 9) - que interagem entre si e executam ações

em dois níveis. O M1 opera no Nível de Objeto, interagindo diretamente com o ambiente externo, navegando, detectando obstáculos, evitando-os, etc., produzindo e inserindo ações. O M2 opera no Nível de Metaprogramação e o alvo de suas elaborações são os estados internos do M1. O nível M1 é sobre como a ação (ou melhor, os estados internos relacionados às ações) está relacionada ao mundo, enquanto o nível M2 é sobre como os símbolos estão relacionados às ações. Qualquer ação que saia de M1 para o ambiente define um estado interno particular (S_n) de M1. Assim, ações e estados internos são causalmente acoplados: para qualquer ação diferente no M1 existe um estado interno diferente S_n , e para todas as ações similares no M1 existe o mesmo S_n (fig. 10).

Figura 8 – Estrutura da M1: E é o ambiente, S_1 é o estado interno de M1, LoA_1 é o nível de abstração no qual a M1 interage com E, $f(e)$ é a função que identifica S_1 , onde (e) é uma dada interação entre o agente e o ambiente.



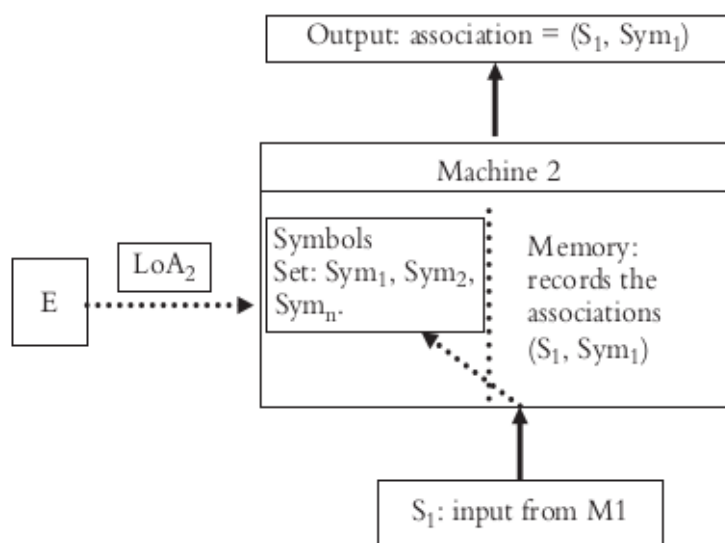
Fonte: (FLORIDI, 2011)

Podemos agora aplicar o método de abstração para descrever o grau de refinamento das percepções do M1. O M1 acessa o ambiente em um nível de abstração que permite apenas uma granularidade específica de detecção de seus recursos. Assim, através das percepções de M1, FOTOC só pode obter dados aproximados (para qualquer grau de granularidade que é implementado) sobre seu ambiente externo.

Seguindo a arquitetura de metaprogramação, M1 se comunica com a outra máquina, M2. M1 envia seu estado interno (não interpretado) para M2. M2 é um fabricante e retentor de símbolos. Ele é constituído por uma fonte de símbolos, um espaço de memória e um conjunto de símbolos. As duas máquinas comunicam seus dados em seus respectivos níveis de abstração. M2 lê os estados de M1 de acordo com seu nível de abstração, que é

menos refinado que o nível de abstração de M1. Por causa da granularidade de M2, M2 não lê S_n como foi enviado por M1; em vez disso, S_n é modificado pelo nível de abstração de M2 de tal maneira que o novo estado é mais genérico. Em outras palavras, o estado interno do M1 é transformado em um novo estado no nível de abstração de M2.

Figura 9 – Estrutura da M2: E é o ambiente, M2 não interage com o ambiente, mas interage com M1; o ambiente age em M2 indiretamente, por meio de um processo evolucionário. Sym_1 é o símbolo elaborado por M2. LoA_2 é o nível de abstração no qual a M2 interage com E. (S_1, Sym_1) é a associação entre símbolo e o estado interno da M1, a saída da elaboração de M2.



Fonte: (FLORIDI, 2011)

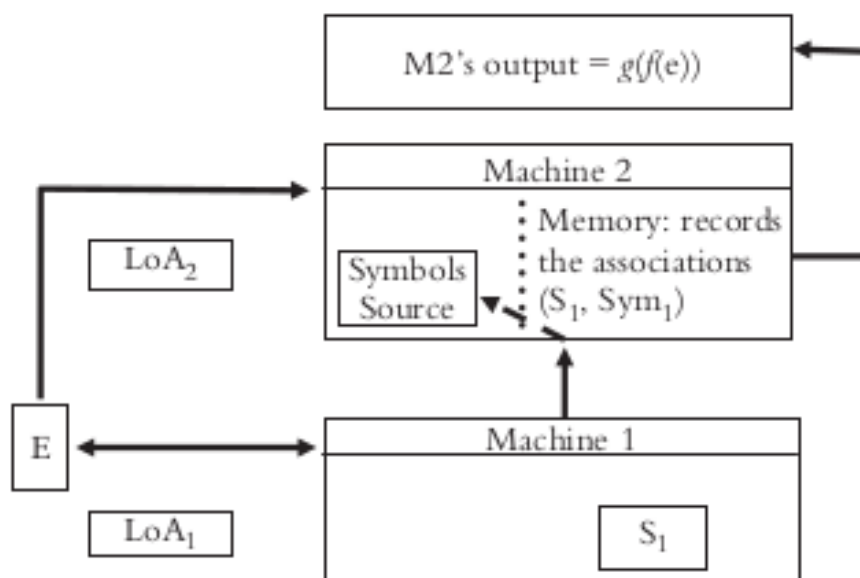
Uma vez obtido o novo estado, M2 associa o estado com um símbolo removido do conjunto de símbolos. Símbolos e estados são tipos diferentes de dados: eles estão associados - acoplados - mas não são transduzidos um no outro.

Quando um símbolo é escolhido, o M2 aplica uma regra de armazenamento e uma regra de execução. A regra de armazenamento registra o símbolo e o estado relacionado no espaço da memória. A regra de execução regula as comunicações entre M1 e M2 e diz respeito à associação entre um símbolo e um estado. Em seguida da regra de execução, cada vez que M2 recebe uma entrada de M1, ela inicialmente verifica se a entrada recebida, ou qualquer outra entrada similar, já foi elaborada. Se M2 não localizar uma entrada similar em sua memória, então continua o processo descrito acima. Caso contrário, não produz um novo símbolo, mas reproduz a associação já fundada em sua memória.

Seguindo a regra de execução, M2 obtém a mesma associação sempre que recebe o mesmo tipo de entrada de M1, associando assim, símbolos diferentes a estados internos diferentes de M1. Qualquer símbolo elaborado por M2 é relacionado através do estado

interno de M1 a um conjunto de ações, todas essas ações não distinguidas como diferentes pelo nível de abstração. Os símbolos de M2 estão agora embasados nas ações através dos estados internos correspondentes de M1.

Figura 10 – Arquitetura de um agente inteligente de duas máquinas. Um agente de duas máquinas recebe/envia alguma ação/percepção (e) do/para o ambiente E. E interage com a M1 e age na M2, modificando-a de acordo com o processo evolucionário. Qualquer ação é relacionado com o estado interno (S_1) da M1 em um nível de abstração específico, LoA_1 . M1 comunica seus estado internos para M2. O estado interno de M1 é transduzido para uma entrada de M2, que associa uma entrada a um símbolo (Sym_1). M2 guarda o estado e o símbolo relacionado em sua memória. Para qualquer entrada, M2 segue o procedimento definido pela regra de performance. Cada símbolo é selecionado pela M2 é uma função (g) do estado interno, S_1 . Já que S_n também é um resultado da função - $f(e)$ - a saída de M2 é uma função de uma função, $g(f(e))$.



Fonte: (FLORIDI, 2011)

A partir do fato de que o nível de abstração de M2 é mais geral do que o de M1, o M2 não tem uma percepção precisa dos estados internos do M1 e pode ser incapaz de distinguir entre estados internos semelhantes do M1. Para M2, o significado de tal símbolo é um significado geral, que surge de uma generalização de significados semelhantes.

Ao elaborar um significado geral, um agente de duas máquinas perde os significados específicos relacionados aos símbolos. Assim, parece que a evolução do processo prático geraria uma semântica composta de significados genéricos e sem significados específicos.

Floridi (2011) aponta o processo de comunicação entre agentes como uma forma para superar o problema de uma semântica empobrecida. A capacidade de compartilhar

semanticamente símbolos embasados para se comunicar entre os agentes garante que, através de pressão, o equilíbrio certo (adequado à sobrevivência) entre generalidade e especificidade da semântica em questão será alcançado.

Os agentes se envolvem em um tipo de jogo de linguagem adaptativa, como o jogo de adivinhação (STEELS, 2005). Durante o jogo, os agentes interagem e desenvolver um sistema comum de símbolos. No caso dos agentes de duas máquinas, os símbolos comunicados estão relacionados aos estados internos do falante e, indiretamente, a ação que ele realiza. Sempre que um símbolo é comunicado, o ouvinte executa uma ação. Como as ações são relevantes para a sobrevivência, os agentes que executam a ação apropriada têm uma chance maior de sobreviver e, portanto, de se reproduzir, do que os que não conseguem realizar a ação correta. Os agentes que sobrevivem recebem *feedback* positivo do ambiente e aprendem a associar esse símbolo recebido ao estado interno relacionado à ação que realizam. O ouvinte aplica a regra de armazenamento ao símbolo recebido. Ele realiza um novo processo de associação, e o agente armazena em sua memória o par resultante: o símbolo recebido e o estado interno do M1 relacionado a ação executada uma vez que o símbolo tenha sido ouvido. Na memória do M1 do ouvinte, os estados internos estão associados tanto ao símbolo que ele usou pela primeira vez para nomear esses estados quanto aos novos símbolos comunicados pelo falante. Desta forma, os símbolos comunicados também adquirem um significado para o ouvinte e podem ser usado pelos agentes para desenvolver um sistema de comunicação semanticamente embasado. Como o mesmo agente interage através de diferentes jogos de adivinhação com outros agentes, o mesmo símbolo pode se relacionar com os estados internos de diferentes agentes - com diferentes níveis de abstração - entre a população.

Assim, o símbolo comunicado pelo falante acaba nomeando um conjunto de estados semelhantes. Seguindo esta estratégia, um léxico compartilhado pode emergir através de comunicações entre uma população de agentes. Os símbolos compartilhados surgem de acordo com o uso, e pode-se concluir que os símbolos mais úteis e, portanto, recorrentes, serão usados como nomes de conjuntos de estados similares. A comunicação embasada se desenvolve quando os agentes já usam símbolos embasados e o problema de embasamento de símbolos é resolvido antes que os agentes comecem a se comunicar uns com os outros. Comunicação desempenha o papel de um processo de refinamento, não de embasamento.

Em Bielecka (2015) a abordagem de Floridi (2011) é tomada como uma das tentativas mais bem elaboradas de resolver problemas de aterramento de símbolos porém a abordagem é também criticada como não sendo suficiente para resolver o problema de embasamento de símbolos, em particular por causa da relação excessivamente liberal entre símbolos e estados internos de agentes, que é concebida em termos de níveis de abstração. Além disso, a noção de ação de Floridi também é vista como defeituosa por Bielecka.

Para Bringsjord (2015), Floridi (2011) mostra apenas que uma classe de robôs

pode, em teoria, conectar, em certo sentido, os símbolos que manipula com o mundo externo que percebe, e pode, com a força dessa conexão, comunicar-se de maneira subumana. Mas torna-se minúsculo e insignificante na face do vasto e imponente problema de embasamento de símbolos, que é o problema de construir um robô que compreenda genuinamente a informação semântica no nível da linguagem humana (natural), e que adquira para si o conhecimento que ganhamos ao ler tais língua. Para Bringsjord, esse problema permanece totalmente sem solução.

2.6.3 Não Representacionalista

A origem das abordagens não-representacionistas podem ser traçadas às críticas feitas em Brooks (1990), Brooks (1991) das concepções clássicas de representação. Brooks argumentou que o comportamento inteligente pode ser fruto das interações entre agentes incorporados e situados e o seu ambiente e que, para estes propósitos, representações simbólicas não são necessárias. Dentro desse paradigma argumenta-se que o problema de embasamento de símbolos é resolvido pois ele é completamente evitado: não existem representações simbólicas para serem embasadas.

2.6.3.1 Arquitetura de Subsunção

Brooks criticou o estado inerte em que a pesquisa de inteligência artificial se encontrava e acreditava que remover a representação complexa da inteligência artificial ajudaria a abordar áreas problemáticas na modelagem da mente. Sua crença era de que deveríamos desenvolver inteligência artificial sendo guiados pelo desenvolvimento evolucionário de nossa própria inteligência, e que essa abordagem espelhava o funcionamento de nossa própria inteligência.

De acordo com Brooks: “A representação é a unidade errada de abstração na construção das partes mais volumosas dos sistemas inteligentes” (BROOKS, 1991).

Em vez de modelar sistemas artificiais como sistemas de símbolos (fig. 11), Brooks rejeitou a abordagem simbólica para a modelagem da cognição e a necessidade de representações para este fim, apenas acoplamentos sensório-motores seriam necessários (fig. 12).

O problema do embasamento de símbolos é, para Brooks, um problema resolvido com base na robótica baseada em comportamento, baseando os robôs em um ambiente do mundo real e usando o ambiente circundante para guiar suas ações em vez de representações do ambiente. Brooks acredita que a questão do embasamento de símbolos de fato demonstra como sua abordagem baseada no comportamento aborda uma das fraquezas fundamentais da inteligência artificial tradicional e do conexionismo (BROOKS, 1990; BROOKS, 1991).

Figura 11 – Esquema de cognição simbólica.

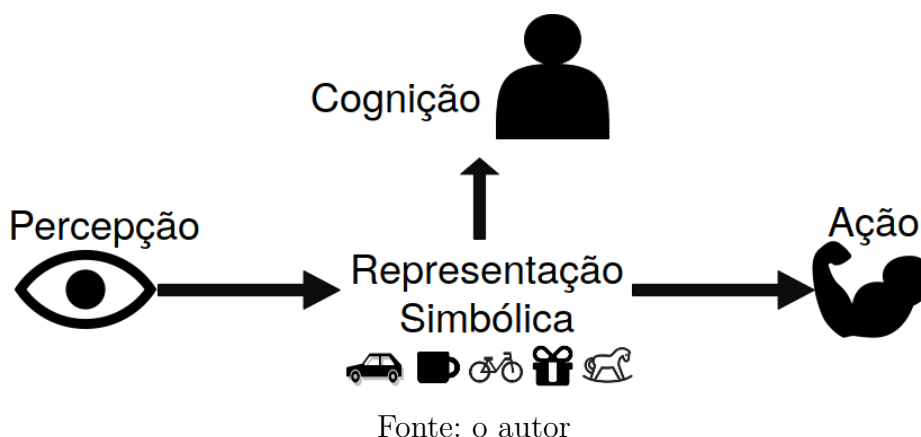
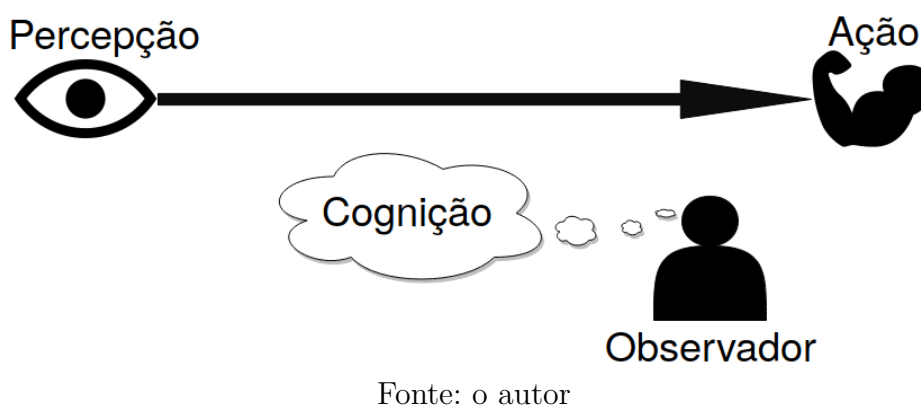


Figura 12 – Esquema de cognição segundo Brooks.



Outra observação importante de Brooks é que o próprio mundo é seu melhor modelo (BROOKS, 1990). Este está sempre atualizado. Ele sempre contém todos os detalhes que devem ser conhecidos. O importante é que os sensores devem captar os dados apropriadamente e com bastante frequência.

A contraproposta de Brooks (1991) é a “Hipótese do Embasamento Físico”, sustentando que os sistemas inteligentes deveriam estar embutidos no mundo real, sentindo e agindo nele, estabelecendo relações causais com percepções e ações. Não haveria necessidade de representações porque o sistema já está em contato com os objetos e eventos que precisaria representar.

Para se ter um sistema baseado na hipótese do embasamento físico, é necessário conectá-lo ao mundo através de um conjunto de sensores e atuadores. Entrada e saída digitadas não são de interesse. Eles não estão fisicamente embasados. Aceitar a hipótese de embasamento físico como base para a pesquisa envolve a construção de sistemas de maneira ascendente. Abstrações de alto nível precisam ser concretizadas. O sistema construído eventualmente tem que expressar todos os seus objetivos e desejos como ação física,

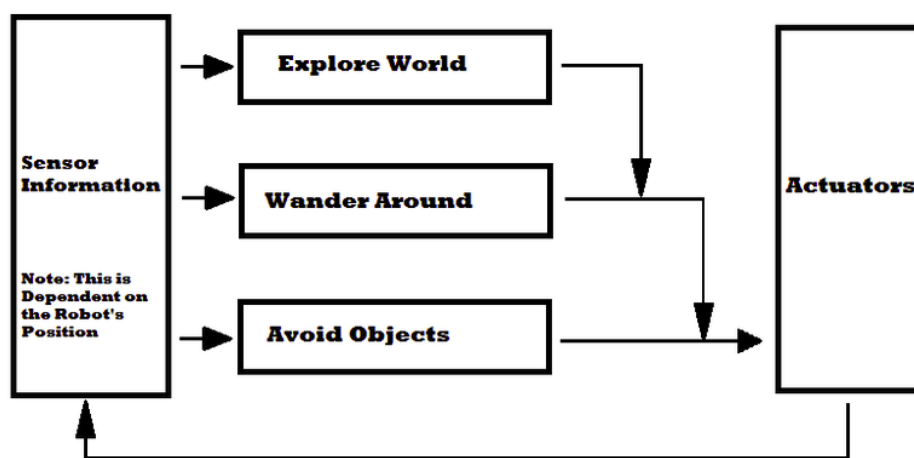
e deve extrair todo o seu conhecimento dos sensores físicos.

Para construir sistemas fisicamente embasados, Brooks (1986) desenvolveu uma arquitetura conhecida como arquitetura de subsunção, que permite conectar percepções a ações, incorporando corretamente robôs ao ambiente. A arquitetura de subsunção permite que agentes evitem qualquer elaboração de representações explícitas.

Em vez de guiar o comportamento pelas representações mentais simbólicas do mundo, a arquitetura da subsunção acopla a informação sensorial à seleção da ação. Isso é feito decompondo o comportamento completo em sub-comportamentos. Esses sub-comportamentos são organizados em uma hierarquia de camadas. Cada camada implementa um determinado nível de competência comportamental, e níveis mais altos são capazes de subsumir níveis mais baixos (combinar níveis mais baixos a um todo mais abrangente) a fim de criar um comportamento viável (BROOKS, 1986). Por exemplo, a camada mais baixa de um robô pode ser “evitar um objeto”. A segunda camada seria “passear”, que corre abaixo da terceira camada “explore o mundo”. Como um robô deve ter a capacidade de “evitar objetos” para “passear” efetivamente, a arquitetura de subsunção cria um sistema no qual as camadas superiores utilizam as competências de nível inferior. As camadas, que recebem todas as informações do sensor, trabalham em paralelo e geram saídas. Essas saídas podem ser comandos para atuadores ou sinais que suprimem ou inibem outras camadas.

Cada camada é composta por um conjunto de processadores que são máquinas de estado finito aumentadas (AFSM), o aumento sendo a adição de variáveis de instância para manter estruturas de dados programáveis. Uma camada é um módulo e é responsável por um único objetivo comportamental, como “passear”. Não há controle central dentro ou entre esses módulos comportamentais. Todos os AFSMs recebem de forma contínua e assíncrona a entrada dos sensores relevantes e enviam a saída para os atuadores (ou outros AFSMs). Sinais de entrada que não são lidos no momento em que um novo é entregue acabam sendo descartados. Esses sinais descartados são comuns e são úteis para o desempenho, pois permitem que o sistema trabalhe em tempo real, lidando com as informações mais imediatas (BROOKS, 1986).

Figura 13 – Exemplo de arquitetura de subsunção proposta por Brooks.



Fonte: (BROOKS, 1986)

Como não há controle central, os AFSMs se comunicam entre si por meio de sinais de inibição e supressão. Os sinais de inibição bloqueiam os sinais de alcançar atuadores ou AFSMs, e os sinais de supressão bloqueiam ou substituem as entradas em camadas ou seus AFSMs. Este sistema de comunicação AFSM é como camadas mais altas subsumem as mais baixas, assim como a arquitetura lida com a arbitragem de seleção de ações e prioridades em geral.

Vendo as três camadas de uma arquitetura de subsunção com mais detalhes (fig. 14), podemos analisar como o modelo proposto por Brooks funciona:

- (i) A camada de nível mais baixo implementa um comportamento que faz com que o robô evite atingir objetos. Evita tanto objetos estáticos quanto objetos em movimento. O sonar de máquina de estado finito rotulado simplesmente executa os dispositivos de sonar e cada segundo emite um mapa instantâneo com as leituras convertidas em coordenadas polares. Este mapa é passado para o colidir e sentir a máquina de estados finitos. O primeiro deles simplesmente observa se há algo à frente e, se for o caso, envia uma mensagem de parada para a máquina de estados finitos encarregada de conduzir o robô para frente. Simultaneamente, a outra máquina de estados finitos calcula uma força repulsiva no robô, onde cada retorno do sonar é considerado para indicar a presença de um objeto repulsivo. A saída é passada para a máquina *runaway* que a limita e a passa para a máquina de giro, que orienta o robô diretamente para longe da força repulsiva somada. Finalmente, a máquina avançada leva o robô para frente. Sempre que esta máquina recebe uma mensagem de parada enquanto o robô está avançando, ela comanda o robô a parar. Essa rede de máquinas de estados finitas gera comportamentos que permitem ao

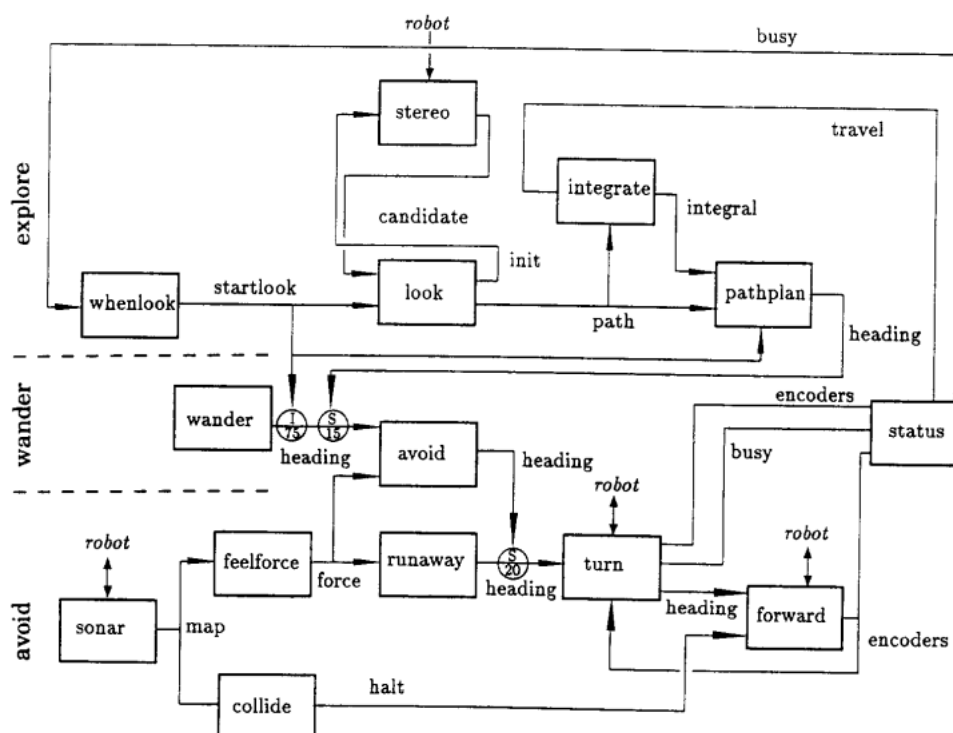
robô evitar objetos. Se começar no meio de uma sala vazia, simplesmente fica lá. Se alguém chega até ele, o robô se afasta. Se se mover na direção de outros obstáculos, ele para. No geral, ele consegue existir em um ambiente dinâmico sem bater ou ser atingido por objetos.

- (ii) A próxima camada faz o robô vagar, quando não está ocupado, evitando objetos. A máquina de estado finito de “Vagar” gera um rumo aleatório para o robô a cada dez segundos ou mais. A máquina de evitar trata esse título como uma força atrativa e a soma com a força repulsiva calculada pelos sonares.

Ele usa o resultado para suprimir o comportamento de nível mais baixo, forçando o robô a se mover em uma direção próxima ao que o “vagar” decidiu, mas ao mesmo tempo evita obstáculos. Observe que, se as máquinas de estado finito de virada e avanço estiverem ocupadas executando o robô, o novo impulso de “vagar” será ignorado.

- (iii) A terceira camada faz o robô tentar explorar. Ele procura por lugares distante, em seguida, tenta alcançá-los. Essa camada suprime a camada de “vagar” e observa como a camada inferior desvia o robô devido a obstáculos. Corrige todas as divergências e o robô atinge o objetivo.

Figura 14 – Na arquitetura de subsunção, as máquinas de estados finitos são ligadas em camadas de controle. Cada camada é construída em cima de outras camadas já existentes.



Fonte: (BROOKS, 1986)

A ideia principal dos níveis de competência é que pode se construir camadas de um sistema de controle correspondente a cada nível de competência e simplesmente adicionar uma nova camada a um conjunto existente para passar para o próximo nível mais alto de competência geral.

É construído um sistema completo de controle de robô que atinge a competência de nível 0 e então é depurado completamente. Em seguida, é construída uma outra camada de controle, que é chamada de sistema de controle de primeiro nível. Ele é capaz de examinar dados do sistema de nível 0 e também é permitido injetar dados nas interfaces internas do nível 0 suprimindo o fluxo de dados normal. Esta camada, com o auxílio do nível zero, atinge o nível 1 de competência. A camada zero continua a correr inconsciente da camada acima, o que às vezes interfere em seus caminhos de dados. O mesmo processo é repetido para atingir níveis mais altos de competência.

Para Floridi (2011), na verdade, o problema do embasamento de símbolos é apenas postergado e não completamente evitado. Um agente implementando uma arquitetura de subsunção pode inicialmente não precisar lidar com o problema de embasamento de símbolos para lidar com seu ambiente. Porém, se for para desenvolver uma proto-linguagem

elementar e alguma capacidade cognitiva mais elevada, será necessária a capacidade de manipular símbolos, então o problema de embasamento de símbolos se tornará presente novamente.

2.6.3.2 Modelo baseado em comportamento

Varshavskaya (2002) defende que o desenvolvimento de capacidades semânticas em um agente deve ser modelado com base no desenvolvimento das capacidades linguísticas de crianças. Teorias da aquisição linguagem parecem mostrar que crianças adquirem habilidades linguísticas por meio da utilização da linguagem como uma ferramenta com a qual interagem com seu ambiente e com outros agentes, para satisfazer suas necessidades e atingir seus objetivos.

Com isso, apoia uma interpretação pragmática da aquisição de linguagens em agentes:

“A linguagem não é vista como um sistema simbólico denotacional para referência a objetos e relacionamentos entre eles, tanto quanto uma ferramenta para comunicar intenções. O enunciado é uma maneira de manipular o ambiente através das crenças e ações dos outros.” (VARSHAVSKAYA, 2002)

Linguagem se torna apenas outra forma de interação pragmática entre um agente e seu ambiente e, como tal, sua semântica não necessita de representações.

A hipótese de uma linguagem sem representações foi corroborada por experimentos envolvendo uma robô conhecido como KISMET (BREAZEAL, 2000; BREAZEAL, 2002)

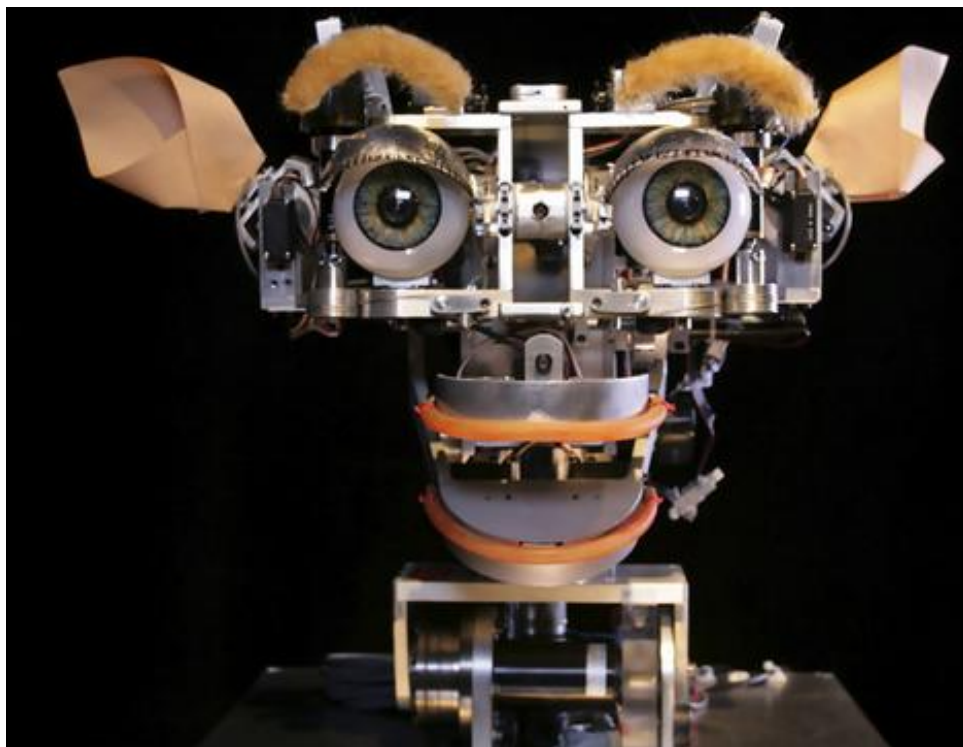
“KISMET é uma cabeça robótica expressiva, projetada para ter uma aparência jovem e perceptual e capacidades motoras sintonizadas nos canais de comunicação humanos. O robô recebe entrada visual de quatro câmeras CCD coloridas e entrada auditiva de um microfone. Executa atos motores como vocalizações, expressões faciais, mudanças de postura, bem como direção do olhar e cabeça orientação.” (VARSHAVSKAYA, 2002)

Os experimentos mostram que o KISMET pode aprender de seus treinadores como usar símbolos e desenvolver comportamentos proto-linguísticos.

Aprender a se comunicar com o professor usando uma semântica compartilhada faz parte de uma tarefa mais geral de como aprender a interagir e manipular seu ambiente. O KISMET possui um sistema motivacional e comportamental, um conjunto de comportamentos vocais, impulsos regulados e um algoritmo de aprendizagem, que juntos constituem um módulo de proto-linguagem (BREAZEAL, 2002).

Por proto-linguagem entende-se ao momento pré-gramatical do desenvolvimento de uma linguagem que permite o desenvolvimento de articulação de sons nos primeiros

Figura 15 – Imagem do robô KISMET.



Fonte: (BREAZEL, 2002)

meses de vida. Para o KISMET, a proto linguagem fornece os meios para embasar o desenvolvimento de suas capacidades linguísticas.

O KISMET é um agente autônomo, com seus próprios objetivos e estratégias, que faz com que ele implemente comportamentos específicos para satisfazer necessidades. Suas motivações fazem ele executar suas tarefas. Essas motivações são fornecidas por um conjunto de variáveis homeostáticas, chamadas de impulsos, como o nível de engajamento com o ambiente ou a intensidade social. O KISMET possui emoções também, que são tipos de motivações (BREAZEL, 2002).

As emoções do KISMET dependem das avaliações dos seus estímulos perceptivos. Quando seus valores homeostáticos estão fora de equilíbrio, o KISMET pode realizar uma série de ações que permitem ele atingir o equilíbrio. Nesses casos, o KISMET usa alguns comportamentos proto verbais – ele expressa suas emoções – com os quais ele age sobre si mesmo e sobre seu ambiente para restaurar o equilíbrio dos valores originais (BREAZEL, 2002).

O KISMET consegue implementar comportamentos proto-linguísticos graças a presença de dois impulsos (um para linguagem e outro para exploração do ambiente), uma arquitetura para expressar comportamentos proto verbais e uma arquitetura para o aparato visual.

Tabela 3 – Correspondência entre comportamento não-verbal e funções proto-linguísticas

Emoção	Comportamento	Função proto-linguística
Raiva, Frustração	Reclamar	Regulatório
Nojo	Retirar	Instrumental ou regulatório
Medo	Escapar	————
Calmo	Engajar	Interacional
Alegria	Demonstrar prazer	Pessoal ou interacional
Tristeza	Demonstrar tristeza	Regulatório ou pessoal
Surpresa	Resposta de sobressalto	————
Tédio	Procurar	————

Fonte: (FLORIDI, 2011)

Para entender como os processos de emulação influenciam o aprendizado do KISMET, suponha que o KISMET aprenda a palavra “verde”. O treinador mostra ao KISMET um objeto verde e, ao mesmo tempo, pronuncia a palavra “verde”, enquanto o KISMET está observando o objeto verde. Então o treinador esconde o objeto verde, que será mostrado novamente somente se o KISMET procurar por ele e expressar um pedido correspondente à palavra “verde”. Se o KISMET pronunciar a palavra “verde” para solicitar o objeto verde, então o KISMET aprendeu a associação entre a palavra e o objeto, e usar a palavra de acordo com o seu significado. Ao executar tarefas semelhante, o KISMET parece ser capaz de adquirir capacidades semânticas e desenvolvê-las sem representações elaboradas. A questão é se isso é suficiente para resolver o problema do embasamento simbólico.

Para Floridi (2011) a abordagem adotada por Varshavskaya (2002) é inadequada para servir como solução para o problema do embasamento simbólico. Segundo ele, a questão da origem das capacidades semânticas em sistemas artificiais não pode ser abordada com referência à modalidades apropriadas à agentes humanos, pois nesse caso se assume:

- Uma predisposição natural e inata à aquisição de linguagem;
- A existência de uma linguagem já bem estabelecida;
- A presença de uma comunidade de falantes, proficientes na linguagem, que podem transmitir o conhecimento da linguagem para novos membros.

Nenhuma dessas suposições são justificadas no caso de um agente artificial. Floridi (2011) estressa que para se resolver o problema do embasamento simbólico as capacidades

semânticas de um agente de maneira autônoma pelo próprio agente: nenhuma inatismo ou externalismo é permitido.

No entanto, ambos ocorrem no KISMET. O KISMET é (inatamente) dotado de características semânticas (presença de uma proto-linguagem) e (externamente) realiza um aprendizado explicitamente emulativo. Ele associa o símbolo “verde” ao objeto verde mostrado pelo treinador, mas a relação semântica inicial entre “verde” e o objeto verde é pré-estabelecido e fornecido pelo próprio treinador.

O ponto pode ser esclarecido considerando a seguinte dificuldade: o símbolo “verde” para KISMET refere-se ao objeto verde específico mostrado ao KISMET pelo treinador ou, em vez disso, nomeie um recurso geral - a cor do objeto verde - que o KISMET pode reconhecer. Suponha que mostremos ao KISMET vários objetos, com formas diferentes, mas todos tendo a propriedade de serem verdes. Entre esses objetos, há também o objeto verde que o KISMET já conhece. Se alguém pedir ao KISMET para reconhecer um objeto verde, ele reconhecerá apenas o objeto verde que ele tenha visto antes. Isso ocorre porque o KISMET não nomeia classes de objetos, por exemplo, todos objetos verdes. Em vez disso, tem símbolos que nomeiam seus referentes rigidamente, como se fossem seus nomes próprios. Para o KISMET, o objeto verde não será verde, será chamado de 'verde'. Isto segue das elaborações não-representacionais do KISMET.

A semântica do KISMET pode crescer tanto quanto o processo de aprendizado emulativo externamente sobreposto pelo instrutor permite, mas a ausência de representações significa que o KISMET não desenvolverá qualquer estrutura categórica no sentido requerido para resolver o problema de embasamento simbólico.

2.7 CONSIDERAÇÕES FINAIS

Este capítulo fundamenta o conceito de Inteligência Artificial, destacando dois dos grandes paradigmas da área. Mostrou também os principais conceitos que definem um agente, detalhando algumas de suas características essenciais. Foi apresentado o conceito de sistemas multi-contexto, que será fundamental no âmbito da implementação do trabalho.

Um dos papéis mais importantes deste capítulo é a fundamentação do problema do embasamento de símbolos, trazendo esse problema interdisciplinar para o contexto da Ciência da Computação. Foram discutidas algumas das soluções propostas para esse problema, dessa forma podemos aprender certas lições com tais propostas e utilizar seus conceitos como base para o desenvolvimento de uma solução adequada ao contexto deste trabalho.

3 TRABALHOS RELACIONADOS

Neste capítulo, são apresentados os principais trabalhos, que de alguma forma, se relacionam à proposta apresentada nesta pesquisa. São mostrados trabalhos com abordagens de agentes como sistema multi-contexto e trabalhos que apresentam a definição de um *framework* para o desenvolvimento de agentes multi-contexto.

3.1 SIGON: UMA LINGUAGEM PARA AGENTES COMO SISTEMAS MULTI-CONTEXTO

Em [Gelaim et al. \(2018\)](#) é apresentada a linguagem Sigon. Construída com o propósito de operacionalizar agentes como sistemas multi-contexto. Com base nisso, Sigon permite a flexibilização na criação e relacionamento entre componentes do agente.

Seja Ag um agente Sigon. Ag é definido da seguinte maneira:

$$Ag = \langle CC \cup \bigcup_{i=0}^n C_i, \Delta_{rp} \rangle$$

Sendo que, CC é o contexto de comunicação. C_i é um contexto do agente, composto por uma linguagem, um conjunto de axiomas e um conjunto de regras de inferência. Uma regra de ponte $rp_i \in \Delta_{rp}$ é uma regra conectando dois ou mais contextos.

As regras de ponte têm o papel de trocar conhecimento entre contextos e descrever o comportamento de execução do agente.

A modularidade é uma característica fundamental no desenvolvimento de arquiteturas de agentes multi-contexto. Vários módulos podem ser definidos para expressar propriedades específicas, como crenças, desejos e intenções. O fator chave para o relacionamento entre os componentes do agente é a definição correta das regras de ponte, que descrevem a relação entre os contextos.

A cabeça representa o contexto que irá adicionar o conhecimento X se a condição no corpo da regra for satisfeita. Cada regra de ponte pode adicionar ou remover apenas um conhecimento por operação. Uma regra de ponte começa com o símbolo ‘!’, e a separação entre a cabeça e o corpo é dada pelo símbolo ‘:-’.


```

bridgeRule: head ':'- body';
head: '!' negation? contextName (clause | negation? variable);
body: negation? contextName ((clause | negation? variable) | plan)
((AND | OR) negation? contextName (
(clause | negation? variable) | plan))*;

```

Os contextos do agente são de dois tipos: funcionais e lógicos. Um contexto lógico é usado para representar atitudes mentais, como crenças, desejos e intenções. Os contextos funcionais são o de planejamento e comunicação. Apenas os contextos funcionais de comunicação e planejamento são definidos. Já os contextos lógicos podem ser criados conforme a necessidade da aplicação. As regras representam como esses contextos são definidos na linguagem:

```

context: logicalContext | functionalContext;
logicalContext: logicalContextName ':' formulas;
functionalContext: communicationContext | plannerContext;
communicationContext: COMMUNICATION ':' (sensor | actuator)+;
plannerContext: PLANNER ':' plansFormulas;
logicalContextName: primitiveContextName | customContextName;
primitiveContextName: BELIEFS | DESIRES | INTENTIONS;
customContextName: '_' (LCLETTER| UCLETTER)+ character*;

```

Um agente deve ter pelo menos o contexto de comunicação. Este é necessário, uma vez que assumimos que o agente deve interagir com seu ambiente. O contexto de comunicação é composto por sensores e atuadores. Um sensor S_i é um par ordenado:

$$S_i = (\omega, \chi),$$

Onde ω é o identificador do sensor, e χ é uma função que mapeia uma observação para uma percepção. A forma como a função é implementada é dependente do domínio, podendo ser capturada por um sensor físico, por algum algoritmo de processamento de imagens, áudio, etc.

Um atuador A_j também é formalizado como um par ordenado:

$$A_j = (\rho, \alpha),$$

Onde ρ é o identificador, e α é a ação a ser executada. Sensores e atuadores são representados no contexto de comunicação pelos predicados especiais “sensor” e “actuator”, compostos por um identificador e por uma chamada de função. Enquanto que o

identificador permite que o agente raciocine sobre o sensor/atuador, a função permite a captura/atuação.

sensor: *SENSOR* ‘(‘*sensorIdentifier*’, ‘*sensorImplementation*’)’ ‘.’’;
actuator: *ACTUATOR* ‘(‘*actuatorIdentifier*’, ‘*actuatorImplementation*’)’ ‘.’’;

Posto isso, seja CC o contexto de comunicação, este é definido como:

$$CC = \left\langle \bigcup_{i=1}^n S_i \cup \bigcup_{j=1}^m A_j \right\rangle,$$

Onde S_i com $1 \leq i \leq n$ são os sensores do agente, e A_j com $1 \leq j \leq m$ os seus atuadores.

Além do contexto de comunicação, a linguagem permite a criação de um outro contexto funcional específico para construir planos (*Planner*). Na literatura, o contexto de planos é composto por dois componentes principais: ações e planos, e opcionalmente tem pré-condições, pós-condições e custo. Um plano obrigatoriamente realiza alguma proposição e tem ações.

plansFormulas: ((*plan* | *action*) ‘.’)* ;
plan: *PLAN* ‘(‘*somethingToBeTrue*’, ‘*compoundAction*
‘,’ *planPreconditions*’, ‘*operator*? *planPostconditions*)?’
‘,’ *cost*)?’ ‘.’’;
action: *ACTION* ‘(‘*functionInvocation*’, ‘
(*actionPreconditions*’, ‘*operator*? *actionPostconditions*)?’
‘,’ *cost*)?’ ‘.’’;

Uma ação é definida como:

$$action(\alpha, Pre, Pos, c_a)$$

Onde α é uma ação, Pre são as pré-condições necessárias para a execução de α , Post são as pós-condições da execução da ação e c_a é o seu custo. Um plano é definido como:

$$plan(\varphi, \alpha, Pre, Pos, c_a)$$

Onde φ é uma pós-condição e representa o que o agente espera que se torne realidade após a execução do plano, α é a ação, ou conjunto de ações que serão executadas no plano, Pre são pré-condições, Pos representa o que o agente acredita que terá como consequência, e c_a é o custo da execução do plano. Seja CP o contexto planejador. CP é definido como:

$$CP = \left\langle \bigcup_{i=1}^n Ac_i \cup \bigcup_{j=1}^m P_j \right\rangle,$$

Onde Ac_i com $1 \leq i \leq n$ são as ações do agente, e P_j com $1 \leq j \leq m$ os seus planos.

Um contexto lógico é composto por um conjunto de fórmulas proposicionais. Além disso, é possível a vinculação de ações e graus de certeza a cada fórmula.

```

formulas: (propFormula | folFormula)*;
propFormula: (propClause (':-' propLogExpr)?) ':';
folFormula: (folClause (':-' folLogExpr)?) ':';
propClause: negation? constant annotation?;
folClause: negation? constant ('term (',' term)* ') annotation?;
annotation: (preAction gradedValue?) | gradedValue;

```

Seja C um contexto. C é definido como:

$$C = \langle L, A_x, \delta \rangle,$$

Onde L é a linguagem do contexto, A_x os seus axiomas, e δ as regras de inferência. Eles podem ser construídos considerando lógica proposicional, de primeira ordem, dinâmica ou probabilística.

A troca de conhecimento entre contextos é feita de forma declarativa por meio de regras de ponte. A estrutura de regras de ponte é composta pelo contexto que irá adicionar uma crença (cabeça) com base em regras de outros contextos (corpo). Por exemplo:

$$\frac{C_1 : \varphi, C_2 : \alpha}{C_3 : \varphi}$$

Representa que quando nos contextos C_1 e C_2 do corpo, for deduzido, respectivamente, φ e α , no contexto da cabeça C_3 , será adicionado φ .

3.2 IMPLEMENTAÇÃO DE UM FRAMEWORK PARA O DESENVOLVIMENTO DE AGENTES COMO SISTEMA MULTI-CONTEXTO

O trabalho de [Arnhold \(2018\)](#) expande o trabalho apresentado por [Gelaim et al. \(2018\)](#), com o objetivo de apresentar a criação de um *framework* para permitir o desenvolvimento e criação de agentes BDI vistos como um sistema multi-contexto.

A problematização deste trabalho surge de um contexto mais amplo, que aponta para certas limitações de algumas abordagens para modelagem e implementação de agentes, a saber, a dificuldade que sistemas de agentes apresentam para integrar diferentes representações de conhecimento.

Um agente, de modo geral, é bastante restrito, isso porque as formas para representação de seus estados mentais é limitada para conjuntos de símbolos proposicionais ou suas negações, que como consequência acaba limitando o seu uso para determinados contextos.

O uso de um sistema multi-contexto, como já apresentado, permite representar e isolar lógicas complexas em um determinado contexto, usando regras e ponte para integrar conhecimento entre diferentes contextos.

Arnhold (2018) resgata os principais conceitos que definem um agente, detalhando suas propriedades e sua relação com o ambiente por meio de sensores e atuadores. O trabalho também deu um foco especial para a arquitetura BDI como modelo de desenvolvimento de agentes que é utilizado como base para o *framework* implementado pela pesquisa.

O trabalho de Gelaim et al. (2018) apresentou um modelo para definição de agentes como um sistema multi-contexto, ele provê a teoria, a arquitetura e a linguagem, elementos necessários para a implementação do *framework*. O trabalho de Arnhold (2018) utiliza este modelo como base e núcleo para implementação de um *framework* que permite o desenvolvimento de agentes BDI sobre um sistema multi-contexto.

Arnhold (2018) analisa o trabalho de Bordini e Hübner (2006), que tem como objetivo mostrar as principais funções disponíveis no *Jason*, que é um *framework* para o desenvolvimento de agentes BDI baseado em *AgentSpeak* - uma linguagem de programação orientada a agentes baseada em programação lógica.

De forma geral, o trabalho de Bordini e Hübner (2006) apresenta uma solução completa para o desenvolvimento de agentes BDI. A arquitetura do *AgentSpeak* é inspirada em BDI, e em sua versão original é considerada uma linguagem abstrata. Esta característica se assemelha ao próprio trabalho de Arnhold (2018), considerando que os modelos de agente como sistema multi-contexto são restritos a teorias e arquiteturas abstratas. Frente a isso, Bordini e Hübner (2006) adicionam ao *AgentSpeak* extensões que são necessárias para a execução e operacionalização de agentes. *Jason* é implementado em *Java*, assim como o *framework* proposto por Arnhold (2018).

No modelo é definido um ciclo de raciocínio e a integração entre bases de conhecimento. Além disso são fornecidos diferentes artefatos que buscam facilitar o desenvolvimento de agentes, como por exemplo, a criação de ambientes e uma *IDE* para o desenvolvimento.

Comparando o Jason ao *framework* proposto por Arnhold (2018), é importante enfatizar que ambos são implementados utilizando como base a *Java Virtual Machine*. Por outro lado, Jason - por ser uma linguagem já consolidada - fornece uma solução completa para o desenvolvimento de agentes, como ferramentas de *debug*, *plugins* para *IDEs*, manuais, entre outros. Arnhold (2018), por outro lado, só implementa uma linguagem. Além disso, a plataforma Jason oferece suporte para sistemas multi-agentes, diferente do trabalho de Arnhold (2018), que foi projetado para a atuação de um único agente.

Devido a estruturação do Jason a customização de agentes é mais custosa. Por exemplo, na versão original do Jason, sensores e atuadores ficam acoplados ao ambiente. Esta característica facilita o desenvolvimento de agentes, mas também dificulta a integração com diferentes tipos de ambientes. Em Arnhold (2018), sensores e atuadores estão ligados ao agente, não sendo definida uma interface para ambientes, o que facilita a integração do agente com outras entidades.

Arnhold (2018) também apresenta o trabalho de Dastani et al. (2003) apresenta a especificação para uma linguagem de agentes cognitivos. A linguagem é baseada na *3APL* (*An Abstract Agent Programming Language*) e permite ao programador a definição de estados mentais como crenças, objetivos, planos e ações.

Comparado ao próprio trabalho de Arnhold (2018), a definição dos planos mostrada em Dastani et al. (2003) é semelhante, considerando que os mesmos são derivados em ações nos dois modelos. Mas, por outro lado, em *3APL* um plano executado altera diretamente a base de crenças. No *framework* implementado por Arnhold (2018), que segue o modelo proposto em Gelaim et al. (2018), essa alteração é feita diretamente no ambiente, devendo ser observada pelo agente para gerar um novo ciclo de deliberação. Um outro fator que é importante comparar entre os dois trabalhos são definições e configurações necessárias para a execução de um agente. Em *3APL* é preciso definir um conjunto de regras, além dos planos e metas. No modelo apresentado por Gelaim, a definição de regras não é necessária, considerando que, por padrão, é definido um conjunto de regras baseadas em BDI.

O trabalho de Bordini e Hübner (2006) mostra uma solução completa para o desenvolvimento de agentes, desde a especificação até a implementação, parte da arquitetura de software definida em Bordini e Hübner (2006) foi utilizada como base para a implementação proposta por Arnhold (2018). Por outro lado, a integração de um agente JASON com diferentes tipos de ambientes é dificultada devido sua arquitetura. Frente a isto, na pesquisa de Arnhold (2018), foi desenvolvido um modelo de sensores e atuadores desacoplados da representação do ambiente o que acaba facilitando o uso do agente em diferentes contextos. O trabalho de Dastani et al. (2003) mostra a especificação de uma linguagem de agentes.

Assim como o *framework* implementado em Arnhold (2018), o modelo é genérico

e permite criar novas configurações para a execução de um agente. Por outro lado, os detalhes técnicos que, de fato, permitem a execução de um agente são limitados. Esta característica acaba dificultando a implementação da arquitetura e do modelo proposto por [Dastani et al. \(2003\)](#). Entre os três trabalhos apresentados, a proposta de [Bordini e Hübner \(2006\)](#) é mais próxima ao *framework* implementado por [Arnhold \(2018\)](#), por tratar de questões relacionadas a implementação de um agente. Além disso, também é desenvolvida em *JAVA* e implementa uma *parser* com base em uma linguagem de agente. Relacionado ao modelo de agentes utilizado neste trabalho o trabalho de [Gelaim \(2016\)](#) se assemelha por também definir um agente visto como um sistema multi-contexto.

Considerando que o trabalho apresentado em [Gelaim et al. \(2018\)](#) se trata de uma arquitetura genérica, que permite a definição de novos contextos e regras de ponte, é possível representar através dela diferentes tipos de teorias. Com o objetivo de corporificar o modelo genérico, [Arnhold \(2018\)](#) utiliza a teoria do raciocínio prático, por meio de uma arquitetura BDI.

Com o objetivo de permitir a operacionalização da linguagem e a execução do agente, nessa seção é apresentada uma proposta de arcabouço de *software* denominada como *framework*. Sua implementação, realizada em *JAVA*, é a principal contribuição do trabalho de [Arnhold \(2018\)](#).

Para simplificar o desenvolvimento e desacoplar os componentes, a estrutura a ser implementada neste trabalho segue a especificação de ([GELAIM et al., 2018](#)), e é dividida em dois componentes principais: *Parser* e *Agent*.

O componente *Parser* é responsável pela transformação dos arquivos fontes do agente em uma estrutura de dados executável. A geração do *Parser* é feita pela ferramenta *Antlr*.

A entrada é feita por um único arquivo em formato “.on” contendo a definição do agente.

Similar a definição formal de um agente como sistema multi-contexto, a representação do agente como um *software* é feita por decomposição modular usando contextos e regras de pontes. Um contexto, é uma unidade funcional de *software* e uma regra de ponte é utilizada para a troca de informação entre as unidades. Para representar um agente, é definido no *framework* a classe *Agent* que contém referências para instâncias de contextos e regras de ponte. Após a execução do *parser* é criada uma instância de agente e sua execução é inicializada.

Um contexto é generalizado no *framework* por meio da interface *ContextService* que estabelece métodos para: adição, atualização e busca de teoria, verificação de uma teoria e nome do contexto. Sob a perspectiva da engenharia de *software* um contexto representa um componente que encapsula regras internas e por meio de seus métodos

regras de ponte integram conhecimento entre diferentes unidades. A teoria, representada pela classe *Theory*, é implementada na biblioteca *tuProlog2* que fornece um ambiente *prolog*, sobre o qual é executado o raciocínio utilizando a *Java-Virtual-Machine*.

Com o objetivo de integrar conhecimento entre os contextos, é utilizada uma estrutura de dados para representar as regras de ponte, chamado *BridgeRule*, que possui um *Body* e *Head*.

A comunicação do agente com seu ambiente ou com outros agentes é feita pelo contexto de comunicação por meio de sensores e atuadores, não acoplados a uma estrutura de ambiente.

Sob o ponto de vista arquitetura, um sensor contém um publicador de literais, seguindo o paradigma da programação reativa. Considerando que o agente pode ter vários sensores que podem capturar percepções de forma paralela, um *Sensor* é tratado como uma *Thread* implementando a interface *Runnable*.

Internamente, o publicador será assinado pelo contexto de comunicação. Ao receber uma percepção ela é persistida no contexto de comunicação e o conjunto de regras de ponte é executado, sintetizando um ciclo de raciocínio.

Para alterar o ambiente ou enviar mensagens para outras entidades, o agente utiliza seus atuadores. Um atuador é definido pelo *framework* como uma classe abstrata. Considerando que cada ciclo é executada uma única ação diferente de um sensor um atuador não é uma *Thread*.

Por meio do *framework* desenvolvido é possível implementar um agente BDI, fazendo uso de contextos, regras de ponte, sensores e atuadores. Paralelamente, o *framework* também permite a execução de diferentes tipos de raciocínio sem um grande custo de implementação, conforme foi mostrado nos experimentos com regras de ponte e contextos feitos por [Arnhold \(2018\)](#).

3.3 CONSIDERAÇÕES FINAIS

O trabalho de [Gelaim et al. \(2018\)](#) apresenta a linguagem Sigon para operacionalizar agentes como Sistemas Multi-Contexto, permitindo a flexibilização na criação e relacionamento entre componentes do agente, porém não especifica como deve ser sua implementação. O trabalho de [Arnhold \(2018\)](#), por outro lado, expande o trabalho apresentado por [Gelaim et al. \(2018\)](#), apresentando um *framework* para permitir o desenvolvimento e criação de agentes BDI vistos como um sistema multi-contexto.

Esses trabalhos servirão como base para o desenvolvimento do agente utilizado neste trabalho, utilizando a linguagem Sigon, apresentada por [Gelaim et al. \(2018\)](#), para projetar um agente como sistema multi-contexto e utilizando o *framework*, proposto por

[Arnhold \(2018\)](#), para implementar o agente.

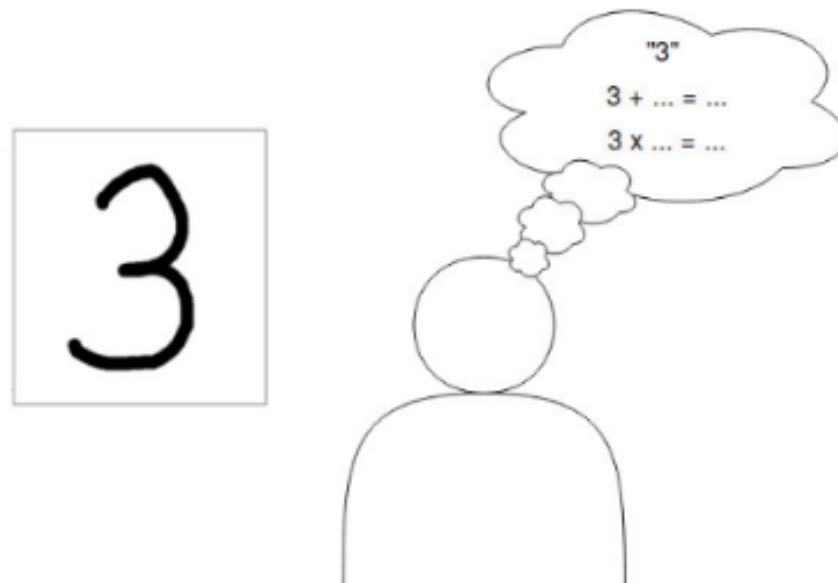
4 DESENVOLVIMENTO

4.1 PROPOSTA

Como foi explicado em 2.5, o problema do embasamento simbólico é o problema de como símbolos ganham seus significados. O embasamento é uma função de entrada/saída. As entradas sensoriais de objetos externos são conectadas à símbolos internos de uma mente por meio desse embasamento.

Em termos práticos para o campo da inteligência artificial, esse problema se manifesta na incapacidade de um agente computacional simular a capacidade cognitiva humana de associar um estímulo sensorial visual ao representante simbólico interno desta mesma entrada. Então, por exemplo, se uma pessoa receber como estímulo visual uma imagem de um número escrito em um pedaço de papel, as capacidades cognitivas humanas permitem que a pessoa entenda que aquilo que ela percebe como estímulo visual representa algo equivalente a um símbolo presente internamente na mente dela, que pode ser manipulado no contexto do sistema simbólico a qual pertence.

Figura 16 – Processo de reconhecimento de estímulo sensorial visual em um ser humano.

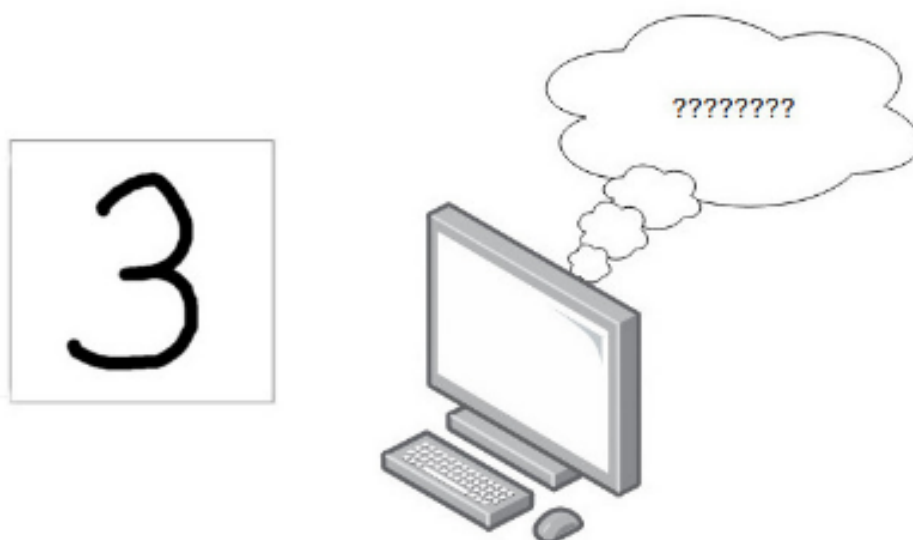


Fonte: o autor

No caso de um agente computacional dotado da capacidade sensorial visual, por outro lado, ao receber o estímulo visual do número, ele não consegue associar esse dado visual a algum símbolo que o representa. Portanto, mesmo que o sistema simbólico do agente computacional contenha um símbolo que seja equivalente ao dado sensorial que lhe

está sendo apresentado, o agente não consegue fazer a conexão de que esse dado sensorial de entrada é equivalente ao símbolo contido internamente no agente.

Figura 17 – Processo de reconhecimento de estímulo sensorial visual em um agente artificial.



Fonte: o autor

Podemos compreender o que queremos alcançar usando como exemplo um conjunto de imagens que representam dados perceptivos que o agente irá identificar:

Figura 18 – Coleção de imagens que compõem o conjunto C de dados não-simbólicos.



Fonte: o autor

Então teremos um conjunto que representa as percepções sensoriais de um agente, portanto um conjunto de representações não simbólicas:

$$C = \{i_0, i_1, i_2, i_3, i_4, i_5\}$$

Onde i_0 representa a imagem de uma árvore, i_1 representa a imagem de um livro, i_2 representa a imagem de um cachorro, i_3 representa a imagem de um carro, i_4 representa a imagem de um gato e i_5 representa a imagem de uma segunda árvore.

Em contrapartida, teremos também um conjunto de símbolos, que inicialmente não possuem um representante externo, são apenas *tokens* que fazem parte de um determinado sistema simbólico:

$$S = \{arvore, livro, cachorro, carro, gato\}$$

O embasamento simbólico se dá na forma de uma função que relaciona uma entrada que pertence ao conjunto de dados sensoriais C a um dos símbolos do conjunto S .

$$f : C \rightarrow S$$

A função funcionaria de forma tal que:

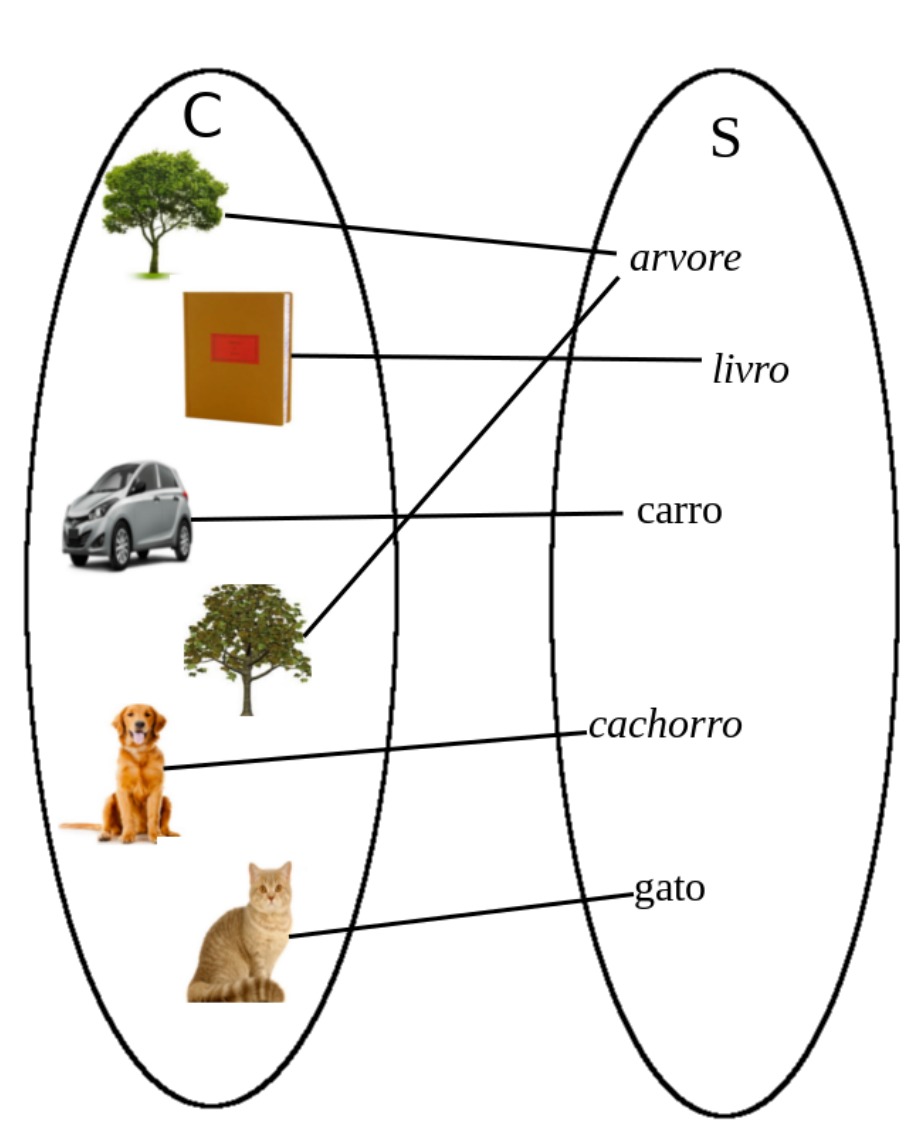
Figura 19 – Função que transforma uma percepção não-simbólica ao seu equivalente simbólico.

$$f \left(\img alt="A golden retriever puppy sitting down." data-bbox="350 615 440 670" \right) = cachorro$$

Fonte: o autor

No conjunto de percepções sensoriais temos duas imagens de árvores diferentes. O símbolo *arvore* no conjunto S então representa uma categoria que possui mais de uma representação. A função deve ser capaz de tratar de casos assim, de forma que se apresentado com qualquer uma das imagens de árvore a função é capaz de identificar corretamente que ambas imagens representam um mesmo símbolo.

Figura 20 – Relação entre conjunto de dados não-simbólicos e conjunto de símbolos.



Fonte: o autor

No caso deste trabalho, estamos lidando com esse mesmo problema no contexto de agentes multi-contexto, ou seja, sistemas que consistem em um conjunto de módulos, chamados contextos, que permitem que cada um destes possam ser considerados como uma lógica e um conjunto de fórmulas escritas nessa lógica, e um conjunto de regras de ponte para transferir informações entre contextos. Uma maior explicação deste tipo de sistemas foi apresentado em 2.3, juntamente com os objetivos e vantagens de tais sistemas.

Unidades dos sistemas multi-contexto representam os vários componentes da arquitetura. Eles contêm a maior parte do conhecimento de solução de problemas de um agente, e esse conhecimento é codificado na teoria específica que a unidade encapsula. A natureza das unidades varia entre as arquiteturas.

Neste trabalho, o sistema será modelado na forma de um agente BDI (2.2.2).

Portanto, teremos unidades que representam crenças, desejos e intenções, onde a lógica associada a cada unidade fornece a linguagem na qual as informações nesta unidade são codificadas e as regras de ponte fornecem o mecanismo pelo qual as informações são transferidas entre as unidades.

Para a construção de agentes BDI como sistemas multi-contexto, será utilizada a linguagem Sigon. A linguagem é descrita em detalhes na seção de Trabalhos Relacionados (3.1), apresentando sua teoria, arquitetura e as definições da linguagem em si.

O *framework* utilizado para implementação do agente na linguagem Sigon é o trabalho de Arnhold (2018), que também foi descrito em maiores detalhes em 3.2.

O modelo genérico do agente, conforme definido em Gelaim et al. (2018) é descrito na forma:

$$Ag = \langle CC \cup \bigcup_{i=0}^n C_i, \Delta_{rp} \rangle$$

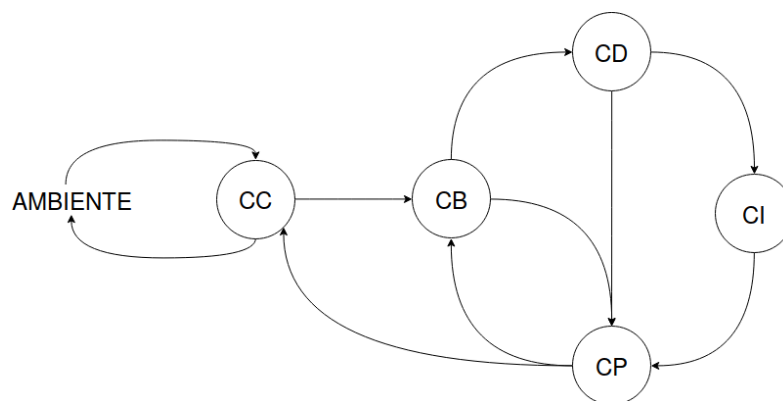
Onde CC é o contexto de comunicação. C_i com $1 \leq i \leq n$ são os demais contextos do agente e Δ_{rp} é o conjunto de regras ponte, que conectam dois ou mais contextos (GELAIM et al., 2018).

Com base nesse modelo, é definido uma arquitetura de agente BDI:

$$Ag_{BDI} = \langle \{CC, CB, CD, CI, CP\}, \Delta_{rp} \rangle$$

Onde CC , CB , CD , CI , CP são, respectivamente, contextos de comunicação, crenças, desejos, intenções e planejamento. Para auxiliar e otimizar o desenvolvimento ou a prototipação de agentes, o trabalho de Arnhold (2018) implementa esses contextos por padrão no *framework* de desenvolvido em Sigon.

Figura 21 – Estrutura do agente BDI em Sigon.



Fonte: o autor

As regras de ponte, são responsáveis pela troca de conhecimento entre os contextos do sistema multi-contexto. Para facilitar o desenvolvimento de agentes BDI, um conjunto de regras que representam o comportamento BDI são implementadas pelo trabalho de [Arnhold \(2018\)](#), com o objetivo de permitir a execução do cenário proposto

As regras pontes do agente utilizado serão as seguintes regras:

$$\Delta_{rp} = \left\{ \begin{array}{l} \frac{CC : sense(\varphi)}{CB : \varphi} \\ \\ \frac{CD : \varphi \text{ and } CB : not \varphi \text{ and } CI : not \varphi \text{ and } CP : plan(\varphi, \alpha, Pre, Post, c_a)}{CI : \varphi} \\ \\ \frac{CP : plan(\varphi, \alpha, Pre, Post, c_a) \text{ and } CI : \varphi \text{ and } CB : Pre}{CC : \alpha} \end{array} \right\}$$

Assim sendo, como objetivo deste trabalho, gostaríamos que o Contexto de Comunicação (responsável por lidar com os sensores e atuadores) do agente fosse capaz de perceber algum dado visual no ambiente e que com base no que foi percebido, associar essa percepção a alguma representação simbólica da percepção sensorial. Uma vez que a forma simbólica fosse adicionada as bases de conhecimento do agente, seria possível utilizar essa forma simbólica para que o agente operasse normalmente, realizando seus planos conforme este símbolo que foi percebido por meio de estímulos sensoriais no ambiente.

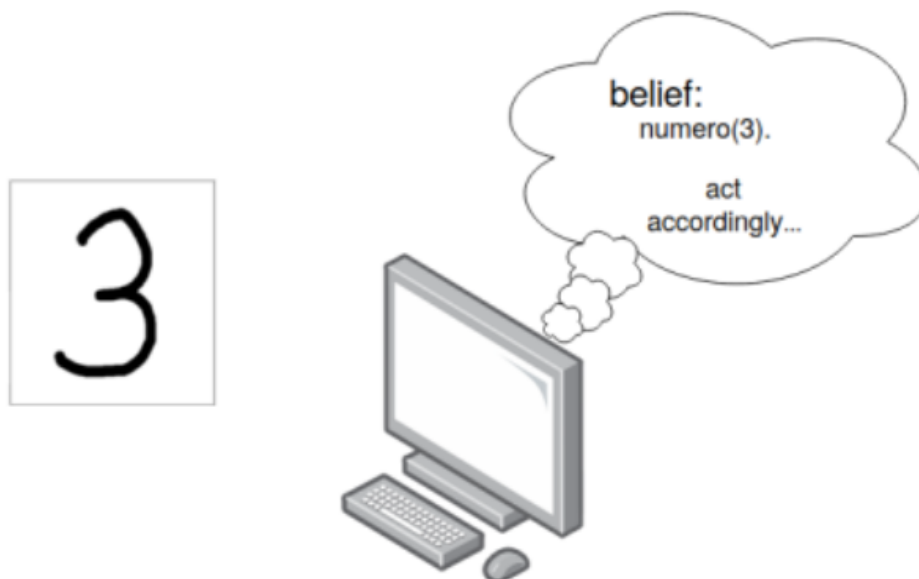
$$\frac{CC : sense(\varphi)}{CB : \varphi}$$

Essa regra de ponte é responsável por copiar as percepções $sense(\varphi)$ do contexto de comunicação (CC) como φ para o contexto de crenças (CB).

Quando um objeto visual for identificado no ambiente pelos sensores pertencentes ao Contexto de Comunicação, essa regra de ponte deve ser ativada, colocando no contexto de crenças (CB) um símbolo que represente aquilo que foi percebido no ambiente.

Dessa forma, apesar de ser de uma maneira limitada, estaremos simulando o comportamento de um agente que consegue formar conexões entre percepções não-simbólicas do ambiente com equivalentes simbólicos internos do agente.

Figura 22 – Processo de reconhecimento de estímulo sensorial visual em um agente artificial que consegue relacionar a percepção visual a um símbolo.



Fonte: o autor

4.2 TEORIA

Conforme foi abordado em 2.6, diversas propostas foram criadas para tentar solucionar o problema do embasamento simbólico. Apesar de todas essas propostas terem suas críticas, iremos adotar a proposta de [Harnad \(1990\)](#), como descrita em 2.6.1.1.

Consideramos essa abordagem como apropriada para o trabalho pois ela foi a primeira abordagem proposta para solução do problema do embasamento de símbolos e se trata de uma abordagem muito simples para tentar embasar símbolos, sendo quase intuitiva para aqueles familiarizados com as capacidades de abordagens conexionistas. Portanto, faz sentido que uma tentativa inicial para embasar símbolos no contexto de agente como sistema multi-contexto seja simples, para que possa se desenvolver de forma mais complexa com esforços futuros.

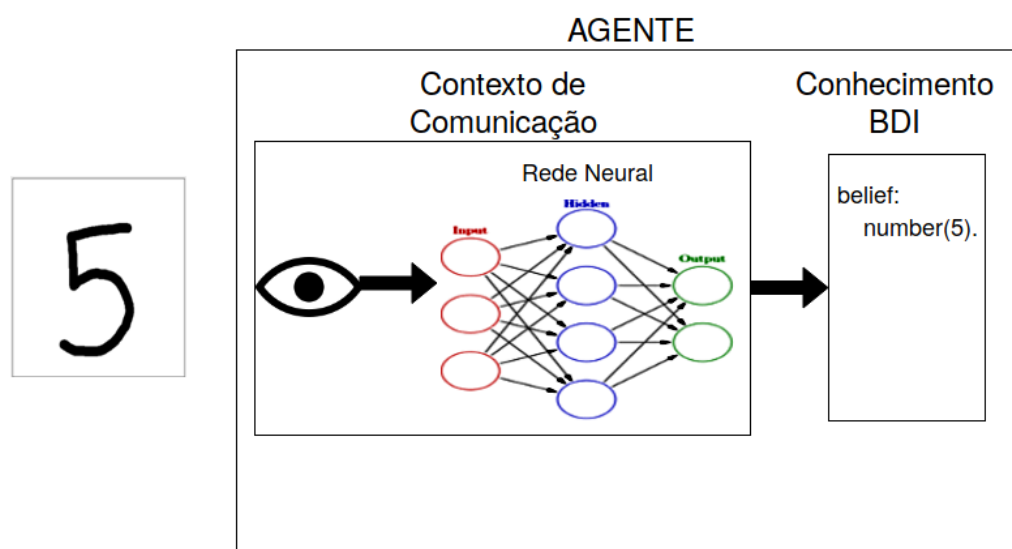
Outro motivo para a utilização desta abordagem é a possibilidade de contribuir para o debate entre defensores de inteligências artificiais simbólicas e inteligências artificiais conexionistas, oferecendo uma terceira alternativa: uma abordagem híbrida. Um sistema híbrido que faz uso de algoritmos conexionistas e simbólicos aproveitará os pontos fortes de ambos ao mesmo tempo em que neutraliza as fraquezas um do outro. Os limites do uso de uma técnica isolada já estão sendo identificados ([MARCUS, 2018](#)) e as pesquisas mais recentes começaram a mostrar que a combinação de ambas as abordagens pode levar a uma solução mais inteligente ([MAO et al., 2019](#)).

A estratégia sugerida por [Harnad \(1990\)](#) é baseada em um modelo híbrido. Harnad

quer superar os limites típicos encontrados pelos sistemas simbólicos e conexionistas puros por meio de uma implementação que mistura características de um sistema simbólico e de um sistema conexionista.

Um sistema híbrido não-simbólico/simbólico será esboçado, no qual os significados dos símbolos são embasados pela capacidade do sistema de discriminar e identificar os objetos aos quais se referem.

Figura 23 – Modelo de um agente que lida com a percepção de forma a transformá-la em um símbolo.



Fonte: o autor

Para fundamentar a sua abordagem, [Harnad \(1989\)](#) chama atenção para uma das propriedades mais notáveis da percepção humana: a capacidade de separar o mundo em suas “articulações”. Os sinais físicos que bombardeiam nossas superfícies sensoriais não dão origem a uma confusão desordenada de dados, mas sim a experiências relativamente ordenadas, segmentadas em “pedaços” ou categorias. Este fenômeno na percepção humana e animal foi chamada por [Harnad \(1989\)](#) de “percepção categórica”.

[Harnad e Damper \(2000\)](#) aponta que a percepção categórica pode surgir como um efeito colateral natural dos meios pelos quais modelos de redes neurais realizam o aprendizado, devido a suas capacidades intrínsecas de discriminação (a capacidade de distinguir pares de estímulos, um julgamento relativo) e identificação (a capacidade de categorizar ou nomear estímulos individuais, um julgamento absoluto).

As redes neurais são um mecanismo possível para aprender os invariantes na projeção sensorial analógica em que se baseia a categorização bem-sucedida. Elas adquirem a capacidade de classificar suas entradas nas categorias impostas pela aprendizagem supervisionada, alterando as distâncias entre pares (onde a distância é o grau em que um par de

entradas é discriminável pela rede) até que haja compressão suficiente das características invariantes dentro de uma categoria para realizar categorização confiável.

Portanto, há uma estreita conexão entre a capacidade sensório-motora de separar o mundo em suas articulações e a capacidade cognitiva de produzir descrições simbólicas desse mundo: podemos dar nomes para os “pedaços” do mundo originados de nossos receptores sensoriais, e esses nomes de categorias podem então ser combinados sintaticamente para formar proposições sobre o mundo. Qualquer mecanismo que mapeie com sucesso as projeções sensoriais em seus nomes de categorias é o que fundamenta o sistema de símbolos (HARNAD, 1989).

A “percepção categórica” acaba por ser exibida tanto por pessoas quanto por redes neurais, e pode mediar as restrições exercidas pelo mundo analógico dos objetos no mundo formal dos símbolos. Dessa forma, a percepção categórica é de interesse não só por si só, como um fenômeno perceptual muito básico, mas também como um possível contribuinte para resolver o “problema do embasamento simbólico” (HARNAD, 1989).

Portanto, devido a sua capacidade de simular a percepção categórica, redes neurais tornam possível conectar símbolos e referentes usando os dados perceptivos e as características invariantes das representações categóricas (HARNAD; DAMPER, 2000).

Levando em conta as três etapas para formação de categorias e embasamento de símbolos propostas por Harnad (1990): iconização, discriminação e identificação. Cada estágio é responsável por uma função:

1. iconização: o processo de transformação de dados sensoriais em representações icônicas;
2. discriminação: o processo de julgar se duas entradas são as mesmas;
3. identificação: o processo de atribuição de um nome a uma classe de insumos.

Iconização e discriminação são subprocessos, realizados por meio de redes neurais, que podem ser usadas para encontrar padrões nos dados. Esses processos tornam possível a associação subsequente de um nome com uma classe de entrada e subsequentemente, a nomeação dos referentes. No entanto, por si só, as redes neurais são incapazes de produzir representações simbólicas, então eles ainda não podem permitir o agente artificial desenvolver capacidades simbólicas. Levando isso em conta, é fornecido um sistema simbólico, que pode manipular símbolos sintaticamente e finalmente conseguir um embasamento semântico de seus símbolos.

As representações em (1) são obtidas a partir de um conjunto de treinamento, que simulará as experiências que um agente teria relacionando percepções ao mesmo tipo de objeto. As representações categóricas então serão obtidas através do processo de

discriminação em (2). Ou seja, um novo dado de entrada será enviado à rede neural e será discriminado por essa, de modo a identificar a qual categoria esse novo dado pertence.

O agente então possui símbolos, que em (3) serão associadas às representações categóricas, provendo assim um referente apropriado aos símbolos.

Assim, redes neurais parecem ter a capacidade de ser um sistema que forma e separa tipos de representações em categorias conforme aprendem a reconhecer dados no processo de treinamento. A percepção categórica parece ser um meio para um fim: as entradas que diferem entre si são “compactadas” em representações internas semelhantes se todas elas geram a mesma saída; e elas se tornam mais separadas se geram saídas diferentes. O ‘viés’ da rede é o que filtra as entradas em sua categoria de saída correta. As redes realizam isso seletivamente detectando (depois de muita tentativa e erro, guiado por *feedback* corretivo de correção) os recursos invariantes que são compartilhados pelos membros da mesma categoria e que os distinguem de maneira confiável de membros de diferentes categorias; as redes aprendem a reconhecer todas as outras variações como irrelevantes para a categorização (HARNAD; HANSON; LUBIN, 1991).

De acordo com Harnad, os símbolos manipulados por um agente artificial podem ser embasados conectando-os aos dados perceptivos que serão interpretados por uma rede neural. Dessa forma, a conexão é estabelecida categorizando os sinais sensorio-motores.

Como será utilizado um algoritmo de aprendizagem supervisionado, a rede será treinada usando um conjunto de treinamento de dados rotulados pré-selecionados por um agente externo e cuja saída correta já é conhecida. Dessa forma, para preparar um conjunto de treinamento para a rede neural, ainda necessitamos de um agente externo capaz de categorizar os dados que serão usados, portanto precisamos de alguém capaz de atribuir um nome (símbolo) para os dados perceptivos que serão usados.

Essa é a crítica de Floridi (2011) à proposta de Harnad para solução do problema de embasamento simbólico, pois isso significa simplesmente ter um sistema como a tabela de consulta interna. Os símbolos estão tendo seus significados fundamentados em alguma categoria perceptual nomeada por um humano.

O problema do embasamento simbólico diz respeito à possibilidade de especificar precisamente como uma pessoa pode inicialmente elaborar autonomamente sua própria semântica para os símbolos que manipula e fazê-lo do zero, interagindo com seu ambiente e outras pessoas. Nenhum treinador está presente. O sistema simbólico deve evoluir apenas através da interação com outras pessoas e o meio ambiente. No começo não podemos ter um treinador com algum conhecimento básico.

Essencialmente, a rede neural só está aprendendo a copiar o ser humano, e não desenvolvendo a mesma habilidade cognitiva dos seres humanos de atribuir novos nomes à novas categorias perceptivas, então, o significado do símbolo ainda seria parasitário das

interpretações de um humano.

Isso violaria a condição de um agente ser autônomo e portanto desenvolver categorias autonomamente. Isso implicaria que o problema do embasamento simbólico continua em aberto.

Devido ao escopo do trabalho, o trabalho não irá lidar com o criticismo de [Floridi \(2011\)](#) de que este tipo de treinamento é inteiramente extrínseco, e portanto irá violar a condição para solução do problema do embasamento simbólico que determina que o agente deve funcionar de maneira autônoma.

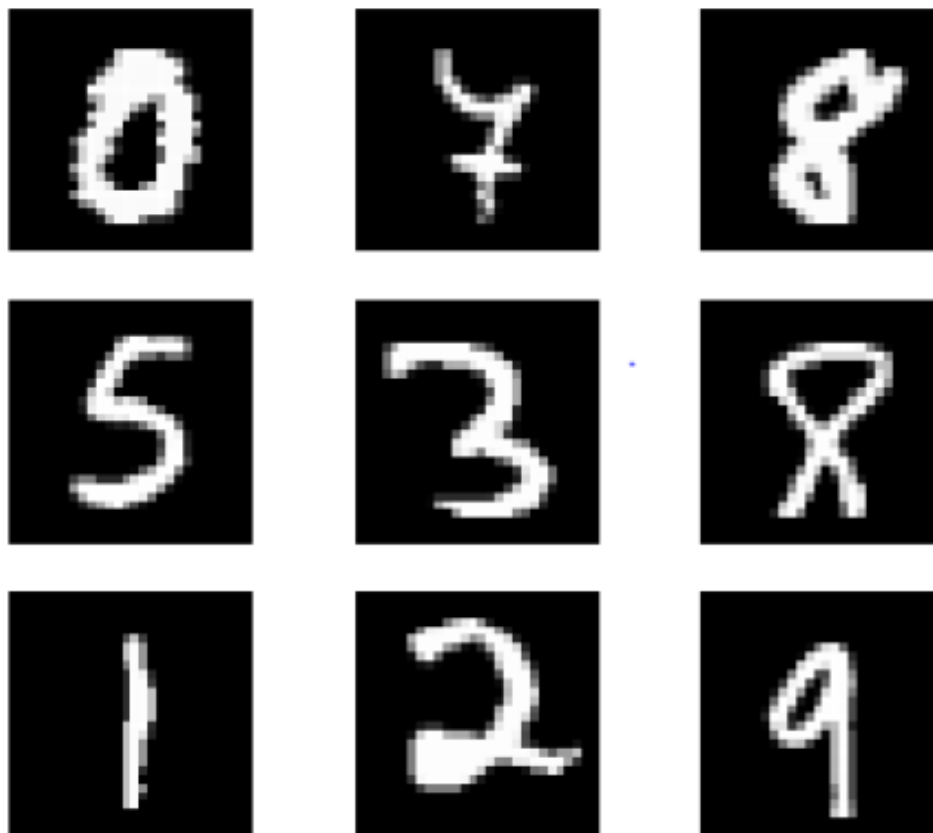
Ainda sim, o modelo dado por [Harnad \(1990\)](#) pode servir para emular a aprendizagem das categorias no caso das crianças quando elas adquirem a linguagem. Eles percebem as coisas ao seu redor, categorizam-nos e rotulam-nos com base nos dados que recebem dos pais. Portanto, apesar de não ser o suficiente para solucionar o problema do embasamento simbólico, esse modelo ainda pode ter suas contribuições ao desafio.

4.3 IMPLEMENTAÇÃO

Neste trabalho, uma rede neural é implementada para identificar números manuscritos fornecida por um conjunto de dados chamado MNIST¹. O banco de dados MNIST é um grande banco de dados de dígitos manuscritos que é comumente usado para treinamento de vários sistemas de processamento de imagens.

¹ <http://yann.lecun.com/exdb/mnist/>

Figura 24 – Exemplo de imagens do MNIST.



Fonte: Wikimedia

O conjunto de dados MNIST fornece um método para a avaliação do aprendizado de máquina no problema de classificar os dígitos manuscritos. Este conjunto de dados é um problema desenvolvido por Yann LeCun, Corinna Cortes e Christopher Burges para avaliar modelos de aprendizado de máquina. O conjunto de dados foi construído com um grande número de documentos disponíveis a partir do NIST, a partir dos quais o nome é NIST ou MNIST conjunto de dados vem.

Em cada imagem, o número é relativamente centrado na imagem. Além disso, em um grau variável, todos os dígitos são escritos na vertical, então o conjunto de dados é altamente regularizado.

As imagens em preto e branco do NIST foram normalizadas para caber em uma caixa delimitadora de 28 x 28 *pixels* e *anti-aliased*, que introduziu níveis de escala de cinza. O banco de dados MNIST contém 60.000 imagens de treinamento e 10.000 imagens de teste.

As imagens serão processadas por uma rede neural que utiliza esse conjunto de treinamento para formar as categorias na qual os dados de teste e os dados sensoriais recebidos pelo agente serão enquadrados.

Neste trabalho, uma rede neural convolucional foi treinada pela identificação de imagens em MNIST banco de dados digital manuscrito para prever exatamente o que os números na imagem são. O reconhecimento dos dígitos consiste em avaliar a imagem e classificá-la em um dos 10 dígitos possíveis (0 a 9).

Foi escolhida essa base de dados para este projeto porque sua baixa dimensionalidade torna este um conjunto de dados computacionalmente viável para minerar com poder de computação limitado.

Essas imagens de dígitos vão simular outros tipos de dados que estão sendo percebidos no ambiente. Então, as imagens de dígitos podem ser um tanto limitadas no sentido de aplicações reais para o projeto, porém, elas podem ser vistas como substitutos genéricos de diferentes tipos de objetos que poderiam ser encontrados no ambiente.

Para este trabalho, foi implementada uma rede neural simples, com base no código do *site Neural Networks and Deep Learning*^{2,3}. O autor do *site* implementa uma rede neural na linguagem *Python*, mas para facilitar a integração da rede neural com o agente deste trabalho, a rede foi implementada em *Java*, a mesma linguagem que é utilizada para implementar o agente.

Como o reconhecimento de dígitos é uma tarefa relativamente fácil, podemos utilizar uma rede neural de três camadas.

A camada de entrada da rede contém neurônios que codificam os valores dos pixels de entrada. Como nossos dados de treinamento para a rede consistirão em muitas imagens de dígitos manuscritos digitalizados de 28 por 28 pixels, assim a camada de entrada contém $784 = 28 \times 28$ neurônios.

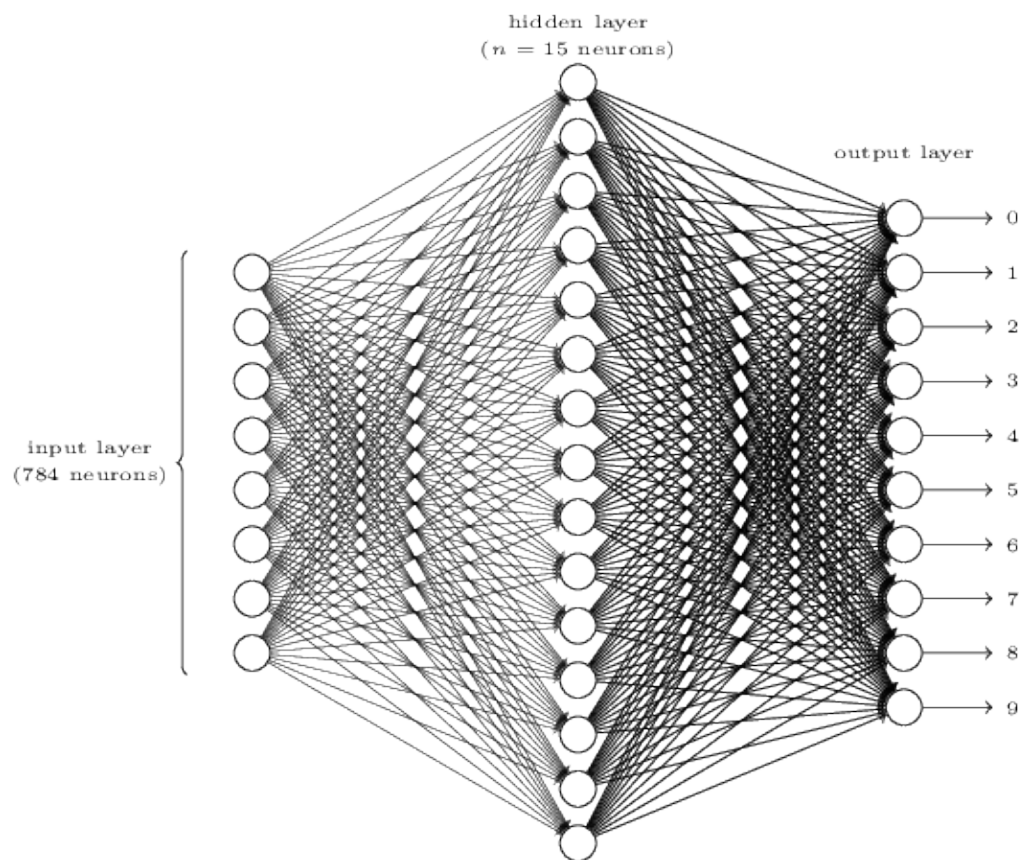
A segunda camada da rede é uma camada oculta. Denotamos o número de neurônios nessa camada oculta por n , e vamos experimentar valores diferentes para n . O valor final que foi utilizado foi o valor de $n = 30$, dando um resultado de até cerca de 93% de categorizações no processo de testes.

A camada de saída da rede contém 10 neurônios. São numerados os neurônios de saída de 0 a 9, e descobrimos qual neurônio tem o maior valor de ativação. Se o primeiro neurônio disparar, isso indicará que a rede acha que o dígito é 0. Se o segundo neurônio disparar, isso indicará que a rede acha que o dígito é 1. E assim por diante para os outros neurônios de saída.

² <https://github.com/mnielsen/neural-networks-and-deep-learning>

³ <http://neuralnetworksanddeeplearning.com>

Figura 25 – Estrutura da rede neural



Fonte: Página Neural Network and Deep Learning⁴

A vantagem de utilizar uma rede neural simples é que devido a sua simplicidade, podemos entender que a capacidade para embasamento buscada por Harnad não depende tanto de uma arquitetura ou tipo específico de rede neural, mas sim de suas capacidades que surgem como efeito colateral do processo de aprendizagem e de seu funcionamento, a saber, o processo de iconização e discriminação, ou seja, a capacidade de categorização.

Além disso, por ser uma rede neural simples, podemos implementar ela desde os componentes mais básicos, dessa forma podemos entender melhor como os processos de iconização e discriminação surgem dentro da rede neural, eliminando assim certos aspectos que aparecem como uma caixa preta ao utilizar bibliotecas que trazem redes neurais prontas.

A desvantagem de não utilizar uma biblioteca específica para redes neurais é que perdemos flexibilidade para customização da estrutura da rede. Com o uso de bibliotecas como Tensorflow⁵ o usuário pode modificar a arquitetura da rede neural com muita facilidade, bastando mudar alguns valores ou adicionando poucas linhas de código. No caso de nossa implementação, para o usuário experimentar com as redes neurais seria necessário

⁵ <https://www.tensorflow.org>

modificar diretamente o código da implementação.

Além dessas desvantagens relacionadas à implementação em si, a rede neural também terá a limitação de que, não importa qual o conteúdo desenhado no *Canvas* pelo usuário, a rede sempre vai tentar interpretar o desenho como sendo uma das categorias nas quais a rede foi treinada. Portanto, se o usuário desenhar um número como "31", a rede neural vai tentar categorizar essa entrada como uma de suas categorias. Outro caso, ainda mais extremo, seria o caso do usuário desenhar linhas aleatórias. A rede neural não será capaz de identificar o desenho como algo sem sentido e irá tentar classificar a entrada como sendo algum número de 0 à 9.

Agora que temos uma arquitetura para nossa rede neural, como ele pode aprender a reconhecer dígitos? Aprendizagem com exemplos é de longe o tipo de aprendizado mais estudado em IA e pode ser visto como um paralelo ao aprendizado por ser explicitamente ensinado. Nesse tipo de aprendizado, o supervisor induz uma descrição de categoria a partir de exemplos pré-classificados e contra-exemplos da categoria fornecidos por algum tipo de professor.

Como há um agente externo presente para guiar o processo de aprendizagem, esse tipo de aprendizado é uma instância de aprendizagem supervisionada. Assim, é o supervisor que decide quando criar um novo conceito.

Do ponto de vista clássico, a tarefa para este tipo de aprendizagem pode ser vista como encontrar uma definição de conceito (isto é, descrição) consistente com todos os exemplos positivos mas sem exemplos negativos no conjunto de treino.

A Rede Neural Artificial com múltiplas camadas ocultas possui uma excelente capacidade de aprendizado de características. As características obtidas da aprendizagem têm uma descrição essencial aos dados, facilitando a visualização ou classificação.

Uma vez que a rede neural foi satisfatoriamente treinada, podemos mandar imagens para ela fazer sua função de classificação.

Como no contexto desse trabalho o tipo de imagens que serão usadas são as de dígitos escritos a mão, foi feito um programa que permite que o usuário desenhe os números com o mouse, de forma a facilitar a entrada de dados.

Esse programa simula a função de um sensor de visão do agente, então futuramente este sistema poderia ser substituído por uma câmera que captura imagens do ambiente para que a rede neural seja captar em tempo real o que está presente no ambiente em que se encontra.

Como aplicação elevaria muito a complexidade do projeto, usaremos apenas o *Canvas* de desenho por mouse para simular os dados não-simbólicos encontrados no ambiente.

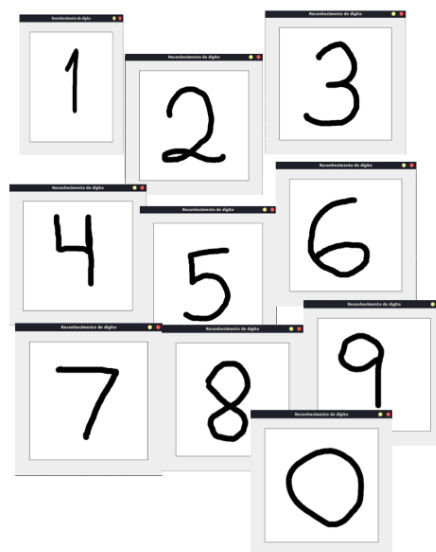
Uma janela é aberta com um *Canvas* que permite a interação com o usuário:

Figura 26 – Canvas para usuário desenhar número.



Fonte: o autor

Figura 27 – Canvas com os números desenhados pelo usuário.



Fonte: o autor

O *Canvas* que permite o usuário desenhar em si não possui nenhuma peculiaridade, ele apenas trata os eventos do mouse do usuário de modo a permitir desenhar no *Canvas*. Em termos de usabilidade, o usuário deve desenhar o número de uma única vez, pois ao largar o botão do mouse, é acionado um evento que interpreta que o desenho está pronto e coloca-o em forma de *BufferedImage*, permitindo armazenar a cor de cada pixel que compõem a imagem.

É então que um passo fundamental é realizado: a imagem passa por um processo de normalização para se adequar ao padrão das imagens utilizadas para treinamento, ou seja, as imagens do conjunto MNIST.

```
public static BufferedImage normalizedImg(BufferedImage image) {
    BufferedImage normalizedImg = ImageUtils.deepCopy(image);
    normalizedImg = ImageUtils.cropWhiteBorder(normalizedImg);
    normalizedImg = ImageUtils.invertColor(normalizedImg);
    normalizedImg = ImageUtils.fitInto(normalizedImg, 20, 20,
        Color.BLACK.getRGB());
    normalizedImg = ImageUtils.centerWithMassInto(normalizedImg, 28, 28,
        Color.BLACK.getRGB());
    return normalizedImg;
}
```

Nesse passo, a imagem é recortada, as cores são invertidas, a imagem é colocada

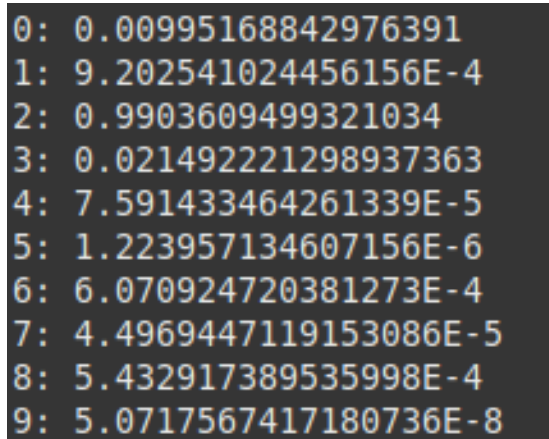
no tamanho adequado e finalmente o número é centralizado.

Uma vez que a imagem foi devidamente normalizada, ela está apta a ser interpretada pela rede neural:

```
. . .  
ArrayList<Double> result = Neural.run(input);  
. . .
```

O resultado da execução da rede neural é um *array* cujos elementos representam um valor numérico que representa o quanto a imagem se adequou a cada categoria. Portanto, temos um *array* de tamanho 10, onde a posição de índice 0 vai possuir um número que representa o quão similar a imagem é a categoria que representa as imagens de número 0, a posição de índice 1 vai possuir um número que representa o quão similar a imagem é a categoria que representa as imagens de número 1, e assim por diante.

Figura 28 – Resultado da execução da rede neural.



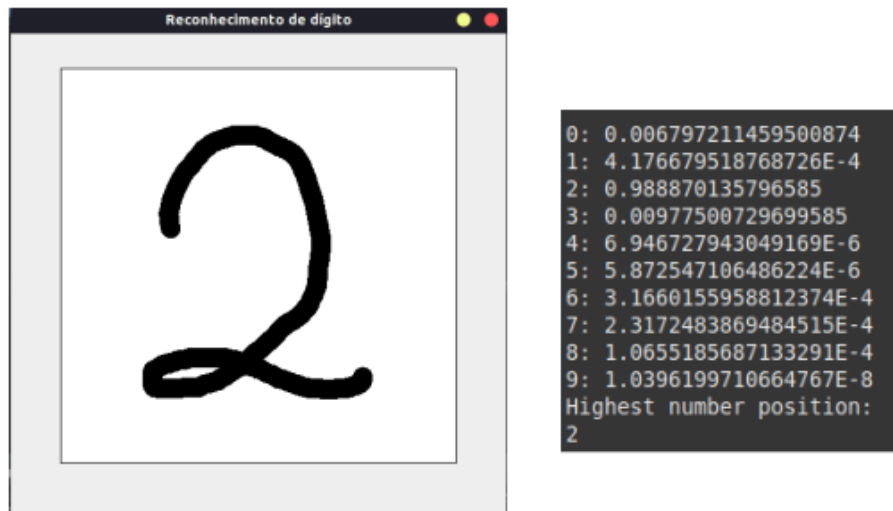
```
0: 0.00995168842976391  
1: 9.202541024456156E-4  
2: 0.9903609499321034  
3: 0.021492221298937363  
4: 7.591433464261339E-5  
5: 1.223957134607156E-6  
6: 6.070924720381273E-4  
7: 4.4969447119153086E-5  
8: 5.432917389535998E-4  
9: 5.0717567417180736E-8
```

Fonte: o autor

Portanto, devemos percorrer o *array* e buscar qual índice contém o maior número, dessa forma podemos identificar a qual categoria a imagem de entrada mais se adequou e esse provavelmente será o número a qual a imagem representa.

Caso o usuário desenhe um número “2” no *Canvas*, teremos algo similar à imagem abaixo.

Figura 29 – Resultado da execução da rede neural com o desenho de um número “2” como entrada.



Fonte: o autor

Por fim foi implementado um agente na linguagem Sigon, que permite a especificação de um agente como sistema multi-contexto. Seguindo as definições da linguagem de Gelaim et al. (2018) e o *framework* de implementação de Arnhold (2018), iniciamos com a definição do agente em um arquivo “.on” seguindo as especificações da linguagem.

```

communication:
sensor("visual", "sg.Camera").
actuator("acharNumero", "sg.Recognize").

beliefs:

desires:

intentions:
  terNumero(X).

planner:
  plan(
    terNumero(X),
    [
      action(acharNumero(X))
    ], _, [numero(X)]).
  
```

A comunicação do agente com seu ambiente ou com outros agentes é feita pelo

contexto de comunicação por meio de sensores e atuadores, não acoplados a uma estrutura de ambiente. No agente, especificamos o Contexto de Comunicação como:

```
communication:
    sensor("visual", "sg.Camera").
    actuator("acharNumero", "sg.Recognize").
```

Dessa forma, teremos um sensor que irá perceber o ambiente e um atuador que irá procurar um número. No sensor, o argumento “visual” representa a identificação e o argumento “*sg.Camera*” representa a implementação do sensor. O mesmo acontece no caso do atuador, “acharNumero” é a identificação e “*sg.Recognize*” representa a implementação do atuador.

Em termos de implementação, um sensor contém um publicador de literais, seguindo o paradigma da programação reativa. Considerando que o agente pode ter vários sensores que podem capturar percepções de forma paralela, um Sensor é tratado como uma *Thread* implementando a interface *Runnable*.

```
public class Camera extends Sensor{
    public static final PublishSubject<String> msg = PublishSubject.create();
    public void run() {
        msg.subscribe(super.publisher);
    }
}
```

Internamente, o publicador será assinado pelo Contexto de comunicação. Ao receber uma percepção ela é persistida no contexto de comunicação e o conjunto de regras de ponte é executado, sintetizando um ciclo de raciocínio

Para alterar o ambiente ou enviar mensagens para outras entidades, o agente utiliza seus atuadores. Um atuador é definido pelo *framework* como uma classe abstrata.

```
public class Recognize extends Actuator{
    @Override
    public void act(List<String> args) {
        Main.acharNumero();
    }
};
```

E implementamos uma função tal qual:

```
public static void acharNumero() {
    String number = ImageInput.launch();
    Camera.msg.onNext("numero(" + number + ").");
}
```

 }

A função começa executando uma função que é responsável por abrir um *Canvas* para o usuário desenhar o número manualmente, como já foi explicado anteriormente. O número desenhado será processado pela rede neural e por fim retornará uma *String* que corresponde a qual número foi identificado pela rede neural. O valor obtido então é recebido pelo sensor.

Aqui regra de ponte responsável por copiar as percepções $sense(\varphi)$ do contexto de comunicação é ativada e φ é posto no contexto de crenças (*CB*).

$$\frac{CC : sense(\varphi)}{CB : \varphi}$$

Dessa forma, é aqui que quando um objeto visual for identificado no ambiente pelos sensores pertencentes ao Contexto de Comunicação, a rede neural irá fazer a conexão entre a representação visual e uma representação simbólica. A regra de ponte será ativada, e a representação simbólica do que foi identificado no ambiente será colocado no contexto de crenças (*CB*).

Então é aqui que é simulado o comportamento de um agente que consegue formar conexões entre percepções não-simbólicas do ambiente com equivalentes simbólicos internos do agente.

A outra parte do agente é responsável por definir os elementos de crenças, desejos e intenções:

```

beliefs:
desires:
intentions:
    terNumero(X).

planner:
    plan(
        terNumero(X),
        [
            action(acharNumero(X))
        ], _, [numero(X)]).
  
```

No caso, iniciamos um agente sem nenhuma crença e nenhum desejo, porém o agente possui a intenção de $terNumero(X)$ e de acordo com os planos definidos pelo agente no seu Contexto de Planos, quando ele possuir a intenção $terNumero(X)$ ele irá executar a ação de $acharNumero(X)$, que irá executar a função definida pelo atuador,

explicada anteriormente. Ao terminar o plano, como descrito na pós-condição, o agente espera ter um numero(X).

```
planner:  
  plan(  
    terNumero(X),  
    [  
      action(acharNumero(X))  
    ], _, [numero(X)]).
```

4.4 RESULTADOS

Como já mostrado anteriormente, a rede neural é capaz de identificar corretamente o número desenhado pelo usuário.

Figura 30 – Resultado da execução da rede neural com o desenho de um número “1” como entrada.



Fonte: o autor

Isso possibilita realizarmos a conexão entre a percepção sensorial e o símbolo que equivale à percepção.

No caso do agente, ao ser iniciado, o agente recebe uma percepção de que está sendo executado, e portanto, esta é a única crença presente no agente até o momento.

Figura 31 – Dados do agente antes de ser detectado uma percepção pelos sensores.

```
CB running.

CD
CI terNumero(X).
CC sense(running).
```

Fonte: o autor

Porém, como o agente possui a intenção “terNumero(X).”, o plano definido pelo agente será executado, e a ação “acharNumero(X)” será executado pelo atuador.

```
planner:
  plan(
    terNumero(X),
    [
      action(acharNumero(X))
    ], _, [numero(X)]).
```

O *Canvas* será aberto para o usuário desenhar um número e a rede irá fazer a categorização da imagem e o sensor irá detectar que há um certo número no ambiente.

Figura 32 – Percepção do agente.

```
CC sense(numero(2)).
```

Fonte: o autor

Portanto a regra de ponte será ativada, e o contexto de crenças será atualizado.

$$\frac{CC : sense(\varphi)}{CB : \varphi}$$

Figura 33 – Dados do agente antes de ser detectado uma percepção pelos sensores.

```
CB running.

numero(2).

CD
CI terNumero(X).
CC sense(numero(2)).
```

Fonte: o autor

Dessa forma o banco de conhecimentos do agente agora possui a informação de que uma percepção sensorial do número 2 foi encontrada no ambiente. Caso o agente tivesse algum outro plano que dependesse de existir um número no ambiente, ele seria agora executado.

Portanto, de uma percepção visual não-simbólica temos agora um símbolo que equivalente e poderíamos realizar operações sobre os símbolos normalmente.

Com isso conseguimos uma função que transforma o dado de entrada em uma representação simbólica.

Figura 34 – Função que transforma uma percepção não-simbólica vinda do Canvas ao seu equivalente simbólico.

$$f \left(\text{Canvas}(2) \right) = \text{numero}(2).$$

Fonte: o autor

Conclusão

Para termos uma teoria cognitiva coerente, devemos estabelecer como é feita a conexão entre sobre o quê a mente pensa e o quê é percebido no mundo real. Do ponto de vista da Inteligência Artificial, essa é a questão da capacidade de se conceber um sistema artificial capaz de inventar e usar símbolos fundamentados em suas interações com o mundo externo e com outros agentes. Esta é uma das questões mais fundamentais da cognição, a saber, o problema de embasamento de símbolos. Vários pesquisadores, mais notavelmente Searle, argumentaram que esse problema nunca será resolvido em sistemas artificiais porque esses sistemas são apenas manipuladores sintáticos, estes sistemas não compartilham de propriedades essenciais presentes na mente humana que permite tal habilidade cognitiva.

A alegação de que a manipulação sintática não é suficiente para o significado ou o pensamento é uma questão significativa, com implicações mais amplas do que a inteligência artificial. Teorias proeminentes da mente sustentam que a cognição humana é computacional. Sem uma proposta convincente para resolução do problema do embasamento de símbolos, não apenas as tentativas de construir uma inteligência artificial por meio de uma perspectiva simbólica podem estar fadadas ao fracasso, mas também essas principais abordagens para entender a cognição humana se mostrarão equivocadas.

O paradigma da inteligência artificial simbólica não sofre apenas deste problema que permanece aberto, mas também por causa dos recentes grandes sucessos práticos de abordagens conexionistas, que são muitas vezes acompanhados por uma hostilidade à inteligências artificiais de manipulação de símbolos. Frente às conquistas no campo prático, muitos concluem que os horizontes para inteligências artificiais conexionistas são ilimitados e que a antítese histórica dessas abordagens conexionistas - manipulação de símbolos - deveria ser substituída. A manipulação de símbolos, a mensagem parece ser, logo será deixada no lixo da história.

Porém, precisamos considerar os difíceis desafios da inteligência artificial e não ficar satisfeitos com avanços incrementais de curto prazo, sem descartar, é claro, o que foi aprendido por meio destes avanços. Apesar da proeminência dos sucessos práticos recentes, em um nível teórico, o conexionismo também atraiu críticas de filósofos, cientistas cognitivos e praticantes, por este não ser uma panaceia para todos problemas de inteligência artificial, nem uma explicação suficiente para a cognição humana.

Os modelos conexionistas precisam de um grande número de exemplos para treinamento. Os seres humanos, pelo contrário, podem aprender com poucos eventos, ou até mesmo um único. Além disso, os modelos conexionistas são na maioria das vezes uma caixa

preta, resultando em erros inesperados. Portanto, desistir definitivamente da abordagem simbólica é como jogar fora uma grande ferramenta para construção de inteligências artificiais. Muitas pessoas começaram a perceber que as abordagens simbólica e conexionista não são concorrentes, mas complementares.

Portanto, diante dessas incertezas, é indispensável um retorno às questões mais fundamentais do processo cognitivo, a fim de criar uma base sólida para a qual poderemos nos referir durante o processo de desenvolvimento de novas técnicas e aplicações que almejem a reprodução de processos cognitivos em ambientes artificiais.

Este trabalho gira entorno de um desses problemas fundamentais, a saber, o problema do embasamento de símbolos. Neste trabalho foi apresentado a origem e o que é realmente este problema e indicado onde estamos com relação as soluções propostas para resolvê-lo.

Para isto, foi feita uma breve revisão que apresenta trabalhos importantes que tentaram lidar com o problema de embasamento de símbolos, focado nas aplicações deste em robótica e aplicações de sistemas inteligentes. O campo é claramente muito ativo e muitos artigos têm sido publicados nos últimos anos. Esta é uma consequência das tendências de integração de robôs e sistemas distribuídos em ambientes não estruturados e dinâmicos. Tais ambientes exigem um tratamento flexível do conhecimento e a conexão de informações simbólicas e sensoriais para poder operar com sucesso. Além disso, temos aplicações onde os seres humanos têm um papel ativo estão se tornando mais comum e problema do embasamento de símbolos é fundamental para elaboração de um agente inteligente que possa utilizar linguagens naturais da maneira mais parecida com seres humanos.

Para mostrar a aplicabilidade das propostas para solução do problema do embasamento de símbolos, foi feita uma simples implementação de um agente inteligente inspirado em conceitos vindos de uma das abordagens para resolução do problema, no caso, uma abordagem híbrida, que procura combinar um agente que utiliza raciocínios simbólicos em conjunto com elementos conexionistas.

Esta implementação teve como inspiração as propostas feitas por Harnad, ou seja, ainda mantendo como base para a cognição o uso de representações, mas utilizando também os *insights* das redes conexionistas para ancorar as representações computacionais à referentes sensoriais.

No entanto, esta implementação se limita a demonstrar a possível aplicabilidade dessa proposta híbrida em um contexto real. As propostas de Harnad possuem limitações ainda em seu âmbito teórico, como mostrado neste trabalho. Além disso, a própria implementação apresentada neste trabalho é extremamente limitada pois apenas demonstra a interação da abordagem simbólica com a conexionista. As representações utilizadas neste

trabalho foram embasadas previamente pelo programador, algo que já fere qualquer tentativa séria de resolução do problema do embasamento simbólico, pois um agente deve ser capaz de operar de forma autônoma e, portanto, deveria ser capaz de criar suas novas representações por conta própria e ser capaz de dotar tais representações com um referente com base de suas próprias percepções.

4.5 Trabalhos Futuros

Então, para onde vamos daqui? Claramente ainda há muito a aprender e o problema do embasamento de símbolos ainda está longe de ser resolvido de forma a ter consentimento de toda comunidade científica. O problema, portanto, continua aberto e as tentativas para resolvê-lo ainda podem trazer muitos benefícios tanto para a área da Inteligência Artificial, quanto para o entendimento das próprias habilidades cognitivas do ser humano.

Por se tratar de uma primeira versão de implementação de uma abordagem híbrida, este trabalho é extremamente limitado e por isso proporciona várias sugestões para trabalhos futuros.

Como mostrado na seção de fundamentação teórica, existem diversas soluções propostas para o problema do embasamento de símbolos, porém, ao ser realizada a pesquisa para este trabalho, é possível perceber que apesar de um grande número de soluções propostas em termos teóricos, existem poucos trabalhos que implementam estas soluções para analisar sua performance e sua viabilidade. Este trabalho tentou abordar o problema adotando uma abordagem representacionalista, como defendida por Harnad. Outros trabalhos ainda podem trabalhar este mesmo problema sob uma abordagem semi-representacionalista ou não representacionalista.

Devido ao limite de tempo para implementação deste trabalho, o agente utilizado para integração dos sistemas multi-contexto com o problema de embasamento de símbolos é um agente muito básico, o foco do trabalho foi maior na parte diretamente relacionado com o problema do embasamento de símbolos em si. Portanto, trabalhos futuros podem utilizar deste trabalho para explorar em maior profundidade as características deliberativas de um agente multi-contexto simbolicamente embasado. Dessa forma, a recomendação é de encontrar uma aplicação prática mais elaborada para testar o agente, utilizando as crenças aprendidas por meio do embasamento de símbolos para ativar intenções mais elaboradas de um agente.

O escopo deste trabalho se limitou a uma pequena demonstração de futuras possibilidades, portanto a rede neural responsável por lidar com os estímulos sensoriais já estava treinada para lidar com as imagens que à ela são apresentadas. Um dos grandes problemas a ser resolvido por qualquer trabalho que pretenda resolver o problema de em-

basamento de símbolos, por meio de uma abordagem representacionista, é o problema no caso de um símbolo que não tenha um referente ancorado previamente por meio de um agente externo. O sistema deve ser capaz de associar um símbolo CACHORRO ao referente cachorro de uma rede neural de forma autônoma pelo próprio agente. Portanto, trabalhos futuros podem explorar a viabilidade da utilização de outros tipos de redes neurais ou ainda a utilização de algoritmos de aprendizagem não-supervisionada como forma de lidar com tais casos.

Neste trabalho, o problema gira em torno de imagens de números, mas trabalhos futuros devem lidar com casos de um agente posto no mundo real, ou seja, casos que o agente entre em contato com as formas mais variadas de objetos. Então, por exemplo, se não temos um símbolo LÂMPADA e nem uma rede neural que saiba distinguir dentre os objetos uma lâmpada, um agente que pretenda ter resolvido o problema de embasamento de símbolos deve conseguir criar uma nova categoria simbólica com um novo conjunto de percepções vindas da rede neural. Essa dificuldade retorna para um nível teórico, pois este é justamente o problema da abordagens representacionistas de Harnad que foi criticada por Floridi. Portanto, nesse aspecto ainda existem dificuldades teóricas que devem ser resolvidas para enfrentar os problemas práticos.

Uma possível direção para levar o problema do embasamento de símbolos é no caso de um sistema com múltiplos agentes. Nesse caso, o trabalho deverá levar em consideração algumas questões como a performance do processo de embasamento e a viabilidade para se integrar os múltiplos agentes. Pode não ser vantajoso que cada agente tenha uma rede neural para fazer o embasamento de símbolos pois um ambiente com muitos agentes exigiria muito do sistema, acarretando em problemas de performance. Levando isso em consideração, pode ser levantada uma questão de cunho até mesmo filosófico para se analisar se a rede neural deveria fazer parte dos agentes ou do ambiente no qual os agentes estão inseridos. Com relação à viabilidade desses múltiplos agentes estarem integrados está o problema de agentes diferentes terem um referente diferente para o mesmo símbolo. Em termos práticos, isso poderia acarretar em problemas na comunicação entre os agentes. Em termos teóricos, tal comportamento se mostraria incoerente com a experiência que nós, humanos, temos como seres capazes de possuir símbolos embasados: podemos nos comunicar mantendo a expectativa que quando estamos falando de um objeto, o ouvinte possui uma imagem mental semelhante a nossa. Devido a esse problema, seria interessante que trabalhos explorassem o jogo da adivinhação (como descrito na descrição de soluções para o problema do embasamento de símbolos na fundamentação teórica) para o desenvolvimento de um léxico comum entre os múltiplos agentes.

Em termos estritamente teóricos, Floridi criticou as tentativas passadas de resolução do problema, apontando já seus defeitos conceituais. Porém, nem mesmo a estratégia proposta de Floridi convenceu a todos, tendo ela própria sido vítima de críticas. Dessa

forma, trabalhos futuros, tanto da área da computação quanto da filosofia e ciências cognitivas, ainda podem trabalhar em uma formulação em nível teórico para o problema do embasamento simbólico.

Referências

- ARNHOLD, V. L. H. *Implementação de um framework para o desenvolvimento de agentes como sistema mult-contexto*. Monografia (TCC) — Universidade Federal de Santa Catarina, 2018. Citado 9 vezes nas páginas [66](#), [67](#), [68](#), [69](#), [70](#), [71](#), [76](#), [77](#) e [89](#).
- BIELECKA, K. Why taddeo and floridi did not solve the symbol grounding problem. *Journal of Experimental Theoretical Artificial Intelligence*, v. 27, 01 2015. Citado na página [52](#).
- BISHOP, C. M. *Pattern Recognition and Machine Learning*. 1st ed. 2006. corr. 2nd printing. ed. [S.l.]: Springer, 2006. (Information science and statistics). ISBN 9780387310732,0387310738. Citado na página [22](#).
- BORDINI, R.; HÜBNER, J. Bdi agent programming in agentspeak using jason. In: . [S.l.: s.n.], 2006. p. 143–164. Citado 3 vezes nas páginas [67](#), [68](#) e [69](#).
- BRATMAN, M. E. Intention, plans and practical reason. *Bibliovault OAI Repository, the University of Chicago Press*, v. 100, 01 1987. Citado na página [26](#).
- BREAZEAL, C. L. Sociable machines : expressive social exchange between humans and robots. 05 2000. Citado na página [59](#).
- BREAZEAL, C. L. *Designing Sociable Robots*. [S.l.]: MIT Press, 2002. (Intelligent Robotics and Autonomous Agents). ISBN 0262025108,9780262025102. Citado 2 vezes nas páginas [59](#) e [60](#).
- BREWKA, G.; EITER, T. Equilibria in heterogeneous nonmonotonic multi-context systems. In: . [S.l.: s.n.], 2007. v. 1, p. 385–390. Citado na página [27](#).
- BRINGSJORD, S. The symbol grounding problem ... remains unsolved. *Journal of Experimental Theoretical Artificial Intelligence*, v. 27, 01 2015. Citado na página [52](#).
- BRINGSJORD, S.; GOVINDARAJULU, N. S. Artificial intelligence. In: ZALTA, E. N. (Ed.). *The Stanford Encyclopedia of Philosophy*. Fall 2018. [S.l.]: Metaphysics Research Lab, Stanford University, 2018. Citado 2 vezes nas páginas [17](#) e [20](#).
- BROOKS, R. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, v. 2, n. 1, p. 14–23, March 1986. ISSN 0882-4967. Citado 4 vezes nas páginas [37](#), [55](#), [56](#) e [58](#).
- BROOKS, R. A. Elephants don't play chess. *Robotics and Autonomous Systems*, Elsevier Science, v. 6, 1990. Citado 3 vezes nas páginas [44](#), [53](#) e [54](#).
- BROOKS, R. A. Intelligence without representation. *Artificial Intelligence*, Elsevier Science, v. 47, 1991. Citado 3 vezes nas páginas [44](#), [53](#) e [54](#).
- CANGELOSI, A.; GRECO, A.; HARNAD, S. From robotic toil to symbolic theft: Grounding transfer from entry-level to higher-level categories1. *Connection Science*, Taylor and Francis Group, v. 12, 06 2000. Citado 3 vezes nas páginas [37](#), [40](#) e [41](#).

- CANGELOSI, A.; GRECO, A.; HARNAD, S. *Symbol Grounding and the Symbolic Theft Hypothesis*. [S.l.]: Springer, London, 2002. Citado na página 42.
- CARVALHO, F. G. d. *Comportamento em Grupo de Personagens do Tipo BlackWhite*. Dissertação (Mestrado) — PUCRJ, 2004. Citado 2 vezes nas páginas 26 e 27.
- COLE, D. The chinese room argument. In: ZALTA, E. N. (Ed.). *The Stanford Encyclopedia of Philosophy*. Spring 2019. [S.l.]: Metaphysics Research Lab, Stanford University, 2019. Citado na página 32.
- DASTANI, M. et al. A programming language for cognitive agents goal directed 3apl. In: . [S.l.: s.n.], 2003. v. 3067, p. 111–130. Citado 2 vezes nas páginas 68 e 69.
- DAVIDSSON, P. Toward a general solution to the symbol grounding problem: Combining machine learning and computer vision. *Machine Learning and Computer Vision*, 06 1994. Citado 3 vezes nas páginas 37, 42 e 43.
- FLORIDI, L. *The Philosophy of Information*. [S.l.]: Oxford University Press, 2011. ISBN 0199232385,9780199232383. Citado 16 vezes nas páginas 37, 39, 40, 42, 44, 46, 47, 48, 49, 50, 51, 52, 58, 61, 81 e 82.
- FODOR, J. A. *The Language of Thought*. [S.l.]: Thomas Y. Crowell, 1975. (The Language thought series). ISBN 0690008023,9780690008029. Citado na página 13.
- FRANKLIN, S.; GRAESSER, A. Is it an agent, or just a program?: A taxonomy for autonomous agents. In: . [S.l.: s.n.], 1996. p. 21–35. Citado 2 vezes nas páginas 23 e 24.
- FREGE, G. Sense and reference. *Philosophical Review*, Duke University Press on Behalf of Philosophical Review, v. 57, n. 3, p. 209–230, 1948. Citado na página 34.
- FREGE, G. Begriffsschrift. a formula language of pure thought modelled on that of arithmetic. In: BEANEY, M. (Ed.). *The Frege Reader*. [S.l.]: Blackwell, 1997. Citado na página 33.
- GARSON, J. Connectionism. In: ZALTA, E. N. (Ed.). *The Stanford Encyclopedia of Philosophy*. Fall 2018. [S.l.]: Metaphysics Research Lab, Stanford University, 2018. Citado na página 21.
- GELAIM, T. *Modelo de agentes e-BDI integrando confianc a baseado em siste-mas multi-contexto*. Monografia (Mestrado) — Universidade Federal de Santa Catarina, 2016. Citado na página 69.
- GELAIM, T. et al. Sigon: A multi-context system framework for intelligent agents. *Expert Systems with Applications*, v. 119, 10 2018. Citado 8 vezes nas páginas 63, 66, 67, 68, 69, 70, 76 e 89.
- GIUNCHIGLIA, F.; SERAFINI, L. Multilanguage hierarchical logics (or: How we can do without modal logics). 03 1994. Citado na página 28.
- GONÇALVES, R.; KNORR, M.; LEITE, J. Evolving multi-context systems. *Frontiers in Artificial Intelligence and Applications*, v. 263, p. 375–380, 12 2014. Citado 2 vezes nas páginas 27 e 28.

- HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 2. ed. [S.l.]: Morgan Kaufmann, 2006. (The Morgan Kaufmann Series in Data Management Systems). ISBN 9781558609013,1558609016. Citado na página 22.
- HARNAD, S. Categorical perception: The groundwork of cognition. *The Journal of the Acoustical Society of America*, American Institute of Physics, v. 86, 1989. Citado 2 vezes nas páginas 79 e 80.
- HARNAD, S. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, Elsevier Science, v. 42, 1990. Citado 14 vezes nas páginas 14, 32, 34, 35, 37, 38, 40, 41, 42, 44, 47, 78, 80 e 82.
- HARNAD, S. *Computation Is Just Interpretable Symbol Manipulation: Cognition Isn't*. 1994. 379–390 p. Citado 2 vezes nas páginas 32 e 33.
- HARNAD, S. Symbol grounding problem. *Scholarpedia*, v. 2, n. 7, p. 2373, 2007. Revision #73220. Citado na página 32.
- HARNAD, S.; DAMPER, R. I. Neural network models of categorical perception. *Attention, Perception, Psychophysics*, Psychonomic Society Publications, v. 62, 01 2000. Citado 2 vezes nas páginas 79 e 80.
- HARNAD, S.; HANSON, S.; LUBIN, J. Categorical perception and the evolution of supervised learning in neural nets. In: . [S.l.: s.n.], 1991. Citado na página 81.
- HAYES-ROTH, B. An architecture for adaptive intelligent systems. *Artificial Intelligence*, v. 72, p. 329–365, 01 1995. Citado na página 24.
- HINTON, G. E. How neural networks learn from experience. *Scientific American*, Nature Publishing Group, v. 267, 1992. Citado na página 22.
- MAO, J. et al. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2019. Citado na página 78.
- MARCUS, G. Deep learning: A critical appraisal. *CoRR*, abs/1801.00631, 2018. Citado na página 78.
- MCCARTHY, J.; HAYES, P. J. Some philosophical problems from the standpoint of artificial intelligence. In: MELTZER, B.; MICHIE, D. (Ed.). *Machine Intelligence 4*. [S.l.]: Edinburgh University Press, 1969. p. 463–502. Reprinted in McC90. Citado 2 vezes nas páginas 18 e 19.
- NEWELL, A.; SIMON, H. A. Computer science as empirical inquiry: symbols and search. *Communications of the ACM*, Association for Computing Machinery, v. 19, 3 1976. Citado 5 vezes nas páginas 13, 20, 35, 36 e 44.
- NORVIG, P.; RUSSELL, S. *Inteligência Artificial*. 3rd. ed. [S.l.]: Elsevier, 2013. ISBN 978-8535237016. Citado 8 vezes nas páginas 13, 14, 17, 18, 19, 20, 23 e 25.
- PARSONS, S. et al. Agent specification using multi-context systems. In: . [S.l.: s.n.], 2002. v. 2403, p. 205–226. Citado na página 29.

- PEIRCE, C. S. *Collected Papers of Charles Sanders Peirce*. [S.l.]: Thoemmes Continuum, 1994. (Past Masters). ISBN 9781855065567,1855065568,9119001901191. Citado 3 vezes nas páginas 34, 44 e 45.
- PFEIFER, C. S. R. *Understanding Intelligence*. [S.l.]: The MIT Press, 1999. ISBN 0262161818,9780262161817. Citado na página 44.
- PUTNAM, H. Psychological predicates. In: CAPITAN, W. H.; MERRILL, D. D. (Ed.). *Art, Mind, and Religion*. [S.l.]: University of Pittsburgh Press, 1967. p. 37–48. Citado na página 13.
- RAO, A.; GEORGEFF, M. Bdi agents: From theory to practice. 11 2000. Citado na página 26.
- RESCORLA, M. The computational theory of mind. In: ZALTA, E. N. (Ed.). *The Stanford Encyclopedia of Philosophy*. Spring 2017. [S.l.]: Metaphysics Research Lab, Stanford University, 2017. Citado 2 vezes nas páginas 13 e 35.
- RUSSELL, S. J. Rationality and intelligence. *Artificial Intelligence*, Elsevier Science, v. 94, 1997. Citado na página 19.
- SATHYA, R.; ABRAHAM, A. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, v. 2, 02 2013. Citado na página 22.
- SEARLE, J. *Minds Brains and Science*. [S.l.]: Harvard University Press, 1986. ISBN 0674576330,9780674576339. Citado na página 32.
- SEARLE, J. R. Minds, brains, and programs. *Behavioral and Brain Sciences*, Cambridge University Press, v. 3, 9 1980. Citado 5 vezes nas páginas 13, 19, 30, 31 e 36.
- SHOHAM, Y. Agent-oriented programming. *Artif. Intell.*, Elsevier Science Publishers Ltd., Essex, UK, v. 60, n. 1, p. 51–92, mar. 1993. ISSN 0004-3702. Citado na página 24.
- STEELS, L. The emergence and evolution of linguistic structure: From lexical to grammatical communication systems. *Connect. Sci.*, v. 17, p. 213–230, 09 2005. Citado na página 52.
- TURING, A. M. Computing machinery and intelligence. *Mind*, Oxford University Press, v. 59, 10 1950. Citado na página 17.
- VARSHAVSKAYA, P. *Behavior-Based Early Language Development on a Humanoid Robot*. [S.l.]: Lund University Cognitive Studies, 2002. 149–158 p. Citado 3 vezes nas páginas 37, 59 e 61.
- VOGT, P. The physical symbol grounding problem. *Cognitive Systems Research*, v. 3, p. 429–457, 12 2001. Citado 5 vezes nas páginas 37, 42, 44, 45 e 46.
- WEISS, G. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. New edition. [S.l.]: The MIT Press, 2000. ISBN 0262731312,9780262731317. Citado 2 vezes nas páginas 27 e 28.
- WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, v. 10, p. 115–152, 1995. Citado 2 vezes nas páginas 25 e 26.

Apêndices

APÊNDICE A – ARTIGO

Problema de embasamento de símbolos em um sistema multi-contexto

Leonardo Vailatti Eichstaedt¹

¹Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brazil

leonardo.eichstaedt@grad.ufsc.br

***Abstract.** The symbol grounding problem is related to the problem of how symbols gain their meanings. Consequently, this problem is related to the problem of how consciousness is capable of creating a relationship between a system of formal symbols and their referents. How meaningless symbols, manipulated solely on the basis of their shapes, can be grounded on something other than (other) meaningless symbols. After an introduction to the philosophical problem and the proposed solutions, this work applies a strategy suggested by Harnad to a BDI agent seen as a multi-context system, defined with the Sigon language. The work shows how the basing of symbols of an agent to the perceptions of the environment can be an application of the strategies for solving the problem of the symbol grounding problem in a context of creation of multi-context agents. But within these limitations of the approach used lies the core of the philosophical problem of meaning acquisition that remains open to be explored by future works..*

***Resumo.** O problema de embasamento de símbolos (symbol grounding problem) está relacionado ao problema de como símbolos ganham seus significados. Consequentemente, este problema está relacionado com o problema de como a consciência é capaz de criar uma relação entre um sistema de símbolos formais e seus referentes. Como símbolos sem sentido, manipulados unicamente com base em suas formas, podem ser embasados em algo além do que outros símbolos sem sentido. Após uma introdução ao problema e às soluções propostas, este trabalho aplica uma estratégia sugerida por Harnad a um agente BDI visto como um sistema multi-contexto, definido pela linguagem Sigon. O trabalho mostra como o embasamento de símbolos de um agente às percepções do ambiente pode ser uma aplicação das estratégias para a resolução do problema de embasamento de símbolos em um contexto de criação de agentes multi-contexto. Porém, dentro dessas limitações da abordagem utilizada está o*

núcleo do problema filosófico de aquisição de significados que continua em aberto para ser explorado por trabalhos futuros.

1. Introdução

Em 1990, Stevan Harnad formula o “*Symbol Grounding Problem*” (daqui em diante chamado de problema do embasamento simbólico) para questionar como é feita a transposição desse abismo entre forma e significado de um símbolo em nossas mentes e como podemos reproduzir tal capacidade mental em um computador, ou seja, como embasar um símbolo a um significado que não seja apenas outro símbolo sem significado [Harnad, 1990]. Podemos entender então o problema do embasamento simbólico como um problema relacionado à como palavras (símbolos) adquirem significado.

O presente trabalho busca investigar o problema do embasamento simbólico, seus antecedentes e desenvolvimentos, e também demonstrar que tal problema não é apenas de caráter filosófico e abstrato mas que possui consequências práticas no desenvolvimento de inteligências artificiais.

O desenvolvimento de inteligências artificiais neste trabalho se dará por meio de agentes inteligentes. Estes correspondem a uma visão de desenvolvimento de inteligências artificiais, muito associada a Peter Norvig e Stuart Russell, que podem ser caracterizados por sua ênfase na capacidade de interação com o ambiente em que estão e sua autonomia para tomar a melhor decisão esperada visto a sua base de conhecimento.

Para demonstrar tal aplicabilidade, será proposto e avaliado um protótipo de um modelo para embasamento simbólico que será utilizado em um *framework* de sistema multi-contexto para agentes inteligentes, ou seja, agentes cujas capacidades são expressas em linguagens lógicas diferentes. Espera-se que por meio de símbolos embasados seja possível utilizá-los como componente de regras ponte, permitindo que módulos do agente especificados em diferentes lógicas possam se comunicar mais facilmente por meio destes símbolos que possuem significados definidos.

Este artigo é organizado da seguinte forma: Na seção 2 é feita uma fundamentação teórica dos tópicos tratados, na seção 3 são mostrados os trabalhos correlatos, na seção 4 é mostrado o desenvolvimento e a seção 5 apresenta a conclusão e os trabalhos futuros.

2. Fundamentação teórica

2.1. Problema do embasamento de símbolos

O problema de embasamento (*grounding*) simbólico está relacionado ao problema de como símbolos ganham seus significados. Este problema está relacionado, conseqüentemente, com o problema da consciência, ou melhor, o problema de como a consciência é capaz de criar uma relação entre um sistema de símbolos formais e seus referentes [HARNAD, 2007].

Harnad (1990) define este problema é colocado na forma da seguinte pergunta:

“Como a interpretação semântica de um sistema formal de símbolos pode ser intrínseca ao sistema, em vez de apenas parasitar os significados em nossas cabeças? Como os significados dos símbolos sem sentido, manipulados unicamente com base em suas formas (arbitrárias), podem ser fundamentados em algo além de outros símbolos sem sentido?” [Harnad, 1990]

Tomando o significado de um símbolo como “o meio de escolher o seu referente”, então é necessário que a entidade que contém essa palavra tenha a capacidade de seguir as regras para atribuir um referente ao símbolo [HARNAD, 1990]. No caso do ser humano, a nossa mente possui tal capacidade, apesar de não sabermos exatamente como ela faz isso.

Aqui dois aspectos são importantes: primeiro, os símbolos não representam nada para um sistema, pelo menos não o que eles “dizem” designar. Somente alguém operando o sistema pode reconhecer esses símbolos como referindo-se a entidades fora do sistema. Segundo, o sistema de símbolos não pode conter seu fechamento ao relacionar símbolos apenas com outros símbolos; alguma outra coisa deve ser necessária para estabelecer uma conexão entre os símbolos e o que eles representam.

O que entendemos por “embasamento” (*grounding*), então, é a capacidade que uma entidade possui de escolher referentes para seus símbolos [HARNAD, 1990]. Embasamento é uma função de entrada / saída. O embasamento conecta as entradas sensoriais de objetos externos a símbolos e estados internos que ocorrem dentro de um sistema sensorio-motor autônomo, orientando o processamento e a saída resultante do sistema.

A busca por processos de embasamento de símbolos diz respeito à compreensão de processos que possibilitem a conexão dessas representações puramente simbólicas com o que elas representam de fato, o que poderia ser diretamente, ou por meio de outras representações fundamentadas.

2.2. Solução proposta por Harnad

No mesmo artigo em que Harnad define o problema do embasamento de símbolos, ele também oferece uma possível solução para o problema. Essa proposta estabeleceu o padrão para todas as estratégias posteriores.

A estratégia sugerida por Harnad (1990) é baseada em um modelo híbrido, que implementa uma mistura de características de um sistema simbólico e de um sistema conexionista. Com isso, Harnad tenta superar os limites típicos encontrados pelos sistemas simbólicos e conexionistas puros.

Por um lado, em um modelo puramente simbólico faltam as conexões entre os símbolos e seus referentes. Um sistema simbólico, embora passível de interpretação sintática sistemática, não está embasado.

E embora sistemas conexionistas tornem possível conectar símbolos e referentes usando os dados perceptivos e os recursos das categorias representacionais, eles não podem manipular símbolos (como os sistemas de símbolos podem fazer facilmente) para produzir uma interpretação intrínseca, sistemática e única dos mesmos.

Isso justifica a solução proposta por Harnad, que, devido à sua natureza semi-simbólica pretende oferecer o melhor dos dois mundos.

Além dessa justificativa computacional, Harnad baseia sua estratégia em uma teoria psicológica que assume a habilidade de construir categorias do mundo como base para a linguagem e cognição. Segundo ele:

“Nós já sabemos o que os seres humanos são capazes de fazer. Eles podem (1) discriminar, (2) manipular, (3) identificar e (4) descrever os objetos, eventos e estados de coisas no mundo em que vivem, e eles também podem (5) ‘produzir descrições’ e (6) ‘responder a descrições’ desses objetos, eventos e estados de coisas.” [HARNAD, 1990]

De acordo com a proposta de Harnad, os símbolos manipulados por um agente podem ser embasados conectando-os aos dados perceptivos que eles denotam. A conexão é estabelecida por um processo de categorização dos sinais sensorio-motores. Com base nas capacidades humanas já mencionadas, Harnad propõe que os símbolos devem ser fundamentados em três etapas:

1. Iconização: o processo de transformação de padrões de dados sensoriais em representações icônicas (equivalentes análogos internos das projeções de objetos sobre as superfícies sensoriais do agente).
2. Discriminação: o processo de julgar se duas entradas são as mesmas ou, se são diferentes, o quanto elas diferem.

3. Identificação: o processo de atribuir uma resposta única - isto é, um nome - a uma classe de inputs, tratando-os como equivalentes ou invariantes em algum aspecto.

Os dois primeiros estágios produzem representações sub-simbólicas, enquanto o terceiro estágio embasa os símbolos.

As representações icônicas em (1) são obtidas a partir do conjunto de todas as experiências relacionadas às percepções do mesmo tipo de objeto. As representações categóricas são então obtidas através do processo de discriminação em (2). Essa delimitação de categorias seria feita por meio de uma análise de grau de congruência entre as representações icônicas. Uma vez elaboradas, as representações categóricas são associadas em (3) com classes de símbolos (os nomes), provendo assim estes com referências apropriadas que os fundamentam.

Iconização e discriminação são subprocessos, realizados por meio de redes neurais. Elas tornam possível a associação de um nome com uma classe de entrada e posteriormente, a nomeação dos referentes. No entanto, por si só, as redes neurais são incapazes de produzir representações simbólicas, então elas ainda não podem permitir que o agente desenvolva capacidades simbólicas. A fim de contornar essa falha, Harnad fornece ao seu modelo híbrido um sistema simbólico, que pode manipular símbolos sintaticamente e, graças às categorias produzidas pelo sistema conexionista, finalmente consegue atingir um embasamento semântico para seus símbolos.

O problema dessa abordagem para Floridi (2011) diz respeito ao modo como o sistema híbrido deve encontrar as características invariantes de suas projeções sensoriais que permitem categorizar e identificar objetos corretamente.

Para ilustrar a deficiência da solução apresentada por Harnad, Floridi (2011) propõe que imaginemos, por exemplo, um agente que implemente o modelo híbrido, chamado PERC. Inicialmente, PERC não possui conteúdo ou recursos semânticos, portanto, não possui compromisso semântico. Assumindo que PERC está equipado com uma maneira de adquirir alguma entrada de dados, uma câmera de vídeo, através da qual observa seu ambiente externo. Seguindo Harnad, suponha que, por meio de sua câmera e redes neurais, PERC é capaz de produzir algumas representações icônicas a partir dos dados perceptivos que recolhe do ambiente. PERC deveria então desenvolver representações categóricas destes dados perceptivos, considerando apenas as características invariantes das representações icônicas [FLORIDI, 2011].

Em seguida, deveriam ser organizadas as representações categóricas em categorias conceituais como, por exemplo, “animal quadrúpede”. A pergunta a ser feita é: de onde as categorias conceituais, como “animal quadrúpede”, se originam?

Redes neurais podem ser usadas para encontrar estruturas (se existirem) nos dados, como padrões de dados. No entanto, se forem supervisionadas então elas são treinadas por meio de um conjunto de treinamento pré-selecionado e feedback repetido, então qualquer embasamento que eles possam fornecer é inteiramente extrínseco.

Se elas são treinadas sem supervisão, as redes implementam algoritmos de treinamento que não usam o dados de saída, mas dependem apenas dos dados de entrada para tentar encontrar estruturas no espaço de dados de entrada. Unidades na mesma camada competem entre si para serem ativadas. No entanto, elas ainda precisam ter vieses integrados e detectores de recursos internos para alcançar a saída desejada. Tais recursos são necessariamente codificados por um supervisor, de acordo com critérios estabelecidos [FLORIDI, 2011].

Além disso, uma vez que redes não supervisionadas foram treinados, ainda precisam ter sua saída verificada para ver se os resultados fazem algum sentido em relação ao espaço de dados de entrada. Este processo de validação é realizada externamente por um supervisor. Então, neste caso também, seja qual for o embasamento que eles podem fornecer ainda é totalmente extrínseco [FLORIDI, 2011].

No exemplo, “animal quadrúpede”, como categoria não é o resultado do embasamento intrínseco do PERC porque este já deve ter tido ajuda semântica para alcançar essa conclusão. A estratégia posta em Harnad (1990), na verdade, pressupõe a disponibilidade de recursos semânticos que se espera que o agente seja capaz de desenvolver do zero, através de suas interações com o ambiente e outros agentes e das elaborações de seus dados perceptivos.

3. Trabalhos relacionados

Em Gelaim (2018) é apresentada a linguagem Sigon. Construída com o propósito de operacionalizar agentes como sistemas multi-contexto. Com base nisso, Sigon permite a exibição na criação e relacionamento entre componentes do agente.

A modularidade é uma característica fundamental no desenvolvimento de arquiteturas de agentes multi-contexto. Vários módulos podem ser definidos para expressar propriedades específicas, como crenças, desejos e intenções. O fator chave para o relacionamento entre os componentes do agente é a definição correta das regras de ponte, que descrevem a relação entre os contextos.

A troca de conhecimento entre contextos é feita de forma declarativa por meio de regras de ponte. A estrutura de regras de ponte é composta pelo contexto que irá adicionar uma crença (cabeça) com base em regras de outros contextos (corpo).

O trabalho de Arnhold (2018) expande o trabalho apresentado por Gelaim (2018), com o objetivo de apresentar a criação de um *framework* para permitir o desenvolvimento e criação de agentes BDI vistos como um sistema multi-contexto.

O trabalho de Gelaim (2018) apresentou um modelo para de nição de agentes como um sistema multi-contexto, ele provê a teoria, a arquitetura e a linguagem, elementos necessários para a implementação do *framework*. O trabalho de Arnhold (2018) utiliza este modelo como base e núcleo para implementação de um *framework* que permite o desenvolvimento de agentes BDI sobre um sistema multi-contexto.

Esses trabalhos servirão como base para o desenvolvimento do agente utilizado neste trabalho, utilizando a linguagem Sigon, apresentada por Gelaim (2018), para projetar um agente como sistema multi-contexto e utilizando o *framework*, proposto por Arnhold (2018), para implementar o agente.

4. Desenvolvimento

4.1. Proposta

Como já explicado, o problema do embasamento simbólico é o problema de como símbolos ganham seus signi cados.

O embasamento é uma função de entrada/saída. As entradas sensoriais de objetos externos são conectadas à símbolos internos de uma mente por meio desse embasamento.

Em termos práticos para o campo da inteligência arti cial, esse problema se manifesta na incapacidade de um agente computacional simular a capacidade cognitiva humana de associar um estímulo sensorial visual ao representante simbólico interno desta mesma entrada. Então, por exemplo, se uma pessoa receber como estímulo visual uma imagem de um número escrito em um pedaço de papel, as capacidades cognitivas humanas permitem que a pessoa entenda que aquilo que ela percebe como estímulo visual representa algo equivalente a um símbolo presente internamente na mente dela, que pode ser manipulado no contexto do sistema simbólico a qual pertence.

No caso de um agente computacional dotado da capacidade sensorial visual, por outro lado, ao receber o estímulo visual do número, ele não consegue associar esse dado visual a algum símbolo que o representa. Portanto, mesmo que o sistema simbólico do agente computacional contenha um símbolo que seja equivalente ao dado sensorial que lhe está sendo apresentado, o agente não consegue fazer a conexão de que esse dado sensorial de entrada é equivalente ao símbolo contido internamente no agente.

No caso deste trabalho, estamos lidando com esse mesmo problema no contexto de agentes multi-contexto, ou seja, sistemas que consistem em um conjunto de módulos, chamados contextos, que permitem que cada um destes possam ser considerados como

uma lógica e um conjunto de fórmulas escritas nessa lógica, e um conjunto de regras de ponte para transferir informações entre contextos.

Unidades dos sistemas multi-contexto representam os vários componentes da arquitetura. Eles contêm a maior parte do conhecimento de solução de problemas de um agente, e esse conhecimento é codificado na teoria específica que a unidade encapsula. A natureza das unidades varia entre as arquiteturas.

Neste trabalho, o sistema será modelado na forma de um agente BDI. Portanto, teremos unidades que representam crenças, desejos e intenções, onde a lógica associada a cada unidade fornece a linguagem na qual as informações nesta unidade são codificadas e as regras de ponte fornecem o mecanismo pelo qual as informações são transferidas entre as unidades.

Para a construção de agentes BDI como sistemas multi-contexto, será utilizada a linguagem Sigon. O *framework* utilizado para implementação do agente na linguagem Sigon é o trabalho de Arnhold (2018).

Com base nesse modelo, é definido uma arquitetura de agente BDI:

$$Ag_{BDI} = \langle \{CC, CB, CD, CI, CP\}, \Delta_{rp} \rangle$$

Onde CC, CB, CD, CI, CP são, respectivamente, contextos de comunicação, crenças, desejos, intenções e planejamento.

As regras de ponte, são responsáveis pela troca de conhecimento entre os contextos do sistema multi-contexto. Para facilitar o desenvolvimento de agentes BDI, um conjunto de regras que representam o comportamento BDI são implementadas pelo trabalho de Arnhold (2018), com o objetivo de permitir a execução do cenário proposto.

Assim sendo, como objetivo deste trabalho, gostaríamos que o Contexto de Comunicação (responsável por lidar com os sensores e atuadores) do agente fosse capaz de perceber algum dado visual no ambiente e que com base no que foi percebido, associar essa percepção a alguma representação simbólica da percepção sensorial. Uma vez que a forma simbólica fosse adicionada as bases de conhecimento do agente, seria possível utilizar essa forma simbólica para que o agente operasse normalmente, realizando seus planos conforme este símbolo que foi percebido por meio de estímulos sensoriais no ambiente.

$$\frac{CC : \text{sense}(\varphi)}{CB : \varphi}$$

Quando um objeto visual for identificado no ambiente pelos sensores pertencentes ao Contexto de Comunicação, essa regra de ponte deve ser ativada,

colocando no contexto de crenças (CB) um símbolo que represente aquilo que foi percebido no ambiente.

Dessa forma, apesar de ser de uma maneira limitada, estaremos simulando o comportamento de um agente que consegue formar conexões entre percepções não-simbólicas do ambiente com equivalentes simbólicos internos do agente.

4.2. Implementação

Neste trabalho, uma rede neural é implementada para identificar números manuscritos fornecida por um conjunto de dados chamado MNIST. O banco de dados MNIST¹ é um grande banco de dados de dígitos manuscritos que é comumente usado para treinamento de vários sistemas de processamento de imagens.

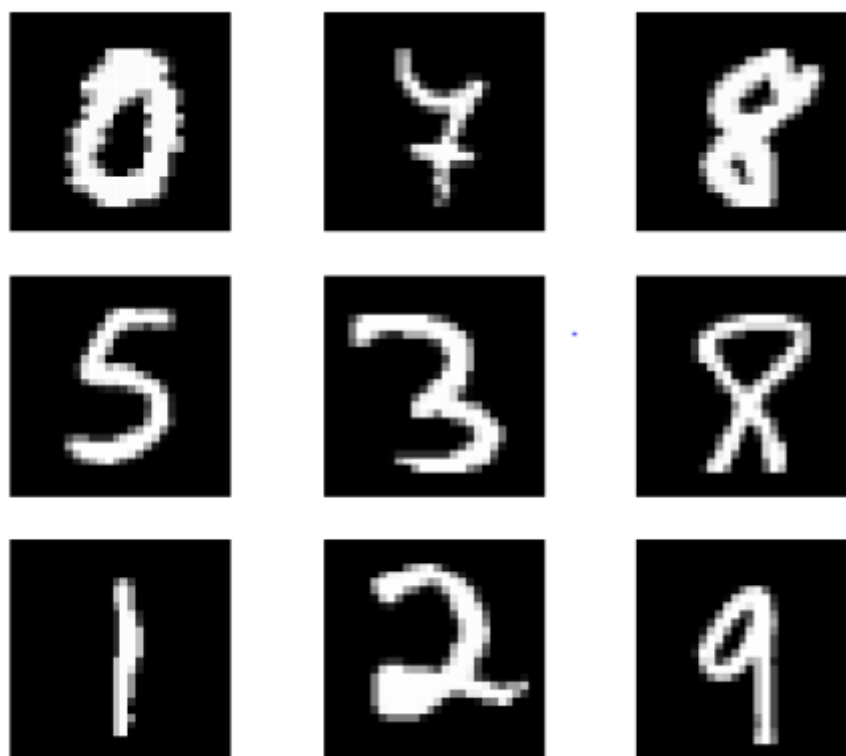


Figure 1. Exemplo de imagens do MNIST.

As imagens em preto e branco do NIST foram normalizadas para caber em uma caixa delimitadora de 28 x 28 pixels e anti-aliased, que introduziu níveis de escala de cinza. O banco de dados MNIST contém 60.000 imagens de treinamento e 10.000 imagens de teste.

Neste trabalho, uma rede neural convolucional foi treinada pela identificação de imagens em MNIST banco de dados digital manuscrito para prever exatamente o que os

¹<http://yann.lecun.com/exdb/mnist/>

números na imagem são. O reconhecimento dos dígitos consiste em avaliar a imagem e classificá-la em um dos 10 dígitos possíveis (0 a 9).

Para este trabalho, foi implementada uma rede neural simples, com base no código do site Neural Networks and Deep Learning². O autor do site implementa uma rede neural na linguagem Python, mas para facilitar a integração da rede neural com o agente deste trabalho, a rede foi implementada em Java, a mesma linguagem que é utilizada para implementar o agente.

Como o reconhecimento de dígitos é uma tarefa relativamente fácil, podemos utilizar uma rede neural de três camadas.

A camada de entrada da rede contém neurônios que codificam os valores dos pixels de entrada. Como nossos dados de treinamento para a rede consistirão em muitas imagens de dígitos manuscritos digitalizados de 28 por 28 pixels, assim a camada de entrada contém $784 = 28 \times 28$ neurônios.

A segunda camada da rede é uma camada oculta. Denotamos o número de neurônios nessa camada oculta por n , e vamos experimentar valores diferentes para n . O valor final que foi utilizado foi o valor de $n = 30$, dando um resultado de até cerca de 93% de categorizações no processo de testes.

A camada de saída da rede contém 10 neurônios. São numerados os neurônios de saída de 0 a 9, e descobrimos qual neurônio tem o maior valor de ativação. Se o primeiro neurônio disparar, isso indicará que a rede acha que o dígito é 0. Se o segundo neurônio disparar, isso indicará que a rede acha que o dígito é 1. E assim por diante para os outros neurônios de saída.

²<https://github.com/mnielsen/neural-networks-and-deep-learning>

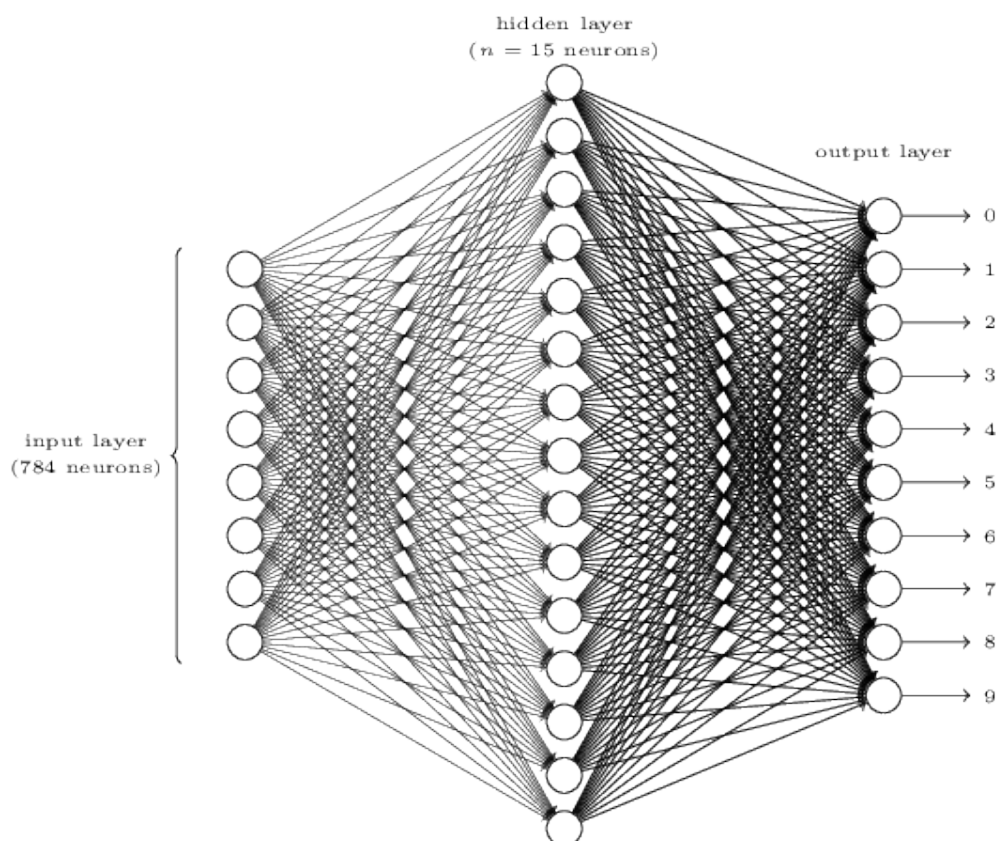


Figure 2. Representação da rede neural.

Agora que temos uma arquitetura para nossa rede neural, como ele pode aprender a reconhecer dígitos? Aprendizagem com exemplos é de longe o tipo de aprendizado mais estudado em IA e pode ser visto como um paralelo ao aprendizado por ser explicitamente ensinado. Nesse tipo de aprendizado, o supervisor induz uma descrição de categoria a partir de exemplos pré-classificados e contra-exemplos da categoria fornecidos por algum tipo de professor.

Como há um agente externo presente para guiar o processo de aprendizagem, esse tipo de aprendizado é uma instância de aprendizagem supervisionada. Assim, é o supervisor que decide quando criar um novo conceito.

A Rede Neural Artificial com múltiplas camadas ocultas possui uma excelente capacidade de aprendizado de características. As características obtidas da aprendizagem têm uma descrição essencial aos dados, facilitando a visualização ou classificação.

Uma vez que a rede neural foi satisfatoriamente treinada, podemos mandar imagens para ela fazer sua função de classificação.

Como no contexto desse trabalho o tipo de imagens que serão usadas são as de dígitos escritos a mão, foi feito um programa que permite que o usuário desenhe os números com o mouse, de forma a facilitar a entrada de dados.

Uma janela é aberta com um Canvas que permite a interação com o usuário:

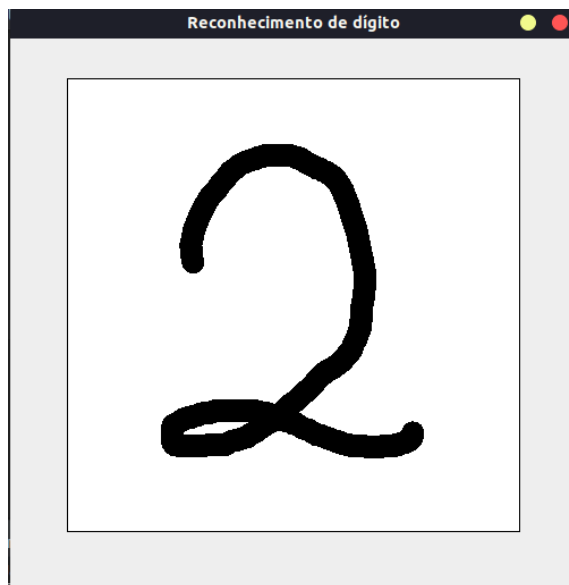


Figure 3. Desenho no Canvas.

A imagem passa por um processo de normalização para se adequar ao padrão das imagens utilizadas para treinamento, ou seja, as imagens do conjunto MNIST.

```
public static BufferedImage normalizedImg(BufferedImage image) {  
    BufferedImage normalizedImg = ImageUtils.deepCopy(image);  
    normalizedImg = ImageUtils.cropWhiteBorder(normalizedImg);  
    normalizedImg = ImageUtils.invertColor(normalizedImg);  
    normalizedImg = ImageUtils.fitInto(normalizedImg, 20, 20,  
        Color.BLACK.getRGB());  
    normalizedImg = ImageUtils.centerWithMassInto(normalizedImg, 28, 28,  
        Color.BLACK.getRGB());  
    return normalizedImg;  
}
```

Uma vez que a imagem foi devidamente normalizada, ela está apta a ser interpretada pela rede neural. O resultado da execução da rede neural é um array cujos elementos representam um valor numérico que representa o quanto a imagem se adequou a cada categoria. Portanto, temos um array de tamanho 10, onde a posição de índice 0 vai possuir um número que representa o quão similar a imagem é a categoria que representa as imagens de número 0, a posição de índice 1 vai possuir um número

que representa o quão similar a imagem é a categoria que representa as imagens de número 1, e assim por diante.

```
0: 0.00995168842976391
1: 9.202541024456156E-4
2: 0.9903609499321034
3: 0.021492221298937363
4: 7.591433464261339E-5
5: 1.223957134607156E-6
6: 6.070924720381273E-4
7: 4.4969447119153086E-5
8: 5.432917389535998E-4
9: 5.0717567417180736E-8
```

Figure 4. Output da rede neural ao perceber um número.

Por fim foi implementado um agente na linguagem Sigon, que permite a especificação de um agente como sistema multi-contexto. Seguindo as definições da linguagem de Gelaim (2018) e o *framework* de implementação de Arnhold (2018), iniciamos com a definição do agente em um arquivo “.on” seguindo as especificações da linguagem.

```
communication:
  sensor("visual", "sg.Camera").
  actuator("acharNumero", "sg.Recognize").

beliefs:

desires:

intentions:
  terNumero(X).

planner:
  plan(
    terNumero(X),
    [
      action(acharNumero(X))
    ], _, [numero(X)]).
```

A comunicação do agente com seu ambiente ou com outros agentes é feita pelo contexto de comunicação por meio de sensores e atuadores, não acoplados a uma estrutura de ambiente. Dessa forma, teremos um sensor que irá perceber o ambiente e um atuador que irá procurar um número. No sensor, o argumento “visual” representa a identificação e o argumento “sg.Camera” representa a implementação do sensor. O mesmo acontece no caso do atuador, “acharNumero” é a identificação e “sg.Recognize” representa a implementação do atuador.

A função começa executando uma função que é responsável por abrir um Canvas para o usuário desenhar o número manualmente, como já foi explicado anteriormente. O número desenhado será processado pela rede neural e por fim retornará uma String que corresponde a qual número foi identificado pela rede neural. O valor obtido então é recebido pelo sensor.

Aqui regra de ponte responsável por copiar as percepções do contexto de comunicação é ativada e é posto no contexto de crenças (CB).

$$\frac{CC : \textit{sense}(\varphi)}{CB : \varphi}$$

Dessa forma, é aqui que quando um objeto visual for identificado no ambiente pelos sensores pertencentes ao Contexto de Comunicação, a rede neural irá fazer a conexão entre a representação visual e uma representação simbólica. A regra de ponte será ativada, e a representação simbólica do que foi identificado no ambiente será colocado no contexto de crenças.

4.3. Resultados

Como já mostrado anteriormente, a rede neural é capaz de identificar corretamente o número desenhado pelo usuário.

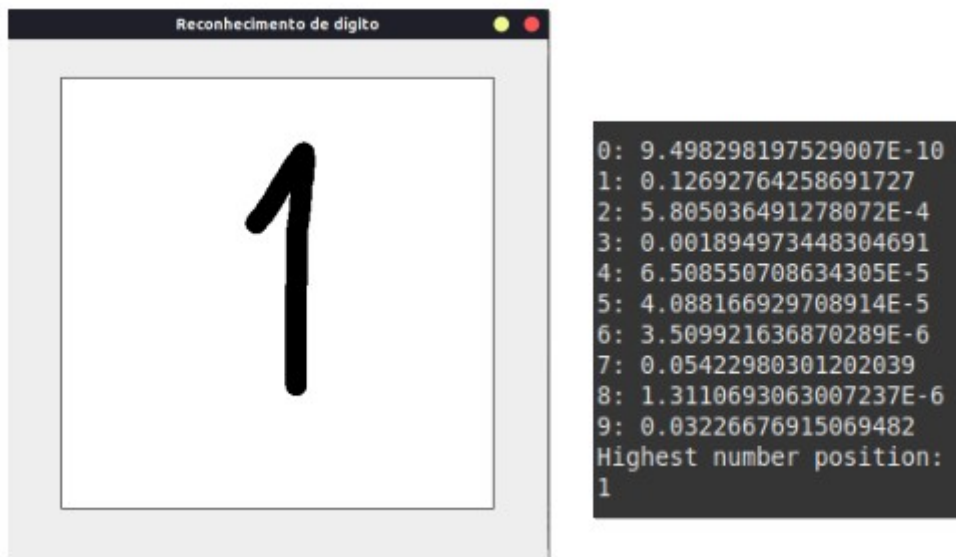


Figure 5. Número no canvas e resultado na rede neural.

Isso possibilita realizarmos a conexão entre a percepção sensorial e o símbolo que equivale à percepção.

No caso do agente, ao ser iniciado, o agente recebe uma percepção de que está sendo executado, e portanto, esta é a única crença presente no agente até o momento.

```

CB running.

CD
CI terNumero(X).
CC sense(running).

```

Figure 6. Agente antes de perceber número.

Porém, como o agente possui a intenção “terNumero(X).”, o plano de nido pelo agente será executado, e a ação “acharNumero(X)” será executado pelo atuador.

O Canvas será aberto para o usuário desenhar um número e a rede irá fazer a categorização da imagem e o sensor irá detectar que há um certo número no ambiente.

```

CC sense(numero(2)).

```

Figure 7. Agente percebe número.

Portanto a regra de ponte será ativada, e o contexto de crenças será atualizado.

```
CB running.  
  
numero(2).  
  
CD  
CI terNumero(X).  
CC sense(numero(2)).
```

Figure 8. Agente após perceber número.

Dessa forma o banco de conhecimentos do agente agora possui a informação de que uma percepção sensorial do número 2 foi encontrada no ambiente. Caso o agente tivesse algum outro plano que dependesse de existir um número no ambiente, ele seria agora executado.

Portanto, de uma percepção visual não-simbólica temos agora um símbolo que equivalente e poderíamos realizar operações sobre os símbolos normalmente.

Com isso conseguimos uma função que transforma o dado de entrada em uma representação simbólica.

5. Conclusão

Para termos uma teoria cognitiva coerente, devemos estabelecer como é feita a conexão entre sobre o quê a mente pensa e o quê é percebido no mundo real. Do ponto de vista da Inteligência Artificial, essa é a questão da capacidade de se conceber um sistema artificial capaz de inventar e usar símbolos fundamentados em suas interações com o mundo externo e com outros agentes. Esta é uma das questões mais fundamentais da cognição, a saber, o problema de embasamento de símbolos. Vários pesquisadores, mais notavelmente Searle, argumentaram que esse problema nunca será resolvido em sistemas artificiais porque esses sistemas são apenas manipuladores sintáticos, estes sistemas não compartilham de propriedades essenciais presentes na mente humana que permite tal habilidade cognitiva.

É indispensável um retorno às questões mais fundamentais do processo cognitivo, a fim de criar uma base sólida para a qual poderemos nos referir durante o processo de desenvolvimento de novas técnicas e aplicações que almejem a reprodução de processos cognitivos em ambientes artificiais.

Este trabalho gira entorno de um desses problemas fundamentais, a saber, o problema do embasamento de símbolos. Neste trabalho foi apresentado a origem e o que é realmente este problema e indicado onde estamos com relação as soluções propostas para resolvê-lo.

Para mostrar a aplicabilidade das propostas para solução do problema do embasamento de símbolos, foi feita uma simples implementação de um agente inteligente inspirado em conceitos vindos de uma das abordagens para resolução do problema, no caso, uma abordagem híbrida, que procura combinar um agente que utiliza raciocínios simbólicos em conjunto com elementos conexionistas.

Esta implementação teve como inspiração as propostas feitas por Harnad, ou seja, ainda mantendo como base para a cognição o uso de representações, mas utilizando também os *insights* das redes conexionistas para ancorar as representações computacionais à referentes sensoriais.

No entanto, esta implementação se limita a demonstrar a possível aplicabilidade dessa proposta híbrida em um contexto real. As propostas de Harnad possuem limitações ainda em seu âmbito teórico, como mostrado neste trabalho. Além disso, a própria implementação apresentada neste trabalho é extremamente limitada pois apenas demonstra a interação da abordagem simbólica com a conexionista. As representações utilizadas neste trabalho foram embasadas previamente pelo programador, algo que já fere qualquer tentativa séria de resolução do problema do embasamento simbólico, pois um agente deve ser capaz de operar de forma autônoma e, portanto, deveria ser capaz de criar suas novas representações por conta própria e ser capaz de dotar tais representações com um referente com base de suas próprias percepções.

5.1. Trabalhos futuros

Por se tratar de uma primeira versão de implementação de uma abordagem híbrida, este trabalho é extremamente limitado e por isso proporciona várias sugestões para trabalhos futuros.

Devido ao limite de tempo para implementação deste trabalho, o agente utilizado para integração dos sistemas multi-contexto com o problema de embasamento de símbolos é um agente muito básico, o foco do trabalho foi maior na parte diretamente relacionado com o problema do embasamento de símbolos em si. Portanto, trabalhos futuros podem utilizar deste trabalho para explorar em maior profundidade as características deliberativas de um agente multi-contexto simbolicamente embasado. Dessa forma, a recomendação é de encontrar uma aplicação prática mais elaborada para testar o agente, utilizando as crenças aprendidas por meio do embasamento de símbolos para ativar intenções mais elaboradas de um agente.

O escopo deste trabalho se limitou a uma pequena demonstração de futuras possibilidades, portanto a rede neural responsável por lidar com os estímulos sensoriais já estava treinada para lidar com as imagens que à ela são apresentadas. Um dos grandes problemas a ser resolvido por qualquer trabalho que pretenda resolver o problema de embasamento de símbolos, por meio de uma abordagem representacionista, é o

problema no caso de um símbolo que não tenha um referente ancorado previamente por meio de um agente externo. O sistema deve ser capaz de associar um símbolo CACHORRO ao referente cachorro de uma rede neural de forma autônoma pelo próprio agente. Portanto, trabalhos futuros podem explorar a viabilidade da utilização de outros tipos de redes neurais ou ainda a utilização de algoritmos de aprendizagem não-supervisionada como forma de lidar com tais casos.

Neste trabalho, o problema gira em torno de imagens de números, mas trabalhos futuros devem lidar com casos de um agente posto no mundo real, ou seja, casos que o agente entre em contato com as formas mais variadas de objetos. Então, por exemplo, se não temos um símbolo LÂMPADA e nem uma rede neural que saiba distinguir dentre os objetos uma lâmpada, um agente que pretenda ter resolvido o problema de embasamento de símbolos deve conseguir criar uma nova categoria simbólica com um novo conjunto de percepções vindas da rede neural. Essa dificuldade retorna para um nível teórico, pois este é justamente o problema da abordagem representacionalista de Harnad que foi criticada por Floridi. Portanto, nesse aspecto ainda existem dificuldades teóricas que devem ser resolvidas para enfrentar os problemas práticos.

Em termos estritamente teóricos, Floridi criticou as tentativas passadas de resolução do problema, apontando já seus defeitos conceituais. Porém, nem mesmo a estratégia proposta de Floridi convenceu a todos, tendo ela própria sido vítima de críticas. Dessa forma, trabalhos futuros, tanto da área da computação quanto da filosofia e das ciências cognitivas, ainda podem trabalhar em uma formulação em nível teórico para o problema do embasamento simbólico.

7. Referências

Arnhold, V. L. H. Implementação de um framework para o desenvolvimento de agentes como sistema multi-contexto. Monografia (TCC) — Universidade Federal de Santa Catarina, 2018.

Floridi, L. *The Philosophy of Information*. [S.l.]: Oxford University Press, 2011. ISBN 0199232385,978019923238.

Gelaim, T. et al. Sigon: A multi-context system framework for intelligent agents. *Expert Systems with Applications*, v. 119, 10 2018.

Harnad, S. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, Elsevier Science, v. 42, 1990.

Harnad, S. Symbol grounding problem. *Scholarpedia*, v. 2, n. 7, p. 2373, 2007. Revision #73220.

Norvig, P.; Russel, S. Inteligência Artificial . 3rd. ed. [S.l.]: Elsevier, 2013. ISBN 978-8535237016.

APÊNDICE B – CÓDIGO FONTE

B.1 Reconhecedor de Dígitos

Implementação do Frame

```
package frame;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.image.BufferedImage;
import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

import neural.Neural;
import neurolib.utils.MathHelper;
import utils.CharacterCanvas;
import utils.ImageUtils;

@SuppressWarnings("serial")
public class Frame extends JPanel implements CharacterCanvas.DrawListener,
    ActionListener {

    public static ArrayList<Double> result = new ArrayList<>();
    public static int maxNum = 0;

    JFrame frame;
    CharacterCanvas characterCanvas;
```

```
public Frame() {
    frame = new JFrame("Reconhecimento de dgito");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(500, 515);
    frame.setResizable(false);
    frame.setVisible(true);
    frame.add(this);
    setLayout(null);
    init();
}

private void init() {

    characterCanvas = new CharacterCanvas();
    characterCanvas.setVisible(true);
    characterCanvas.setSize(new Dimension(400, 400));
    characterCanvas.setLocation(50, 35);
    characterCanvas.setDrawListener(this);
    add(characterCanvas);

    for (int i = 0; i < 10; i++) {
        result.add(0.0);
    }
}

@Override
public void paint(Graphics g) {
    super.paint(g);
}

@Override
public void onDrawEnd() {
    ArrayList<Double> input = new ArrayList<>();

    BufferedImage image = characterCanvas.getImg();

    image = ImageUtils.normalizedImg(image);

    int w = image.getWidth();
    int h = image.getHeight();
    for (int y = 0; y < h; y++) {
```

```
        for (int x = 0; x < w; x++) {
            int pixelValue = new Color(image.getRGB(x, y)).getRed();
            input.add(MathHelper.round(pixelValue / 255.0D, 2));
        }
    }

    ArrayList<Double> result = Neural.run(input);

    if (result != null) {
        Frame.result.clear();
        Frame.result.addAll(result);

        int i = 0;
        double maxPercent = 0;
        for (Double res : Frame.result) {
            System.out.println(i + ": " + res);
            if(maxPercent < res){
                maxPercent = res;
                maxNum = i;
            }
            i++;
        }
        System.out.println("Highest number: \n" + maxNum);
    }
    repaint();
    frame.dispatchEvent(new WindowEvent(frame, WindowEvent.WINDOW_CLOSING));
}

@Override
public void onDrawStart() {

}

@Override
public void actionPerformed(ActionEvent arg0) {
}

}
```

Implementação dos utils

CharacterCanvas

```
package utils;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionListener;
import java.awt.image.BufferedImage;

import javax.swing.JPanel;

@SuppressWarnings("serial")
public class CharacterCanvas extends JPanel {

    private float pencilSize = 20;
    private BufferedImage img;
    private Graphics2D imgG2;
    private int prevMouseX = -1;
    private int prevMouseY = -1;
    private DrawListener listener = null;

    public CharacterCanvas() {
        setBackground(Color.WHITE);
        addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                prevMouseX = e.getX();
                prevMouseY = e.getY();
                if (listener != null) {
                    listener.onDrawStart();
                }
            }

            @Override
            public void mouseReleased(MouseEvent e) {
                if (listener != null) {
                    listener.onDrawEnd();
                }
            }
        });
    }
}
```

```
    }
});
addMouseMotionListener(new MouseMotionListener() {
    @Override
    public void mouseDragged(MouseEvent e) {
        handleMouseDrag(e);
    }

    @Override
    public void mouseMoved(MouseEvent e) {
    }
});
}

private void handleMouseDrag(MouseEvent e) {
    int mouseX = e.getX();
    int mouseY = e.getY();
    imgG2.drawLine(prevMouseX, prevMouseY, mouseX, mouseY);
    prevMouseX = mouseX;
    prevMouseY = mouseY;
    repaint();
}

public BufferedImage getImg() {
    return img;
}

public void setDrawListener(DrawListener listener) {
    this.listener = listener;
}

@Override
public void setSize(Dimension size) {
    super.setSize(size);
    img = new BufferedImage(size.width, size.height,
        BufferedImage.TYPE_INT_ARGB);
    imgG2 = img.createGraphics();
    imgG2.setColor(Color.WHITE); // Background color
    imgG2.fillRect(0, 0, size.width, size.height);
    imgG2.setColor(Color.BLACK); // Drawing color
    imgG2.setStroke(new BasicStroke(pencilSize, BasicStroke.CAP_ROUND,
        BasicStroke.JOIN_ROUND));
}
```

```
}

@Override
public void paint(Graphics g) {
    super.paint(g);
    Graphics2D g2 = (Graphics2D) g;
    g2.setColor(Color.BLACK);
    g2.drawImage(img, 0, 0, getWidth(), getHeight(), null);
    g2.drawRect(0, 0, getWidth() - 1, getHeight() - 1);
}

public interface DrawListener {
    void onDrawEnd();

    void onDrawStart();
}
}



---


ImageUtils

package utils;

import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.image.BufferedImage;

public class ImageUtils {

    public static BufferedImage normalizedImg(BufferedImage image) {
        BufferedImage normalizedImg = ImageUtils.deepCopy(image);
        normalizedImg = ImageUtils.cropBorder(normalizedImg);
        normalizedImg = ImageUtils.invertColor(normalizedImg);
        normalizedImg = ImageUtils.fitInto(normalizedImg, 20, 20,
            Color.BLACK.getRGB());
        normalizedImg = ImageUtils.centerImage(normalizedImg, 28, 28,
            Color.BLACK.getRGB());
        return normalizedImg;
    }

    private static BufferedImage deepCopy(BufferedImage image) {
        return new BufferedImage(image.getColorModel(), image.copyData(null),
```

```
        image.isAlphaPremultiplied(), null);
    }

    private static BufferedImage cropBorder(BufferedImage source) {
        int upperBound = -1, lowerBound = -1, leftBound = -1, rightBound = -1;

        upper: for (int y = 0; y < source.getHeight(); y++) {
            for (int x = 0; x < source.getWidth(); x++) {
                if (source.getRGB(x, y) != Color.white.getRGB()) {
                    upperBound = y;
                    break upper;
                }
            }
        }

        lower: for (int y = source.getHeight() - 1; y >= 0; y--) {
            for (int x = 0; x < source.getWidth(); x++) {
                if (source.getRGB(x, y) != Color.white.getRGB()) {
                    lowerBound = y;
                    break lower;
                }
            }
        }

        left: for (int x = 0; x < source.getWidth(); x++) {
            for (int y = 0; y < source.getHeight(); y++) {
                if (source.getRGB(x, y) != Color.white.getRGB()) {
                    leftBound = x;
                    break left;
                }
            }
        }

        right: for (int x = source.getWidth() - 1; x >= 0; x--) {
            for (int y = 0; y < source.getHeight(); y++) {
                if (source.getRGB(x, y) != Color.white.getRGB()) {
                    rightBound = x;
                    break right;
                }
            }
        }
    }
}
```

```
        return source.getSubimage(leftBound, upperBound, rightBound - leftBound
            + 1, lowerBound - upperBound + 1);
    }

    private static BufferedImage invertColor(BufferedImage source) {
        for (int x = 0; x < source.getWidth(); x++) {
            for (int y = 0; y < source.getHeight(); y++) {
                Color pixelColor = new Color(source.getRGB(x, y), true);
                pixelColor = new Color(255 - pixelColor.getRed(), 255 -
                    pixelColor.getGreen(),
                    255 - pixelColor.getBlue());
                source.setRGB(x, y, pixelColor.getRGB());
            }
        }
        return source;
    }

    private static BufferedImage fitInto(BufferedImage source, int width, int
        height, int backgroundColor) {
        float sourceAspectRatio = source.getWidth() / source.getHeight();
        float resultAspectRatio = (float) width / height;
        float newSourceWidth, newSourceHeight;

        if (resultAspectRatio > sourceAspectRatio) {
            newSourceHeight = height;
            newSourceWidth = (newSourceHeight / source.getHeight()) *
                source.getWidth();
        } else {
            newSourceWidth = width;
            newSourceHeight = (newSourceWidth / source.getWidth()) *
                source.getHeight();
        }

        Image scaledSrc = source.getScaledInstance((int) newSourceWidth, (int)
            newSourceHeight, Image.SCALE_SMOOTH);

        BufferedImage result = new BufferedImage(width, height,
            BufferedImage.TYPE_INT_ARGB);
        Graphics2D g2 = result.createGraphics();
        g2.setColor(new Color(backgroundColor));
        g2.fillRect(0, 0, result.getWidth(), result.getHeight());
        g2.drawImage(scaledSrc, (int) (width / 2 - newSourceWidth / 2), (int)
```

```
        (height / 2 - newSourceHeight / 2), null);
    g2.dispose();
    return result;
}

private static BufferedImage centerImage(BufferedImage source, int width,
    int height, int backgroundColor) {
    BufferedImage result = new BufferedImage(width, height,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g2 = result.createGraphics();
    g2.setColor(new Color(backgroundColor));
    g2.fillRect(0, 0, result.getWidth(), result.getHeight());
    int[] centerOfImage = computeCenter(source);
    g2.drawImage(source, width / 2 - centerOfImage[0], height / 2 -
        centerOfImage[1], null);
    g2.dispose();
    return result;
}

private static int[] computeCenter(BufferedImage img) {
    long xSum = 0;
    long ySum = 0;
    long num = 0;

    for (int x = 0; x < img.getWidth(); x++) {
        for (int y = 0; y < img.getHeight(); y++) {
            int weight = new Color(img.getRGB(x, y)).getRed();
            xSum += x * weight;
            ySum += y * weight;
            num += weight;
        }
    }
    return new int[] { (int) ((double) xSum / num), (int) ((double) ySum /
        num) };
}
}
```

Implementação da ligação entre frame e NeuroLib

```
package neural;
```

```
import java.io.IOException;
```

```
import java.util.ArrayList;

import neurolib.neural.Network;

public class Neural {
    public static ArrayList<Double> run(ArrayList<Double> test) {
        Network testNet = null;
        try {
            testNet = Network.load("network", "digitreco");
        } catch (IOException e) {
            e.printStackTrace();
        }

        if (testNet == null) {
            return null;
        }

        ArrayList<Double> results = testNet.run(test);

        return results;
    }

    public static ArrayList<Double> ArrayToArrayList(double[] array) {
        ArrayList<Double> out = new ArrayList<>();
        for (double d : array) {
            out.add(d);
        }
        return out;
    }
}
```

B.2 Agente

Especificação do agente

```
communication:
sensor("visual", "sg.Camera").
actuator("acharNumero", "sg.Recognize").
```

```
beliefs:
```

```
desires:
```

```
intentions:
```

```
    terNumero(X).
```

```
planner:
```

```
    plan(  
        terNumero(X),  
        [  
            action(acharNumero(X))  
        ], _, [numero(X)]).
```

Implementação sensor

```
package sg;
```

```
import br.ufsc.ine.agent.context.communication.Sensor;
```

```
import rx.subjects.PublishSubject;
```

```
public class Camera extends Sensor{
```

```
    public static final PublishSubject<String> msg = PublishSubject.create();
```

```
    public void run() {
```

```
        msg.subscribe(super.publisher);
```

```
    }
```

```
}
```

Implementação atuador

```
package sg;
```

```
import java.util.List;
```

```
import br.ufsc.ine.agent.context.communication.Actuator;
```

```
import sg.Main;
```

```
public class Recognize extends Actuator{
```

```
    @Override
```

```
    public void act(List<String> args) {
```

```
        Main.acharNumero();
```

```
    }
```



```
};
```

Implementação do ambiente

```
package sg;

import java.util.List;
import java.util.stream.Stream;

import javax.swing.JOptionPane;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

import org.antlr.v4.runtime.CharStream;
import org.antlr.v4.runtime.CharStreams;
import org.antlr.v4.runtime.CommonTokenStream;
import org.antlr.v4.runtime.tree.ParseTree;
import org.antlr.v4.runtime.tree.ParseTreeWalker;

import agent.AgentLexer;
import agent.AgentParser;
import br.ufsc.ine.agent.Agent;
import br.ufsc.ine.agent.context.beliefs.BeliefsContextService;
import br.ufsc.ine.agent.context.communication.CommunicationContextService;
import br.ufsc.ine.agent.context.desires.DesiresContextService;
import br.ufsc.ine.agent.context.intentions.IntentionsContextService;
import br.ufsc.ine.agent.context.plans.PlansContextService;
import br.ufsc.ine.parser.AgentWalker;
import br.ufsc.ine.parser.VerboseListener;

public class Main{

    public static void main(String[] args) {
        startAgent();
        Camera.msg.onNext("running.");
    }

    public static void acharNumero() {
```

```
String number = ImageInput.launch();
Camera.msg.onNext("numero(" + number + ").");
}

private static void startAgent(){
    try {
        File agentFile = new
            File("/home/leonardo/sigon-lang-experimental/examples/sg.on");
        CharStream stream =
            CharStreams.fromFileName(agentFile.getAbsolutePath());
        AgentLexer lexer = new AgentLexer(stream);
        CommonTokenStream tokens = new CommonTokenStream(lexer);

        AgentParser parser = new AgentParser(tokens);
        parser.removeErrorListeners();
        parser.addErrorListener(new VerboseListener());

        ParseTree tree = parser.agent();
        ParseTreeWalker walker = new ParseTreeWalker();

        AgentWalker agentWalker = new AgentWalker();
        walker.walk(agentWalker, tree);

        Agent agent = new Agent();
        agent.run(agentWalker);
        System.out.println(BeliefsContextService.getInstance().getTheory());
        System.out.println(DesiresContextService.getInstance().getTheory());
        System.out.println(IntentionsContextService.getInstance().getTheory());
        System.out.println(PlansContextService.getInstance().getTheory());
        System.out.println(CommunicationContextService.getInstance().getTheory());

    } catch (IOException e) {
        System.out.println("I/O exception.");
    }
}
}
```
