Pontifícia Universidade Católica de São Paulo
PUC-SP

Ricardo Maciel Gazoni

Computadores eletrônicos como agentes semióticos autônomos

DOUTORADO EM TECNOLOGIAS DA INTELIGÊNCIA E DESIGN DIGITAL

São Paulo
2019

Pontifícia Universidade Católica de São Paulo
PUC-SP

Ricardo Maciel Gazoni

Computadores eletrônicos como agentes semióticos autônomos

DOUTORADO EM TECNOLOGIAS DA INTELIGÊNCIA E DESIGN DIGITAL

Tese apresentada à Banca Examinadora da Pontifícia Universidade Católica de São Paulo, como exigência parcial para obtenção do título de DOUTOR em Tecnologias da Inteligência e Design Digital, sob a orientação do Prof. Dr. Winfried Maximilian Nöth.

São Paulo
2019

BANCA EXAMINADORA

_____

_____

_____

_____

_____

Às aves gasosas:
Cristiane, Felipe e Marcelo Alves Gazoni.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) - Código de Financiamento 001 nº 88887.199157/2018.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 no. 88887.199157/2018.

Agradecimentos

- Professor Winfried Nöth;
- Professora Lucia Santaella;
- Professor Ítalo Vega;
- Professores Marcus Bastos, Sérgio Basbaum e Nelson Brissac Peixoto;
- Edna Conti;
- Doutores Guilherme Cestari e Jonathas Magalhães Pereira da Silva;
- Elisabete Waller e Júlio Osamu Yoshida;
- Ricardo Motz Lubachescki;
- Eugênio Ouriques de Carvalho;
- Amigos do TIDD, e dos grupos de estudo TransObjeto e Leituras Avançadas em Peirce.

“Do not block the way of inquiry.”
Charles Sanders Peirce (CP 1.135, 1899)

Resumo

O estudo examina o conceito de agência semiótica e seus componentes à luz da filosofia de Charles Sanders Peirce (1839-1914), e aplica esses conceitos aos computadores. Apresenta elementos básicos da moldura filosófica peirciana e os utiliza na análise de fenômenos semióticos, evidenciando a visão abrangente de Peirce sobre conceitos como mente e ação mental, e se estende na conceituação da comunicação e da vagueza intrínseca às linguagens naturais. Analisa, concomitantemente, o computador eletrônico que, como implementação de uma máquina de Turing universal, é um dos objetos mais versáteis já inventados; tal flexibilidade suscitou, desde cedo, conjeturas quanto à possibilidade de que a máquina apresentasse comportamento inteligente. O ferramental peirciano permite estender o conceito de ação semiótica ao computador eletrônico, e postular como ele pode atuar como agente semiótico autônomo. Uma estrutura conceitual para tal é proposta, permitindo conjeturar novas formas de uso do computador, como a possibilidade de programação através de um processo dialógico. Adicionalmente, permite levantar questionamentos que se estendem à ciência da computação, filosofia e inteligência artificial, abrindo caminho para continuidade da pesquisa em direções tais como ampliação da compreensão dos limites de decidibilidade e computabilidade, a expansão do conceito de agência semiótica autônoma para outros agentes, como automóveis e robôs, e a ampliação do conhecimento filosófico a respeito da ação semiótica e da inteligência.

Palavras-chaves: Semiótica computacional, agência semiótica autônoma, programação dialógica, Peirce.

Abstract

The paper examines the concept of semiotic agency in light of the philosophy of Charles Sanders Peirce (1839-1914) and applies Peirce's semiotics to the study of computers. It presents essential elements of Peirce's philosophical framework, such as sign, interpretation, communication, and vagueness, as it is inherent in any natural language, before it focuses on the concepts of semiotic agency as well as Peirce's broad concepts of mind and mental action. The author analyzes the electronic computer, an implementation of a universal Turing machine, one of the most versatile devices ever invented, which, early on, has raised conjectures as to the possibility of exhibiting intelligent behavior. The elements of Peirce's philosophy introduced by the study are used in the analysis of the electronic computer to argue that it can be interpreted as a semiotic agent. A conceptual framework is proposed to introduce new ways of using the computer, particularly in programming in a dialogic process. The study addresses fundamental questions of computer science, computer philosophy, and artificial intelligence, aiming at a broader understanding of the limits of mathematical decidability and computability. Furthermore, it aims at extending the concept of autonomous semiotic agency to nonhuman agents, such as robots and autonomous automobiles. The results of this study are relevant to computer philosophy as well as to research on semiotic agency and intelligence in general.

Keywords: Computational semiotics, autonomous semiotic agency, dialogical programming, Peirce

Sumário

Lista de ilustrações

Lista de tabelas

1 Introdução

O presente trabalho é o desdobramento da pesquisa iniciada em 2013 indagando, na ocasião, as razões para o frequente insucesso na programação de computadores, particularmente para fins comerciais e de negócio (cf. STANDISH GROUP, 2013), um problema conhecido há mais de meio século (NAUR; RANDELL, 1969; BUXTON; RANDELL, 1970) que ainda não encontrou resposta adequada. Alguns afirmam que tal resposta não virá tão cedo (BROOKS JR., 1986) por conta da enorme complexidade envolvida em projetos comerciais de grande magnitude (cf. BROOKS JR., 2009). A intenção da pesquisa era procurar na semiótica do filósofo americano Charles Sanders Peirce (1839-1914) elementos que possibilitassem a análise da programação de computadores.

O primeiro resultado foi a dissertação de mestrado (GAZONI, 2015b). Nela, a teoria peirciana é compreendida e se analisa literatura que tenta aplicá-la à programação de computadores. Analisou-se o livro de Tanaka-Ishii (2010) — das abordagens estudadas na ocasião, a que mais fundo foi na tentativa de aplicação da semiótica estritamente à programação de computadores. No livro, Tanaka-Ishii criou uma teoria semiótica própria, tentando conciliar as teorias de Peirce e do linguista e semioticista suíço Ferdinand de Saussure (1857-1913). O livro resultante carece do poder analítico e expressivo da teoria peirciana, à luz da qual a dissertação o abordou.

A constatação que primeiro deu impulso à ideia de uma nova maneira de interagir com computadores é a de que vagueza e indeterminação são propriedades intrínsecas da linguagem natural (NÖTH; SANTAELLA, 2011). A vagueza intrínseca possibilita, por exemplo, que novos termos sejam introduzidos na língua mesmo que os objetos a que se referem sejam apenas vagamente conhecidos. A introdução desses novos termos é uma etapa necessária para o aumento da determinação a respeito desses objetos vagamente definidos, num processo em que os signos crescem (cf. NÖTH, 2014). O reconhecimento da incerteza intrínseca da linguagem natural é crucial para a compreensão dos desafios encontrados no processo de programação de computadores, já que um programa de computador não aceita instruções vagas ou indeterminadas.

Sendo do interesse de Peirce o modo como se dá a cognição (SANTAELLA, 2012, p. 75), sua semiótica resulta em uma teoria sígnica do conhecimento. Seu caráter abrangente permite utilizá-la em diversos ramos da ciência da computação (cf. GAZONI, 2015a), inclusive na inteligência artificial, empreitada que encontra dificuldade na própria falta de compreensão e divulgação dos conceitos mais profundos da filosofia de Peirce. Donde a ideia de abordar o computador como um agente semiótico, de acordo com essa filosofia.

Contexto da pesquisa

Esta tese propõe uma abordagem para utilização de computadores que os leva em conta como agentes semióticos de acordo com a filosofia e a semiótica de Peirce. A programação de computadores, dentro dessa moldura, pode ser vista como um processo de aprendizagem, no qual os programadores, ao criar e executar programas, estão alterando os hábitos das máquinas.

Não se trata de uma proposta de método para desenvolvimento de sistemas no qual requisitos são levantados e registrados para servir de base para a programação. A intenção aqui é caracterizar agentes semióticos de acordo com a filosofia de Peirce, destacando os principais elementos, e, tomando os computadores eletrônicos como agentes semióticos, apontar novas formas de atuar sobre eles.

Também não se trata de uma proposta de definição de um novo tipo de interface homem-computador. O trabalho de Souza (2005, 2013) utiliza a semiótica para analisar os elementos que participam do mecanismo de interação entre homem e computador, resultados aproveitados no ramo de conhecimento conhecido como HCI – Human-Computer Interface, ou interface humano-computador.

Não é, também, uma proposta de abordagem para o estudo da inteligência artificial, ao menos como esse campo é conhecido hoje. Quando Turing (1950) sugeriu que os computadores digitais poderiam apresentar comportamento considerado inteligente, inaugurou, na prática, a inteligência artificial, um campo de pesquisa que passou por várias fases e se desdobra em diversas vertentes até os dias atuais. Tal diversidade se explica não somente pela ausência de consenso a respeito do que venha a ser um comportamento inteligente, ou a respeito de quais elementos podem compor o comportamento inteligente. Várias abordagens já foram tentadas, com resultados recorrentemente aquém do esperado (cf. DREYFUS, 1992). Nesse cenário, a filosofia de Peirce, caso estivesse à disposição dos pesquisadores de inteligência artificial, poderia ser mais uma teoria a ser explorada. De fato, o conceito peirciano de *abdução* chegou a inspirar algumas abordagens para inteligência artificial (cf. KAKAS; TONI; KOWALSKI, 1993) cuja pesquisa progride até os dias de hoje, mas talvez a falta de acesso à íntegra da teoria de Peirce sobre o assunto levou a uma aplicação que, na essência, não corresponde ao imaginado pelo filósofo.

Muitas abordagens para a inteligência artificial têm em comum o fato de principiarem por uma ideia do que seja comportamento inteligente para, em seguida, proporem uma implantação computadorizada de uma versão dessa ideia, um fenômeno identificado pelos críticos da inteligência artificial, como Dreyfus (1992), Searle (1984) e Quaresma (2018). Em particular, nossa abordagem difere da de Gudwin e Queiroz (2007): o próprio título (*Semiotics and Intelligent Systems Development*, *Semiótica e o desenvolvimento de sistemas inteligentes*) explicita a

intenção que os autores declaram (p. vii): o foco do livro é perquirir como a semiótica trabalha com técnicas de sistemas inteligentes para robustecê-los.

A intenção do presente trabalho não é apontar um caminho para a implementação, nas máquinas, de comportamento inteligente semelhante ao humano. A ideia é mapear alguns dos principais elementos que, de acordo com Peirce, formam a agência semiótica (humana ou não) e explorá-los para encontrar-lhes um uso.

Problema, questão e hipótese

Segundo Brooks Jr. (1986), o desenvolvimento de software não deveria esperar um ganho de produtividade de uma ordem de grandeza num futuro próximo. Seu artigo aponta como razões essenciais: complexidade, conformidade (adequação a necessidades e interfaces muitas vezes mutáveis), mutabilidade ("o software é uma entidade constantemente sujeita a pressões por mudança") e invisibilidade (dificuldade em representar visualmente o software); além disso elenca mudanças passadas que deram ganhos de produtividade e aponta as tecnologias que, então, traziam esperanças de melhoria. Entre elas, a inteligência artificial.

Por outro lado, a programação de computadores é baseada num modelo no qual o programador determina o que o computador fará. Essa determinação pode se dar de acordo com vários diferentes paradigmas, mas todos têm em comum a característica de serem baseados em elementos que, tomados individualmente, são inequívocos: cada comando a ser executado, ou cada declaração que especifica o resultado desejado do programa, define com rigor o que o programador quer. Esse modo de programar impossibilita a introdução da vagueza na programação. E, embora seja possível criar programas que, quando utilizados, permitem um certo grau de indeterminação na interação, essa vagueza está restrita à utilização do programa. A ideia de programação exclui a possibilidade de vagueza: como o computador poderia executar um programa que apresenta um comando vagamente definido? Tal execução pressupõe uma mudança de paradigma. Uma ideia proposta (cf. GAZONI, 2018) é que, diante de um comando vagamente definido, a reação mais sensata do executor não é tentar executar o que não está exatamente definido, mas tentar avançar no que julga ser a intenção do comandante, ao mesmo tempo em que interage com ele. Uma atuação dialógica, que pressupõe que o executante seja um agente com certa autonomia, que de certa forma apresente alguma inteligência, ainda que não necessariamente semelhante à humana.

As previsões de Brooks permanecem válidas até hoje (cf. STANDISH GROUP, 2013). Por outro lado, a semiótica e a filosofia de Peirce nunca foram utilizados a fundo na tentativa de determinar formas de programar o computador (cf. GAZONI, 2015b); menos ainda tomá-lo como agente semiótico. Diante disso, propõe-se aqui responder à questão: compreendendo os elementos básicos da agência semiótica à luz da filosofia de Peirce, é possível tomar o computador

como um agente semiótico e dessa forma propor novas formas para sua programação?

Objetivos

O objetivo geral do presente trabalho é determinar os elementos básicos da agência semiótica à luz da filosofia peirciana de modo que se possa compreender o computador como um agente semiótico. Com isso se espera, especificamente, propor uma forma de aplicação dos elementos da agência semiótica a computadores e, adicionalmente, apontar novas formas possíveis de atuação sobre computadores como agentes semióticos.

2 Elementos da filosofia peirciana

Este capítulo apresenta conceitos da filosofia peirciana necessários para a compreensão do argumento da tese. Duas considerações a respeito do texto que segue: primeiro, não houve preocupação em associar cada conceito ao todo da obra filosófica de Peirce; tampouco são feitas considerações metafísicas. Antes, procurou-se expor o necessário de forma a permitir que se reflita sobre a agência semiótica em computadores. Segundo, não é um capítulo sobre a obra de Peirce. Não há preocupação, por exemplo, em determinar a evolução cronológica dos conceitos, ou cotejar exaustivamente diferentes versões de cada tema.

A obra de Peirce vem sendo cada vez melhor compreendida. Portanto, é possível que alguns dos conceitos aqui apresentados evoluam no futuro. Começamos pelos mais básicos.

2.1 As categorias peircianas

O estudo das categorias peircianas é sedutor. São conceitos abstratos, aos quais Peirce retornou inúmeras vezes ao longo da vida, quer por razões didáticas, quer para esclarecer e ampliar a teoria. E o que são? Nas palavras de Peirce,

> Tento uma análise do mundo. Aquilo com que estamos lidando não é metafísica: é lógica, apenas. Portanto, não perguntamos o que realmente existe, apenas o que aparece a cada um de nós em todos os momentos de nossas vidas. Analiso a experiência, que é a resultante cognitiva de nossas vidas passadas, e nela encontro três elementos. Denomino-os *Categorias*.[1] (CP 2.84, 1902)[2]

As categorias ocupam uma posição central na filosofia de Peirce. Constituem os elementos com os quais ele constrói todos os seus mais importantes raciocínios. Peirce afirma que são universais — estão presentes em tudo — e são todas as que existem, não havendo outras categorias fundamentais, apenas esses três.[3] A exposição a seguir inicia pela mais aparente no dia-a-dia.

---

[1] I essay an analysis of what appears in the world. It is not metaphysics that we are dealing with: only logic. Therefore, we do not ask what really is, but only what appears to everyone of us in every minute of our lives. I analyze experience, which is the cognitive resultant of our past lives, and find in it three elements. I call them *Categories*.

[2] Citações dos Collected Papers (PEIRCE, 1931-1958) são feitas entre parênteses no formato "CP V.P, A", onde V é o volume, P o número do parágrafo e A o ano em que o texto foi escrito. A versão para o português, quando sem indicação de tradutor, é do autor.

[3] Em comparação com as categorias propostas por outros filósofos são surpreendentemente poucas: Aristóteles postula dez categorias, Kant nos deu doze, divididas em quatro grupos iguais.

2.1.1 Secundidade

Peirce descreveu fenômenos de secundidade com palavras como "força bruta", "binaridade" (CP 2.84, 1902), "*obsistência* (sugerindo *obviar*, *objeto*, *obstinado*, *obstáculo*, *insistência*, *resistência*, etc.)" (CP 2.89, 1902).[4] A secundidade descreve fenômenos baseados em relações diádicas: identificamos dois elementos interagindo. A maneira mais fácil de observá-la é olhar em volta: o leitor certamente está cercado de objetos que persistem existindo. O mundo material se impõe sobre nós, independentemente da nossa vontade, através de uma ação bruta.

Mas a secundidade não se limita à materialidade. Alguns fenômenos mentais também resistem à mudança. Peço que o leitor imagine, agora, um dragão — ou qualquer outra criatura inexistente, desde que tenha uma cor. Então, responda à pergunta: de que cor é a criatura que acabou de imaginar? Não importa qual seja a resposta, a cor imaginada no momento da resposta não mudará. Claro, teria sido possível imaginar outra criatura, ou a mesma criatura com outra cor. Mas uma vez imaginada, e respondida a questão, nada mudará a cor da criatura, mesmo que a memória falhe. Um caso de secundidade sem materialidade; tudo o que nos precede, portanto, o passado, se impõe sobre nós dessa forma (cf. CP 2.84, 1902).

Peirce vai além, classificando as inferências dedutivas como sendo da natureza da secundidade: em tais modos de raciocínio as conclusões são compelidas pelas premissas. Não é possível negar as conclusões de uma dedução tendo aceito suas premissas.

2.1.2 Terceiridade

Para ver a terceiridade em ação é suficiente pensar sobre o futuro. Não sendo possível que um evento futuro nos afete diretamente — em uma ação impelida pela secundidade —, conclui-se que essa ação do futuro sobre nós é indireta. Uma relação indireta, mediada, tal qual qualquer palavra escrita, como por exemplo a palavra "mar". As marcas que formam a palavra no papel, ou na tela do computador, são uma manifestação de secundidade. Não são o aspecto mais importante da palavra, porque o importante não são as características físicas deste particular conjunto de letras. O que importa é que a palavra, corporificada por suas letras escritas, representa algo que não está aqui; ela cria uma relação entre três elementos — a palavra, o mar e o leitor. Muitos fenômenos são proeminentemente determinados pela terceiridade. Nas palavras de Peirce,

> Algumas ideias de proeminente Terceiridade que, por conta de sua grande importância em filosofia e ciência, requerem estudo atento são generalidade, infinitude, continuidade, difusão, crescimento, e inteligência.[5] (CP 1.340, 1895)

[4] [...] brute force [...] binarity [...] (CP 2.84, 1902). *Obsistence* (suggesting *obviate*, *object*, *obstinate*, *obstacle*, *insistence*, *resistance*, etc.)[...] (CP 2.89, 1902)

[5] Some of the ideas of prominent Thirdness which, owing to their great importance in philosophy and in science, require attentive study are generality, infinity, continuity, diffusion, growth, and intelligence.

Também o conceito de signo, que é o tema da semiótica:

> A [ideia nas quais a terceiridade é predominante] mais fácil dentre as que são de interesse filosófico é a ideia de um signo, ou representação. Um signo representa algo para a ideia que produz ou modifica.[6] (CP 1.339, sem data)

Observa-se que a natureza das leis é de terceiridade, ainda que suas aplicações sejam da natureza da secundidade.

2.1.3 Primeiridade

Qualquer fenômeno de primeiridade não é compelido por nada (não faz parte de uma relação de secundidade) nem é determinado por nenhuma lei (determinado por terceiridade). É dado por si mesmo, somente. Dificilmente é diretamente percebido. Mesmo assim, é possível inferir algumas de suas características. Nas palavras do próprio Peirce:

> A ideia do absolutamente primeiro deve ser inteiramente separada de toda concepção ou referência a qualquer outra coisa; pois o que envolve um segundo é ele próprio um segundo para aquele segundo. O primeiro deve portanto ser presente e imediato, de modo a não ser o segundo para uma representação. Deve ser fresco e novo, pois se velho é segundo do seu estado anterior. Deve ser iniciativo, original, espontâneo e livre; de outro modo é segundo de uma causa determinante. É também algo vívido e consciente; somente assim evita ser o objeto de alguma sensação. Precede toda síntese e toda diferenciação; não tem unidade nem partes. Não pode ser articuladamente pensado: afirme-o, e já terá perdido sua inocência característica; pois a afirmação sempre implica a negação de algo outro. Pare de pensar nele, e terá voado! O que o mundo era para Adão no dia em que abriu os olhos para ele, antes de ter traçado quaisquer distinções, ou tenha se tornado consciente de sua própria existência – isso é primeiro, presente, imediato, fresco, novo, iniciativo, original, espontâneo, livre, vívido, consciente e evanescente. Lembre-se somente que qualquer descrição dele necessariamente o falseia.[7] (CP 1.357, 1887–1888)

---

[6] The easiest of those which are of philosophical interest is the idea of a sign, or representation. A sign stands for something to the idea which it produces, or modifies.

[7] The idea of the absolutely first must be entirely separated from all conception of or reference to anything else; for what involves a second is itself a second to that second. The first must therefore be present and immediate, so as not to be second to a representation. It must be fresh and new, for if old it is second to its former state. It must be initiative, original, spontaneous, and free; otherwise it is second to a determining cause. It is also something vivid and conscious; so only it avoids being the object of some sensation. It precedes all synthesis and all differentiation; it has no unity and no parts. It cannot be articulately thought: assert it, and it has already lost its characteristic innocence; for assertion always implies a denial of something else. Stop to think of it, and it has flown! What the world was to Adam on the day he opened his eyes to it, before he had drawn any distinctions, or had become conscious of his own existence – that is first, present, immediate, fresh, new, initiative, original, spontaneous, free, vivid, conscious, and evanescent. Only, remember that every description of it must be false to it.

Peirce não evita a ideia do "primeiro no segundo; ao contrário, o segundo é precisamente o que não pode ser sem o primeiro"[8] (CP 1.358, 1887-1888). Secundidade não pode existir sem primeiridade, e terceiridade depende da secundidade para se incorporar.

2.2 Signo

A literatura sobre signo e semiose é bastante vasta; uma excelente fonte é o recente texto de Nöth e Santaella (2017). O que segue é o essencial para compreensão do argumento, parte já publicada por Gazoni (2018).

Iniciemos tomando como exemplo as palavras nas linguagens naturais. São signos por excelência. Tendemos a pensar em linguagens naturais como construtos culturais, o que é verdade, mas dizer isso não é o mesmo que afirmar que as palavras podem ser criadas de propósito. Toda tentativa na história de criar uma língua natural universal falhou. Numa moldura peirciana, pode-se encontrar uma explicação para isso a partir de uma afirmação intrigante: signos são determinados por seus objetos (CP 4.531, 1905). Surpreendentemente, essa afirmação contraintuitiva se confirma numa análise mais profunda. Começamos com o conceito se signo.

2.2.1 Signo: básico

A definição de Peirce de signo evoluiu ao longo do tempo, e em seus escritos podemos encontrar inúmeras conceituações. Marty (1997) aponta nada menos que 76 definições retiradas dos textos de Peirce. Algumas mais completas e complexas, outras simplificadas por razões didáticas. Um delas diz:

> Um Signo, ou Representamen, é um Primeiro que está em uma relação tão genuinamente triádica com um Segundo, chamado seu Objeto, que é capaz de determinar que um Terceiro, chamado seu Interpretante, assuma a mesma relação triádica com o Objeto na qual ele próprio se encontra. A relação triádica é genuína, ou seja, seus três membros são unidos por ela de um modo que não consiste em nenhum complexo de relações diádicas. Esta é a razão pela qual o Interpretante, ou Terceiro, não pode estar em uma mera relação diádica com o Objeto, mas deve estar na relação com ele do mesmo modo que o Representamen está.[9] (CP 2.274, 1903)

[8] But we need not, and must not, banish the idea of the first from the second; on the contrary, the second is precisely that which cannot be without the first.

[9] A Sign, or Representamen, is a First which stands in such a genuine triadic relation to a Second, called its Object, as to be capable of determining a Third, called its Interpretant, to assume the same triadic relation to its Object in which it stands itself to the same Object. The triadic relation is genuine, that is its three members are bound together by it in a way that does not consist in any complexus of dyadic relations. That is the reason the Interpretant, or Third, cannot stand in a mere dyadic relation to the Object, but must stand in such a relation to it as the Representamen itself does.

Nessa definição, Peirce iguala signo e representamen. Em outras, o signo é a relação entre representamen, objeto e interpretante. O que se destaca é que um signo tem três constituintes: o representamen, seu objeto e um interpretante. Os três se relacionam não de dois em dois, mas os três numa única e mesma relação, que é a característica mais importante do signo na filosofia peirciana. Algumas características dos três componentes do signo em si:

1. O representamen é o primeiro, ou seja, tem de estar presente para haja ação do signo. Não há uma regra que defina o que pode vir a ser o representamen de um signo. Qualquer coisa pode ser um representamen, o que não quer dizer que tudo seja um representamen. Cada palavra neste texto é um representamen, mas também todo o texto o é, bem como qualquer capítulo dele.
2. O interpretante é o efeito do signo; o processo que o cria é chamado "semiose". Do mesmo modo que para os representamens, não há regras que definam o que pode vir a ser um interpretante. Qualquer coisa pode ser um interpretante, desde que seja o efeito de um signo. Em particular, um interpretante pode ser um outro signo, que tem seu próprio interpretante, que também pode ser outro signo, e assim por diante, numa cadeia de signos potencialmente infinita.
3. Qualquer coisa também pode ser o objeto de um signo, uma vez que o objeto é aquilo a que o signo se refere.

Fica claro que a natureza dos componentes não é o mais importante num signo, já que praticamente qualquer coisa pode participar como componente de um signo. O que precisa ser levado em conta é como esses componentes se relacionam, como o objeto determina o signo. Comecemos com o uso de um signo, tomando o uso de uma palavra já existente como exemplo. Suponha que alguém lhe telefone e pergunte: — Qual a cor de sua roupa? A resposta verdadeira, claro, depende da cor do que estiver vestindo. Essa cor determina a palavra: "verde", por exemplo. O objeto — a cor da roupa — determinou o signo, que é a palavra.

Mas temos a criação de novas palavras. Como o signo — continuando a tomar palavras como exemplo — é determinado pelo objeto aqui? A análise também é simples. A palavra existe porque há algo que precisa ser expresso por ela. O que motiva a criação de palavras é a necessidade de designar o que quer que seja relevante. Palavras que designam objetos irrelevantes dificilmente subsistirão na língua — se subsistirem, é porque o objeto que designam têm, ou ganharam, alguma relevância. Ainda que o som exato da palavra inventada seja arbitrário, ele é pouco importante para o significado. A palavra "árvore" em português foi criada graças ao mesmo objeto que levou à criação da palavra "tree" em inglês. Elas são muito diferentes, mas essa diferença é irrelevante: o objeto[10] é o mesmo.

[10] Objeto dinâmico (cf. Seção 2.2.3), ou seja, aquilo que efetivamente levou à criação das palavras.

Assim, tanto ao criar quanto ao utilizar uma palavra, somente sua forma — o representamen — é arbitrária. Em ambos os casos o signo é determinado por um objeto. Diferentes intérpretes produzirão diferentes interpretantes a partir de um mesmo representamen; lembremos que as diferentes interpretações,[11] quando existem, não determinam o signo: ao contrário, também são determinadas por ele. O signo em si é determinado por seu objeto.

2.2.2 Dez classes de signo

Palavras são apenas um tipo de signo. Diferentes tipos de signo têm diferentes formas de semiose. Apresentamos a seguir a principal (cf. CP 8.341, 1904) classificação peirciana dos tipos de signo para ilustrar a abrangência desse conceito. Peirce chegou a escrever que o universo é "perfundido com signos, se não for composto exclusivamente de signos"[12] (CP 5.448, 1905, rodapé). Signos mais relevantes para a caracterização da inferência, como os argumentos, serão estudados com mais detalhe alhures.

A classificação em dez classes de signo se baseia em três tricotomias. A primeira tricotomia, que classifica de acordo com o signo em si, se é qualidade, existente ou uma lei geral. A segunda classifica os signos de acordo com a relação que têm com seus objetos, e a terceira de acordo com o interpretante. A primeira tricotomia gera as seguintes três subclasses de signo:

Qualissigno que é uma qualidade que é signo. Peirce acrescenta: "não pode agir realmente como signo enquanto não é corporificada; mas a corporificação não tem nada a ver com seu caráter como signo"[13] (CP 2.244, 1903).

Sinsigno que é uma "coisa existente real ou evento que é signo. Só pode sê-lo através de suas qualidades; portanto envolve um qualissigno, ou melhor, vários qualissignos. Mas esses qualissignos são de um tipo peculiar e somente formam um signo por serem realmente corporificados"[14] (CP 2.245, 1903).

Legissigno "é uma lei que é um signo. Esta lei é normalmente estabelecida pelos homens. Todo signo convencional [uma palavra, por exemplo] é um legissigno [mas não o contrário]. Não é um objeto singular, mas um tipo geral que tendo sido acordado, será significante. Todo legissigno significa através de uma instância de sua aplicação, que pode ser denominada

[11] A existência de diferentes interpretações pode ser considerada um dos motores do processo de crescimento do signo, no qual se passa a conhecer melhor seu objeto (cf. NÖTH, 2014).

[12] [...] all this universe is perfused with signs, if it is not composed exclusively of signs.

[13] A *Qualisign* is a quality which is a Sign. It cannot actually act as a sign until it is embodied; but the embodiment has nothing to do with its character as a sign.

[14] A *Sinsign* [...] is an actual existent thing or event which is a sign. It can only be so through its qualities; so that it involves a qualisign, or rather, several qualisigns. But these qualisigns are of a peculiar kind and only form a sign through being actually embodied.

> sua *Réplica*. [...] A Réplica é um Sinsigno. Assim, todo Legissigno requer Sinsignos. Mas estes não são Sinsignos comuns"[15] (CP 2.246, 1903).

Essa primeira tricotomia, como as demais, é subdividida conforme as categorias peircianas. A categoria característica do qualissigno é a primeiridade, e as do sinsigno e do legissigno são, respectivamente, a secundidade e a terceiridade.

A segunda tricotomia estabelece-se na relação do signo com seu objeto. São, também, três diferentes relações com base nas categorias:

Ícone é "um signo que se refere ao Objeto que denota meramente em virtude de características próprias, e que ele possui, exatamente as mesmas, quer tal Objeto exista realmente ou não"[16] (CP 2.247, 1903). Essa definição exige um pouco de aprofundamento: trata-se de uma relação cuja característica é a primeiridade. Nöth e Santaella (2017, p. 51) nos lembram que um qualissigno icônico (ícone puro) é "uma mera possibilidade hipotética da existência de um signo, já que um signo concreto, atualizado é sempre um signo singular e, portanto, não poderia ser um qualissigno em seu estado puro". Portanto, prosseguem os autores, um ícone puro seria um signo não comunicável. Poderia ocorrer em circunstâncias especiais que não fazem parte da semiose cotidiana. Nöth e Santaella nos auxiliam a identificar os ícones quando aparecem em sua forma cotidiana, como sinsignos e legissignos icônicos:

> O critério para defini-los é o da similaridade entre *representamen* e objeto. Peirce fala de um signo que é "semelhante" ao seu objeto (CP 3.362, 1885), mas também se refere a um signo que participa "do caráter do objeto" (CP 4.531, 1905) e, ainda, de um signo "cujas qualidades são semelhantes às do objeto e excitam sensações análogas na mente para a qual é uma semelhança" [(CP 2.299, c. 1895)]. Os seus exemplos são de retratos, pinturas (CP 2.92, 1903), fotografias (CP 2.280, c.1895), metáforas, diagramas, gráficos lógicos (CP 4.418-20, 1903) e até fórmulas algébricas.
>
> Muitos desses signos não são semelhantes aos seus objetos, no sentido ordinário da palavra, ou seja, de semelhança na sua aparência. Por que, por exemplo, as fórmulas algébricas e os diagramas seriam ícones? A chave da iconicidade desses signos reside na noção das correspondências relacionais. Peirce explica: "muitos diagramas não se assemelham de modo algum aos seus objetos quanto à aparência; a semelhança entre eles consiste apenas da relação entre suas partes". (CP 2.282, 1893; NÖTH; SANTAELLA, 2017, p. 52)

[15] A *Legisign* is a law that is a Sign. This law is usually established by men. Every conventional sign is a legisign [but not conversely]. It is not a single object, but a general type which, it has been agreed, shall be significant. Every legisign signifies through an instance of its application, which may be termed a *Replica* of it. [...] The Replica is a Sinsign. Thus, every Legisign requires Sinsigns. But these are not ordinary Sinsigns[...]

[16] An *Icon* is a sign which refers to the Object that it denotes merely by virtue of characters of its own, and which it possesses, just the same, whether any such Object actually exists or not.

Índice é "um signo que se refere ao Objeto que denota em virtude de ser realmente afetado por esse Objeto"[17] (CP 2.248, 1903), uma relação de secundidade. Nöth e Santaella (2017, p. 54) especificam: "tais relações têm, principalmente, o caráter de causalidade, espacialidade e temporalidade".

Símbolo é "um signo que se refere ao Objeto que denota em virtude de uma lei, usualmente uma associação de ideias gerais, que operam fazendo com que o Símbolo seja interpretado como se referindo àquele Objeto. É portanto um tipo geral ou lei, vale dizer, um Legissigno. Como tal age através de uma Réplica"[18] (CP 2.249, 1903). Nöth e Santaella (2017, p. 55) especificam: é o signo da segunda tricotomia "que participa da categoria da terceiridade", situando "o hábito, a regra, a lei" na relação entre representamen e objeto.

A terceira tricotomia considera o signo "do ponto de vista do interpretante de um signo", podendo ser "um rema, um dicente (ou dicissigno) ou um argumento. Esta tríade se baseia na tríada da lógica aristotélica termo, proposição e argumento" (NÖTH; SANTAELLA, 2017, p. 58):

Rema "na lógica é ' simplesmente um nome de classe ou um nome próprio' (CP 8.337, 1904). No sentido mais geral da semiótica, um rema é, portanto, 'qualquer signo que não é verdadeiro nem falso, como quase cada uma palavra por si, exceto sim e não' (*ibid.*)" (NÖTH; SANTAELLA, 2017, p. 58).

Dicente é "a unidade mínima para exprimir ideias que podem ser ou verdadeiras ou falsas. [...] 'A prova característica mais à mão que mostra se um signo é um dicissigno ou não é que o dicissigno é ou verdadeiro ou falso, mas não fornece as razões de ser desta ou daquela maneira' (CP 2.310, 1903)" (NÖTH; SANTAELLA, 2017, p. 60). Uma consideração importante é que as proposições são compostas por um índice, que tem o papel de sujeito, e um ícone, que constitui o predicado da proposição (CP 2.309–331, c.1902).

Argumento de acordo com Nöth e Santaella (2017, p. 60): "logo que o signo supera o quadro proposicional, e passa a participar de um discurso racional mais estendido, chega à categoria da terceira tricotomia. Um *argumento* conecta a informação de signos dicentes por uma necessidade lógica. Ele é um signo do discurso racional. [...] Um argumento é, portanto, 'o signo de uma lei' (CP 2.252, 1903), 'a saber, a lei segundo a qual a passagem das premissas para as conclusões tende a ser a verdadeira' (CP 2.263, 1903)".

O leitor atento observará que havendo três tricotomias — signo enquanto representamen, em sua relação com o objeto e com o interpretante — e tendo cada tricotomia três classificações

[17] An *Index* is a sign which refers to the Object that it denotes by virtue of being really affected by that Object.

[18] A *Symbol* is a sign which refers to the Object that it denotes by virtue of a law, usually an association of general ideas, which operates to cause the Symbol to be interpreted as referring to that Object. It is thus itself a general type or law, that is, is a Legisign. As such it acts through a Replica.

possíveis, deveria haver 27 (3^3) classes de signo. Mas só há dez. A razão disso é que há situações impossíveis do ponto de vista semiótico. Por exemplo, um qualissigno, uma qualidade que é signo: a relação dele com seu objeto não se dá diretamente (ou por relação causal, espacial ou temporal), nem se dá em virtude de uma lei. Portanto, não pode ser nem um índice, nem um símbolo: só pode ser um ícone. Se lembrarmos que a terceiridade pode envolver secundidade e esta envolve primeiridade, mas não o contrário, podemos dizer, como regra geral, que o primeiro (representamen), segundo (objeto) e terceiro (interpretante) relatos de um signo são tais que as categorias dos relatos subsequentes são iguais ou inferiores às categorias dos relatos precedentes. Assim, um sinsigno só pode ser representamen de um índice ou de um ícone, mas não de um

Figura 1 – Classes de signos possíveis, adaptado de Merrell (1996, p. 8)

CATEGORIA	**TRICOTOMIA**		
	1ª (representamen)	**2ª (objeto)**	**3ª (interpretante)**
Primeiridade	qualissigno	ícone	rema
Secundidade	sinsigno	índice	dicente
Terceiridade	legissigno	símbolo	argumento

símbolo (a não ser no caso degenerado em que é uma réplica); um índice participa de um signo cujo interpretante pode ser um rema ou um dicente, mas não um argumento, e assim por diante. Todas as combinações possíveis são assinaladas pelas flechas na Figura 1. Assim, são possíveis apenas dez classes de signo, descritas na Tabela 1.

2.2.3 Constituição do signo: mais detalhes

O próprio Peirce reconheceu que a classificação em dez classes de signo poderia ser estendida; em uma carta a Lady Victoria Welby afirma que há 3^{10} ou 59.049 configurações possíveis de signo. Ele mesmo não as estudou, deixando-as "para futuros exploradores"[19] (CP 8.343, 1908). Tal abundância decorre do detalhamento dos constituintes do signo, que seria composto por um representamen, dois objetos e três interpretantes. O representamen, já vimos. Vejamos os outros dois.

Objeto imediato e objeto dinâmico

Nöth e Santaella (2017, p. 43) nos lembram que "Peirce reconheceu duas espécies de objeto: o objeto imediato e o objeto mediato, real ou dinâmico". Nas palavras de Peirce:

[19] [...] I have 3^{10} or 59049, difficult questions to carefully consider; and therefore I will not undertake to carry my systematical division of signs any further, but will leave that for future explorers.

Tabela 1 – As dez principais classes de signo

Representamen	Objeto	Interpretante	Características	Exemplo
Qualissigno	Icônico	Remático	qualidade que é um signo	sensação de uma cor ou de um sabor
Sinsigno	**Icônico**	Remático	signo concreto, que representa seu objeto por suas qualidades	exemplar individual de um mapa ou diagrama
Sinsigno	**Indicial**	**Remático**	dirige a atenção a um objeto concreto pela sua mera presença	grito espontâneo, por exemplo, de surpresa ou de dor
Sinsigno	Indicial	**Dicente**	é afetado diretamente por seu objeto e dá informações sobre ele	cata-vento, termômetro
Legissigno	**Icônico**	Remático	lei representada iconicamente	placa de trânsito "pedestres" quando mostrada na apostila; diagrama da hélice do DNA; qualquer onomatopeia
Legissigno	**Indicial**	**Remático**	cada um de seus casos é realmente afetado por seu objeto, e atrai atenção para esse objeto	pronome demonstrativo
Legissigno	**Indicial**	**Dicente**	lei geral afetada por um objeto real, de tal modo que forneça informação definida a respeito desse objeto	pregão de vendedor de rua; placa de trânsito específica na rua; comando militar
Legissigno	**Simbólico**	**Remático**	signo convencional que não tem o caráter se uma proposição	qualquer substantivo
Legissigno	Simbólico	**Dicente**	combina símbolos remáticos em uma proposição	qualquer proposição completa
Legissigno	Simbólico	**Argumento**	signo do discurso racional	silogismo

Fonte: adaptado de Nöth e Santaella (2017, p. 61–66)

> Temos de distinguir o Objeto Imediato, o qual é o Objeto como o próprio Signo o representa, e cujo Ser é portanto dependente da sua Representação no Signo, do Objeto Dinâmico, o qual é a Realidade que de algum modo conspira para compelir o Signo à sua Representação.[20] (CP 4.536, 1905)

Para Santaella e Nöth (ibid., p. 43) o objeto dinâmico, fora do signo, é o segmento da realidade mediato e dinâmico porque só pode ser indicado no processo de semiose. Por outro lado, o objeto imediato é como o signo o representa. A relação do signo com seus objetos é melhor ilustrada por Peirce na seguinte passagem:

[20] [W]e have to distinguish the Immediate Object, which is the Object as the Sign itself represents it, and whose Being is thus dependent upon the Representation of it in the Sign, from the Dynamical Object, which is the Reality which by some means contrives to determine the Sign to its Representation.

> Eu defino um *Signo* como qualquer coisa que por um lado é determinado por um Objeto e por outro determina uma ideia na mente de uma pessoa, tal que esta última determinação, que eu denomino *Interpretante* do signo, é desse modo mediatamente determinada por aquele Objeto. Um signo, portanto, tem uma relação triádica com seu Objeto e com seu Interpretante. Mas é necessário distinguir o *Objeto Imediato*, ou o Objeto como o Signo o representa, do *Objeto Dinâmico*, ou Objeto realmente eficiente mas não imediatamente presente.[21] (CP 8.343, 1908)

Fica aqui claramente ilustrado o papel do objeto dinâmico: é ele que determina o signo, que por sua vez o representa como objeto imediato. Este último tem seu ser no signo, nunca fora dele. Fica claro também como pode ser a visão de que "tudo é signo": qualquer coisa que se apresenta à nossa mente é signo, não objeto dinâmico — ainda que em certas circunstâncias nos sintamos tentados a dizer que os objetos que nos cercam são, eles mesmos, que se apresentam à nossa mente. De acordo com o modo peirciano de enxergar o fenômeno, isso é falso: somente o signo tem efeito em nossa mente, não os objetos dinâmicos. E seu efeito é o interpretante.

Interpretante: imediato, dinâmico e final

Iniciemos com o excelente resumo da noção de interpretante que nos é dada por Nöth e Santaella (2017, p. 46):

> Peirce deu uma definição pragmática da significação quando definiu o interpretante como o "próprio resultado significante", ou seja, o "efeito do signo" (CP 5.474-475, c.1903), podendo também ser "algo criado na mente do intérprete" (CP 8.179, 1909). Em conformidade com sua teoria de que as ideias são signos e com a sua visão da interpretação como processo de semiose, também definiu o interpretante como signo: "um signo dirige-se a alguém, isto é, cria na mente dessa pessoa um signo equivalente, ou talvez um signo mais desenvolvido. Chamo o signo assim criado o interpretante do primeiro signo". (CP 2.228, c.1897)

Há mais de um: são três interpretantes. Na classificação que é, aparentemente, a mais madura de Peirce, o interpretante imediato "consiste na *Qualidade* da Impressão que um signo está apto a produzir, não em nenhuma reação real"[22] (CP 8.315, 1909) ou seja, é tudo o que o signo potencialmente pode produzir sobre uma mente. O interpretante dinâmico é "qualquer interpretação que qualquer mente efetivamente faça de um signo"[23] (CP 8.315, 1909) e o interpretante final "não consiste no modo como nenhuma mente age, mas no modo como

[21] I define a *Sign* as anything which on the one hand is so determined by an Object and on the other hand so determines an idea in a person's mind, that this latter determination, which I term the *Interpretant* of the sign, is thereby mediately determined by that Object. A sign, therefore, has a triadic relation to its Object and to its Interpretant. But it is necessary to distinguish the *Immediate Object*, or the Object as the Sign represents it, from the *Dynamical Object*, or really efficient but not immediately present Object.

[22] The Immediate Interpretant consists in the *Quality* of the Impression that a sign is fit to produce, not to any actual reaction.

[23] The Dynamical Interpretant is whatever interpretation any mind actually makes of a sign.

qualquer mente agiria"[24] (CP 8.315, 1909), uma tese melhor explicada em outro trecho: "é aquele que *decidir-se-ia finalmente* ser a interpretação verdadeira se a consideração a respeito do assunto fosse levada tão longe que atingisse uma opinião derradeira"[25] (CP 8.184, 1909).

Essa não é, entretanto, a única caracterização dos três interpretantes feitas por Peirce. Liszka (1990, p. 25) aponta treze versões de classificação nos textos peircianos; Lalor (1997) as resume a duas caracterizações principais: numa, que situa em 1909, os interpretantes são os que vimos acima, "imediato", "dinâmico" e "final". Noutra, que localiza perto de 1906, a tricotomia é "emocional", "energético" e "lógico". A maneira como essas diferentes caracterizações se relacionam tem sido objeto de debate. Lalor, nesse mesmo artigo, expõe a tese de que a classificação de 1906 é um caso especial da classificação de 1909: tratar-se-ia, grosso modo, do caso "antropocêntrico" desta. As teses de Liszka e Lalor foram explicitamente refutadas por Short (1996).

Com efeito, Santaella (2000, p. 66) nos informa que a teoria e as classificações dos interpretantes são foco de grandes controvérsias entre os interpretes de Peirce. Essa autora investiga principalmente duas versões: numa, a classificação que Lalor situa em 1906 se aplica a cada tipo da classificação de 1909 (SANTAELLA, 2000, p. 82–83):

1. Imediato
    1.1. Emocional
    1.2. Energético
    1.3. Lógico
2. Dinâmico
    2.1. Emocional
    2.2. Energético
    2.3. Lógico
3. Final
    3.1. Emocional
    3.2. Energético
    3.3. Lógico

A outra versão investigada pela autora classifica como "emocional", "energético" e "lógico" apenas o interpretante dinâmico (SANTAELLA, 2000, p. 81):

[24] The Final Interpretant does not consist in the way in which any mind does act but in the way in which every mind would act.

[25] But we must also note that there is certainly a third kind of Interpretant, which I call the Final Interpretant, because it is that which *would finally* be decided to be the true interpretation if consideration of the matter were carried so far that an ultimate opinion were reached.

1. Imediato
2. Dinâmico
    2.1. Emocional
    2.2. Energético
    2.3. Lógico
3. Final

Adotamos essa interpretação no presente trabalho. Na Seção 3.4 apresentaremos mais detalhes quanto à classificação do interpretante como “emocional”, “energético” e “lógico”.

3 Agência semiótica

Neste capítulo aprofundaremos conceitos da filosofia peirciana relacionados à agência semiótica. Procuramos uma conceituação genérica, que possa ser aplicada a objetos como o computador. Uma dificuldade natural nessa tarefa é que muitos desses conceitos são apresentados por Peirce em textos que se referem a seres humanos e a seres vivos em geral. É preciso compreender como esses conceitos se aplicam a objetos. Por exemplo, ao tratar de estética e ética, é evidente que não se pode esperar que objetos, ou mesmo animais, apresentem um comportamento baseado numa moral. Não obstante, ética e estética têm, segundo Peirce, um papel na lógica, ou semiótica. O que fizemos foi compreender esse papel e procurar elementos que possam exercer esse mesmo papel num objeto ou num computador. Assim, ao nos referirmos, mais adiante no texto, aos hábitos éticos do computador, claro está que não nos referimos a um comportamento baseado numa moral, mas aos elementos que têm, no raciocínio do agente semiótico computador, um papel similar ao dos hábitos éticos no raciocínio humano.

O capítulo foi organizado iniciando com uma definição de agente na Seção 3.1. Segue com uma introdução a respeito da mente na Seção 3.2, na qual são apresentados conceitos que serão utilizados e ampliados nas seções seguintes. A Seção 3.3 conceitua como se dá o raciocínio em si, sem preocupação como a maneira como o agente evolui ou como se relaciona com o ambiente. A evolução do agente através do aprendizado, ou mudança de hábito, é vista na Seção 3.4. Na Seção 3.5 exploraremos como se dá a interação do agente com o mundo, utilizando conceitos da biossemiose de acordo com o proposto pelo biólogo alemão Jakob von Uexküll (1864-1944); nessa seção estudaremos também a visão peirciana de raciocínio instintivo. O raciocínio abdutivo, uma das formas de raciocínio vistas na seção 3.3, merecerá conjeturas adicionais na Seção 3.6. Como uma forma de consolidar o que foi visto até esse ponto, apresentamos alguns diagramas ilustrativos na Seção 3.7. A seção 3.8 apresenta algumas considerações que serão importantes para a compreensão do processo de diálogo de acordo com Peirce, além de conceituar a vagueza e a indeterminação à luz de sua filosofia. Uma visão resumida do capítulo está na Seção 3.9. A lista de símbolos matemáticos utilizados nas proposições lógicas aparece no Apêndice A.

3.1 Caracterização de agente

Antes de iniciar, é preciso esclarecer quais aspectos do conceito de agente são importantes no presente trabalho. As considerações aqui feitas são inteiramente aderentes à filosofia peirciana, embora possam parecer, nesse momento, um pouco contraintuitivas para os leitores menos familiarizados com a obra do filósofo.

Santaella (2005) afirma que, de acordo com Peirce, há "dois tipos de ação no universo, a ação diádica, que 'é bruta, não inteligente e desligada do resultado que pode advir dela' (CP 6.332), e a ação triádica que é ação governada por lei".[1] Uma ação semiótica é triádica, e uma dificuldade parece residir no fato de que o computador, sendo um objeto, aparentemente só atua de forma diádica. Podemos começar a decifrar esse enigma com a ideia de localização da mente.

Para Peirce, a localização da mente é localização "num sentido no qual uma coisa pode estar em dois lugares ao mesmo tempo"[2] (CP 7.366, 1902). Não está em um só lugar. Uma mesma ação semiótica pode começar em um agente e terminar em outro. Escrevemos uma ideia no papel; a ideia está localizada simultaneamente em quem escreveu, no texto no papel e em quem leu o texto. Em particular, o texto físico participa de uma ação triádica, uma ação semiótica; uma ação que pode ocorrer sem a presença do autor da ideia. Nesse sentido, o texto é um agente autônomo. De fato, todo signo o é, como nos lembra Nöth (2009, p. 11). Entretanto, estamos mais interessados na forma como o computador atua em termos da relação entre o programa e o equipamento, que é apenas uma aspecto de sua atuação como signo.

Valemo-nos então do conceito peirciano de "personalidade". Peirce a caracteriza como um feixe de hábitos:[3] "...cada personalidade é baseada num 'feixe de hábitos', conforme o dito no qual o homem é um feixe de hábitos"[4] (CP 6.228, 1898). Estamos interessados, portanto, nos hábitos do agente. Por ora, vale-nos que qualquer entidade, ou conjunto de entidades, que apresente um conjunto identificável de hábitos, possui características que nos interessam ao tomá-la como agente. Por exemplo, a natureza, a sociedade e cada um de seus membros, as colmeias de abelha, além, é claro, dos computadores programados.

3.2 Introdução: mente e ação semiótica

Santaella (2005) apresenta um quadro bastante completo do conceito de mente em Peirce; tal conceito se apoia no conceito de semiose, cuja noção nos permite "analisar desde as formas mais rudimentares de quase-mente até os sistemas integralmente inteligentes" (SANTAELLA, 2005). O texto de Santaella foi usado como um guia nesta parte do trabalho.

Para apreciar a abrangência do conceito peirciano de mente é preciso levar em conta a posição de Peirce a respeito da filosofia e do método de Descartes. Santaella (2004, p. 32) afirma que Peirce foi o primeiro filósofo a refutar completamente o método de Descartes e apresentar uma alternativa consistente. Uma caraterística da filosofia de Descartes (2004) à qual Peirce se opõe frontalmente é a dualidade mente-matéria.

---

1 Aqui, a autora utiliza a palavra "lei" num sentido peirciano, que é um sentido que se aproxima da ideia de hábito, como veremos na Seção 3.2.2.

2 It is localization in a sense in which a thing may be in two places at once.

3 O conceito de hábito será melhor explicado adiante, na Seção 3.2.2.

4 [...] each personality is based upon a "bundle of habits," as the saying is that a man is a bundle of habits.

3.2.1 Mente e matéria

Descartes (2004, p. 70) conclui a dualidade mente-corpo a partir da aplicação de seu método. Para ele, a essência da mente e a essência do corpo são duas substâncias essencialmente diferentes. Essa doutrina ganhou o nome de "dualismo mente-corpo" ou "dualismo cartesiano". Peirce era um monista (em oposição a "dualista") e faz a seguinte consideração:

> A velha noção dualista, tão proeminente no cartesianismo, de que mente e matéria são dois tipos radicalmente diferentes de substância dificilmente encontrará defensores hoje em dia. Rejeitando-a, somos levados a uma forma de hilopatia[5] chamada monismo. Então surge a questão se leis físicas por um lado e leis psíquicas por outro devem ser tomadas como–
> (a) independentes, uma doutrina frequentemente denominada monismo, mas que eu nomearia neutralismo; ou,
> (b) a lei psíquica sendo derivada e especial, somente a lei física sendo primordial, que é o materialismo; ou,
> (c) a lei física sendo derivada e especial, somente a lei psíquica sendo primordial, que é o idealismo.
> A doutrina materialística parece-me tão repugnante para a lógica científica quanto para o bom-senso; uma vez que requer que suponhamos que um certo tipo de mecanismo sentirá uma regularidade, derradeira e inexplicável — uma hipótese absolutamente irredutível à razão; e a única justificativa possível para uma teoria é que ela torna as coisas claras e razoáveis.
>
> O neutralismo é suficientemente condenado pela máxima lógica conhecida como navalha de Ockham, isto é, que não se suponha mais elementos independentes do que o necessário. Parear aspectos interiores e exteriores da substância parece torná-los ambos primordiais.
>
> A única teoria inteligível do universo é a do realismo objetivo, matéria sendo mente exaurida, hábitos inveterados tornando-se leis físicas.[6] (CP 6.24–25, 1891)

Portanto, para Peirce, não só não há diferença essencial entre mente e matéria, como, além disso, as leis mentais são primevas; a natureza essencial do que há é mental, sendo a matéria "effete

[5] A capacidade de um espírito penetrar e afetar a matéria.

[6] The old dualistic notion of mind and matter, so prominent in Cartesianism, as two radically different kinds of substance, will hardly find defenders today. Rejecting this, we are driven to some form of hylopathy, otherwise called monism. Then the question arises whether physical laws on the one hand and the psychical law on the other are to be taken –
(a) as independent, a doctrine often called monism, but which I would name neutralism; or,
(b) the psychical law as derived and special, the physical law alone as primordial, which is materialism; or,
(c) the physical law as derived and special, the psychical law alone as primordial, which is idealism.
The materialistic doctrine seems to me quite as repugnant to scientific logic as to common sense; since it requires us to suppose that a certain kind of mechanism will feel, which would be a hypothesis absolutely irreducible to reason – an ultimate, inexplicable regularity; while the only possible justification of any theory is that it should make things clear and reasonable.

Neutralism is sufficiently condemned by the logical maxim known as Ockham's razor, i.e., that not more independent elements are to be supposed than necessary. By placing the inward and outward aspects of substance on a par, it seems to render both primordial.

The one intelligible theory of the universe is that of objective idealism, that matter is effete mind, inveterate habits becoming physical laws.

mind" — mente estéril, gasta, exausta. Isso também se apoia numa outra doutrina peirciana, a do sinequismo.

Sinequismo: num texto intitulado "A lei da mente"[7] (CP 6.102–163, 1891), Peirce postula que a continuidade é um conceito primordial em filosofia (CP 6.103, 1891) e dá a isso o nome de sinequismo. De acordo com ele, ideias se concatenam umas às outras por continuidade (CP 6.143, 1891) e não são afetadas a não ser por outras ideias. Diante da constatação natural de que ideias são afetadas pela matéria — afinal percebemos o mundo material e pensamos sobre ele — somos obrigados a concluir que "o que chamamos de matéria não está completamente morto, é apenas mente constrangida por hábitos"[8] (CP 6.158, 1891).

A visão de que as leis mentais são primordiais e as leis físicas derivadas é ilustrada em muitos pontos na obra de Peirce, o que facilita a compreensão do alcance do conceito. Por exemplo,

> O pensamento não está necessariamente conectado a um cérebro. Aparece no trabalho das abelhas, em cristais, e por todo o mundo físico; e não se pode negar que ele esteja realmente lá, tanto quanto as cores, formas, etc., dos objetos estão realmente lá.[9] (CP 4.551, 1905)

Em outro trecho Peirce inicia afirmando que a mente não pode ser confundida com a consciência, sendo esta um aspecto secundário daquela, ao contrário da crença dos psicólogos de sua época. E prossegue com uma argumentação que permite concluir que a mente se estende para além do corpo:

> Um psicólogo corta fora um lobo do meu cérebro (*nihil animale me alienum puto*) e então quando percebe que não posso me expressar diz: "Veja que sua capacidade para linguagem estava localizada naquele lobo". Sem dúvida estava; e se então ele tivesse furtado meu tinteiro, eu não teria sido capaz de continuar minha argumentação até que tivesse conseguido outro. Sim, os próprios pensamentos não me teriam vindo. Portanto, minha capacidade para argumentação está do mesmo modo localizada no meu tinteiro. Trata-se de localização num sentido no qual uma coisa pode estar em dois lugares ao mesmo tempo. Dentro da teoria de que a distinção entre fenômenos psíquicos e físicos é a distinção entre causação final e eficiente, é suficientemente claro que o tinteiro e o lobo cerebral têm a mesma relação geral com as funções da mente.[10] (CP 7.366, 1902)

---

7 The Law of Mind.

8 [. . .] what we call matter is not completely dead, but is merely mind hidebound with habits.

9 Thought is not necessarily connected with a brain. It appears in the work of bees, of crystals, and throughout the purely physical world; and one can no more deny that it is really there, than that the colors, the shapes, etc., of objects are really there.

10 A psychologist cuts out a lobe of my brain (*nihil animale me alienum puto*) and then, when I find I cannot express myself, he says, 'You see your faculty of language was localized in that lobe.' No doubt it was; and so, if he had filched my inkstand, I should not have been able to continue my discussion until I had got another. Yea, the very thoughts would not come to me. So my faculty of discussion is equally localized in my inkstand.

Talvez um exemplo mais familiar seja a realização de uma operação aritmética. Podemos pensar que se trata de uma ação que pode ser executada exclusivamente no cérebro. Isso é verdade, se os operandos são relativamente pequenos. Mas eles podem ser muito longos. Tomemos, por exemplo, dois números com cem dígitos cada, escolhidos ao acaso. Podemos afirmar que dificilmente uma pessoa será capaz de calcular o produto deles utilizando apenas o cérebro: será preciso também, no mínimo, lápis e papel. Nesse caso, somos obrigados a concordar que uma parte da ação que é executar a multiplicação desses dois números, do jeito que aprendemos na escola primária, se dá fora do cérebro, no lápis e papel. Ou: assim como precisamos do cérebro para realizar essa multiplicação porque sem ele a tarefa se torna impossível, precisamos também de lápis e papel, sem o quê a multiplicação se torna igualmente impossível.

A ausência de localização específica para a mente permite inferir que a ação semiótica pode ocorrer a distância e independentemente de sincronia. A ideia de um trem de pensamentos (expressão utilizada por Peirce) que vai de uma pessoa a outra é observável em fenômenos culturais: grande parte de nossas escolhas, comportamentos e preferências pode ser considerada resultante de ações semióticas que começaram fora de nós, até mesmo em outras épocas.

3.2.2 Hábito e aprendizagem

Uma questão que pode surgir diante da afirmação de que a matéria é mente estéril (Seção 3.2.1) tem a ver com o fato de que objetos inanimados não se comportam como mente: uma cadeira não reage como um ser humano, não importa o quanto se insista. Aqui revelam-se importantes os aspectos do agente que destacamos na Seção 3.1. A mente em si não é necessariamente localizada; já algo que possui um feixe de hábitos pode ser localizado mais facilmente. Vejamos então o conceito peirciano de hábito:

> [Prontidão] para agir de certo modo sob certas circunstâncias e quando impulsionado por um dado motivo é um hábito; e um hábito deliberado, ou autocontrolado, é precisamente uma crença. (CP 5.480, 1907)[11]

Ou seja, hábito é a tendência a comportamento similar em similares circunstâncias — mas não deterministicamente, observe as palavras "tendência" e "similar" (cf. também CP 5.487, 1907). Para Peirce o conceito de hábito não se aplica somente ao que é vivo: leis mecânicas "não são nada além de hábitos adquiridos, como todas as regularidades da mente, incluindo a própria tendência de formar hábitos"[12] (CP 6.268, 1891). Assim, se quisermos enxergar a mente na

It is localization in a sense in which a thing may be in two places at once. On the theory that the distinction between psychical and physical phenomena is the distinction between final and efficient causation, it is plain enough that the inkstand and the brain-lobe have the same general relation to the functions of the mind.

[11] [Readiness] to act in a certain way under given circumstances and when actuated by a given motive is a habit; and a deliberate, or self-controlled, habit is precisely a belief. (CP 5.480, 1907)

[12] [...] mechanical laws are nothing but acquired habits, like all the regularities of mind, including the tendency to take habits, itself; [...]

cadeira, devemos procurar o feixe de hábitos que a caracteriza: no caso, são as leis da natureza que regem seu comportamento.

Aprendizagem é mudança de hábito: aquisição de novos e abandono de antigos (cf. CP 1.390, 1890). Os hábitos da cadeira não mudam, ela não aprende. Esses hábitos da cadeira são também os hábitos de muitos outros objetos, são as leis que regem o comportamento do que é inanimado: uma parte da mente do Universo.

De um modo geral, quase todos os objetos, se tomados como agentes, têm o mesmo comportamento: o feixe de hábitos que os caracteriza são as leis da natureza. Isso não quer dizer que seu comportamento seja simples. Uma cadeira é mais simples que um alicate, que por sua vez é mais simples que uma bicicleta, esta mais simples que um automóvel. Embora o feixe de hábitos de todos eles possa ser conhecido através do conhecimento das leis da natureza e da construção de cada um, a complexidade de nossa interação com cada um deles é bastante diferente. Jiang, Korpas e Raney (2019), inspirados no comportamento de plantas, criaram objetos ainda mais complexos: apresentam comportamento lógico. Sem aparatos eletrônicos ou controles externos, os autores imprimiram em 3D objetos que incorporam operadores lógicos e com isso dão respostas complexas a conjuntos de estímulos. Por exemplo, se comportam como uma armadilha que se fecha quando tocada, mas somente em presença de um solvente.

Um tipo de objeto entretanto foge à previsibilidade: não existe um procedimento geral capaz de predizer o comportamento de um computador, pensado como uma implementação de uma máquina de Turing universal: tal procedimento geral é uma impossibilidade matemática. Não é que o comportamento dos computadores seja imprevisível no mesmo sentido em que, digamos, o decaimento radioativo é imprevisível. As máquinas de Turing são capazes de poucos movimentos diferentes, e cada movimento tomado isoladamente é inteiramente previsível. O artigo de Turing (1936) demonstra que, mesmo que se saiba como foi construída, não é possível encontrar um procedimento geral capaz de determinar o que a máquina fará (examinaremos com um pouco mais de detalhe esse comportamento dos computadores nos Capítulos 4 e 5). Chama a atenção que um resultado matemático, um produto puro da mente humana — o artigo de Turing —, de certo modo invade o mundo das coisas que podem ser tocadas e permite a criação de objetos cujo comportamento não poderá jamais ser determinado por nenhum procedimento geral, permanecendo como a continuação de alguma ação semiótica prévia.

3.2.3 Controle e autocontrole

O processo de aprendizado, para Peirce (ou seja, aprendizado como mudança de hábito), não é necessariamente intencional. Peirce concebe aprendizado como um processo no qual controle e autocontrole se dão em diversos graus:

> Retornando ao autocontrole, o qual posso brevemente esboçar no momento, evidentemente há inibições e coordenações que escapam inteiramente à consciência. Há, em seguida, modos de autocontrole que parecem instintivos. A seguir, há um tipo de autocontrole que resulta de treinamento. Depois, um homem pode ser seu próprio treinador e assim controlar seu autocontrole. Quando esse ponto é atingido muito, senão todo o treinamento pode ser conduzido na imaginação. Quando um homem treina a si mesmo, desse modo controlando o controle, ele tem de ter alguma regra moral em vista, ainda que seja uma regra especial ou irracional. Mas em seguida ele pode dedicar-se a melhorar essa regra; ou seja, exercitar um controle sobre seu controle sobre o controle. Para fazê-lo ele tem de ter em vista algo mais elevado que uma regra irracional. Ele tem de ter algum tipo de princípio moral. Esse, por sua vez, pode ser controlado tendo como referência um ideal estético do que é bom. Há certamente mais graus do que os que enumerei. Talvez seu número seja indefinido. Os brutos são certamente capazes de mais de um grau de controle; mas parece-me que nossa superioridade em relação a eles é mais devida ao nosso maior número de graus de autocontrole do que à nossa versatilidade.[13] (CP 5.533, 1905)

Claramente a autonomia do agente, e sua superioridade, é proporcional ao controle que tem sobre os próprios hábitos. É conveniente observar que, como muitos outros conceitos em Peirce, o conceito de autonomia não é binário, no sentido de que ou se tem ou não tem autonomia, ou autocontrole. Percebe-se uma gradação, uma continuidade que vai da situação de menor autonomia para a de maior autonomia.

O trecho também destaca o papel do princípio moral nesse processo, que foi abordado com mais detalhe por Peirce no contexto exposto a seguir.

3.2.4 Estética, ética e lógica

Peirce entendia estética, ética e lógica, ou semiótica, como ciências normativas porque determinam as leis de raciocínio que deverão ser seguidas por todas as demais ciências: "lógica quanto a representações da verdade, ética quanto aos esforços da vontade, e estéticas em objetos considerados simplesmente em suas presentações"[14] (CP 5.36, 1903). As relações entre essas ciências sublinham a dependência que a semiótica tem das outras duas.

---

[13] To return to self-control, which I can but slightly sketch, at this time, of course there are inhibitions and coördinations that entirely escape consciousness. There are, in the next place, modes of self-control which seem quite instinctive. Next, there is a kind of self-control which results from training. Next, a man can be his own training-master and thus control his self-control. When this point is reached much or all the training may be conducted in imagination. When a man trains himself, thus controlling control, he must have some moral rule in view, however special and irrational it may be. But next he may undertake to improve this rule; that is, to exercise a control over his control of control. To do this he must have in view something higher than an irrational rule. He must have some sort of moral principle. This, in turn, may be controlled by reference to an esthetic ideal of what is fine. There are certainly more grades than I have enumerated. Perhaps their number is indefinite. The brutes are certainly capable of more than one grade of control; but it seems to me that our superiority to them is more due to our greater number of grades of self-control than it is to our versatility.

[14] [...] *Logic* in regard to representations of truth, *Ethics* in regard to efforts of will, and *Esthetics* in objects considered simply in their presentation.

> Se a conduta deve ser completamente deliberada, o ideal deve ser um hábito de sentimento que cresceu sob a influência de uma série de autocríticas e de heterocríticas; e a teoria da formação deliberada de tais hábitos de sentimento é o que deveria ser entendido como estética.[15] (CP 1.574, 1905)

Inspirado em Auguste Comte (1798-1857), Peirce classificava as ciências de acordo com a abstração, da mais abstrata para a menos abstrata. Estas últimas, as ciências mais concretas e especiais, "extraindo seus princípios das mais abstratas e gerais"[16] (CP 2.119, 1902). Dentro dessa ordenação, a lógica ou semiótica extrai seus princípios da ética e da estética; a ética, por sua vez, os extrai da estética (cf. CP 2.120, 1902).

Santaella (1994) debruçou-se sobre a questão estética em Peirce. Resumindo seu texto ao limite — o que é suficiente para nosso propósito —, podemos dizer que as razões pelas quais a lógica deveria perseguir proposições verdadeiras não podem ser justificadas pela lógica somente. Ela depende da ética, que define quais ações devem ser tomadas. E ainda que a ética defina as ações, o que é desejável por si é definido pela estética; portanto, deve determinar a finalidade das ações escolhidas pela ética. Vamos aqui também, nos lembra Santaella, as tonalidades das categorias peircianas: estética e primeiridade, ética e secundidade, semiótica e terceiridade.

3.3 Raciocínio

Inferir, ou raciocinar, consiste em chegar a conclusões a partir de premissas. A classificação peirciana das formas de inferência evoluiu com o tempo: sua classificação inicial foi alterada por volta de 1905, especialmente no que toca à dedução e à indução, descritas a seguir. Nessa última versão, os argumentos são classificados de acordo com a relação entre conclusões e premissas. As classes de inferência — dedução, abdução e indução — são baseadas nas categorias.

3.3.1 As formas de raciocínio

Na *dedução* as conclusões são ligadas diretamente à premissas; é principalmente um fenômeno de secundidade, porque a conclusão é compulsória no sentido de que "você é obrigado logicamente a admitir a conclusão"[17] (PEIRCE, 1963–66, manuscrito 754, 1907). Por essa razão, as deduções podem ser implementadas em máquinas, uma vez que esta ligação pode ser reproduzida fisicamente.

---

[15] If conduct is to be thoroughly deliberate, the ideal must be a habit of feeling which has grown up under the influence of a course of self-criticisms and of hetero-criticisms; and the theory of the deliberate formation of such habits of feeling is what ought to be meant by esthetics.

[16] [...] those which are more concrete and special drawing their principles from those which are more abstract and general.

[17] Compulsive means that you are logically forced to admit the conclusion.

A proposição resultante de uma *abdução* não é necessariamente verdadeira; é antes um hipótese que, se verdadeira, explica um estado de coisas. É uma forma de inferência que não é gerada por nenhuma regra, nem por nenhuma força compulsiva. Peirce afirma que a adoção desta hipótese se dá por similaridade: os fatos *parecem* se ajustar à veracidade de uma hipótese abdutiva. Também afirma que a abdução é "o único tipo de argumento que inicia uma nova ideia"[18] (CP 2.96, 1902), sendo por isso classificada como "originária". Sua categoria mais característica é a primeiridade.

A *indução* ocorre em mais de uma etapa: dado um estado de coisas, cria-se por abdução uma hipótese sobre ele; então, supondo-se que a hipótese é verdadeira, deduz-se experimentos que a confirmem, ou que possam negá-la. É o resultado dos experimentos que valida ou não a veracidade da hipótese. Essa veracidade permanece intacta enquanto os experimentos não mostrarem o oposto. É um fenômeno de terceiridade; sua validade "consiste no fato de que ela age de acordo com um método que apesar de poder dar resultados provisórios incorretos, ainda assim, se aplicado continuamente, eventualmente corrigirá tais erros"[19] (PEIRCE, 1976, p.319, 1906).

Interessante notar que a classificação anterior de Peirce, de 1892 (cf. CP 6.144–147, 1892), leva em conta a abrangência e o modo de formação do hábito que leva às inferências. Assim, a dedução seria caracterizada pela terceiridade porque obedece às leis do silogismo; e a indução, pela secundidade, pois o que a confirma são eventos que se impõem no mundo físico; nesse caso, a inferência teria natureza probabilística, muito como a hipótese, nome que Peirce então dava à abdução.

3.3.2 O raciocínio diagramático

O processo de inferência não é necessariamente instantâneo. De acordo com Peirce, todo raciocínio é diagramático; e um importante passo nisso é a criação de abstrações. Nas suas palavras:

> Todo raciocínio necessário, sem exceção, é diagramático. Ou seja, construímos um ícone do nosso estado hipotético de coisas e passamos a observá-lo. Essa observação nos leva a suspeitar de que algo é verdadeiro, o que podemos ou não conseguir formular com precisão, e passamos a investigar se esse algo é mesmo verdadeiro. Para esse propósito é necessário criar um plano de investigação e esta é a parte mais difícil de toda a operação. Não somente temos de selecionar quais aspectos do diagrama são pertinentes à atenção, mas é também muito importante repetidamente retornar a certas características. De outro modo, embora nossas conclusões possam estar corretas, não serão as conclusões que particularmente estamos almejando. Mas a maior arte consiste em introduzir abstrações adequadas. Com isso quero dizer tal transformação em nossos

[18] An Abduction is Originary in respect to being the only kind of argument which starts a new idea.

[19] The validity of Induction consists in the fact that it proceeds according to a method which though it may give provisional results that are incorrect will yet, if steadily pursued, eventually correct any such error.

> diagramas que características de um diagrama podem aparecer em outro como coisas.[20] (CP 5.162, 1903)

O trecho descreve o processo de raciocínio baseado no diagrama. Consiste, principalmente, na observação de um ícone: um signo que guarda semelhança com seu objeto. Um ícone que pode ser mental, uma imagem que construímos e passamos a observar. Essa observação gera *insight*, ideias que mesmo que não sejam formuladas com precisão, convidam e direcionam reflexão adicional sobre o tema. Aprofundando-se no tema dos modelos científicos, Nöth (2018) adiciona que a natureza icônica dos diagramas não os reduz a ícones puros; ao contrário, embora seja sua natureza icônica que se presta à observação livre, são compostos por índices e símbolos que complementam o signo observado, enriquecendo assim o processo de inferência.

Note-se que Peirce, indiretamente, toca no tema do autocontrole quando propõe que é preciso escolher aspectos a observar, mas retornar a características do diagrama como uma forma de garantir que se chega à conclusão esperada, e não a outra conclusão, ainda que correta.

3.3.3 Abstração

Na seção anterior, vemos que Peirce destaca a importância de construir "abstrações adequadas", e as caracteriza como transformações nos diagramas, transformações nas quais características de um diagrama passam a serem vistas como coisas em outro. Não foi o único texto em que tratou a questão, que aparece em outros pontos de sua obra.

Peirce cita com frequência um trecho da comédia "O Doente Imaginário", de Molière, no qual um doutor em medicina, num exame oral, pergunta ao candidato, "por que o ópio faz com que as pessoas durmam?", e o bacharel responde: "porque o ópio possui uma virtude dormitiva". Uma piada no texto, sem dúvida, porque é um malabarismo de linguagem que não acrescenta nada ao pensamento; não obstante, Peirce aponta esse mecanismo como essencial ao pensamento matemático: trata-se de uma *abstração hipostática*. Vejamos o que é e como se diferencia de outro tipo de abstração nas palavras de Peirce:

> Procure nos modernos tratados de lógica, e verá que quase todos caem em um de dois erros, que é como os considero; o de deixar de lado a doutrina da abstração (no sentido no qual um nome abstrato aponta uma abstração) como

[20] All necessary reasoning without exception is diagrammatic. That is, we construct an icon of our hypothetical state of things and proceed to observe it. This observation leads us to suspect that something is true, which we may or may not be able to formulate with precision, and we proceed to inquire whether it is true or not. For this purpose it is necessary to form a plan of investigation and this is the most difficult part of the whole operation. We not only have to select the features of the diagram which it will be pertinent to pay attention to, but it is also of great importance to return again and again to certain features. Otherwise, although our conclusions may be correct, they will not be the particular conclusions at which we are aiming. But the greatest point of art consists in the introduction of suitable abstractions. By this I mean such a transformation of our diagrams that characters of one diagram may appear in another as things.

> sendo um tópico gramatical com o qual o lógico não precisa particularmente se preocupar; e o de confundir abstração, nesse sentido, com a operação mental através da qual prestamos atenção a um traço do percepto a despeito de outros. As duas coisas são inteiramente desconectadas. O fato mais ordinário do percepto, tal como "isto é luz", envolve abstração *prescisiva*, ou *prescisão*. Mas abstração *hipostática*, a abstração que transforma "isto é luz" em "há luz aqui", que é o sentido que eu normalmente atribuo à palavra abstração (uma vez que *prescisão* atribuo à abstração prescisiva) é um modo muito especial de pensamento. Consiste em tomar um traço de um ou mais perceptos (depois de já ter sido prescindido de outros elementos do percepto) de modo a tomar forma proposicional em um julgamento (de fato, pode operar em qualquer julgamento que seja), e em conceber esse fato como consistindo na relação entre o sujeito daquele julgamento e outro sujeito, que tem um modo de ser que consiste meramente na verdade das proposições das quais o termo concreto correspondente é predicado. Assim, transformamos a proposição "mel é doce" em "mel possui doçura". "Doçura" pode ser considerada fictícia em certo sentido. Mas uma vez que o modo de ser atribuído a ela *consiste* em nada além do fato de que algumas coisas são doces, e não se finge ou imagina que ela tenha algum outro modo de ser, não há, no fim das contas, ficção nenhuma.[21] (CP 4.235, 1902)

O trecho, além de explicar os conceitos de abstração prescisiva e hipostática, detalha a ideia de tomar uma característica de um diagrama e transformá-lo em coisa no outro diagrama, vista na seção anterior. A tal "virtude dormitiva" é uma abstração hipostática, pois cria uma coisa a partir de uma característica do ópio; é essa coisa que será o sujeito das proposições subsequentes decorrentes de uma possível investigação científica.

Num manuscrito que não publicou — uma palestra que daria em Harvard —, Peirce deixa claro que se pode determinar uma espécie de hierarquia de abstrações:

> Uma abstração é uma substância cujo ser consiste na verdade de alguma proposição a respeito de uma substância mais primária.
>
> Por uma substância primária quero dizer uma substância cujo ser é independente do que possa ser verdade a respeito de qualquer outra coisa. Se há ou não

[21] Look through the modern logical treatises, and you will find that they almost all fall into one or other of two errors, as I hold them to be; that of setting aside the doctrine of abstraction (in the sense in which an abstract noun marks an abstraction) as a grammatical topic with which the logician need not particularly concern himself; and that of confounding abstraction, in this sense, with that operation of the mind by which we pay attention to one feature of a percept to the disregard of others. The two things are entirely disconnected. The most ordinary fact of perception, such as "it is light," involves *precisive* abstraction, or *prescission*. But *hypostatic* abstraction, the abstraction which transforms "it is light" into "there is light here," which is the sense which I shall commonly attach to the word abstraction (since *prescission* will do for precisive abstraction) is a very special mode of thought. It consists in taking a feature of a percept or percepts (after it has already been prescinded from the other elements of the percept), so as to take propositional form in a judgment (indeed, it may operate upon any judgment whatsoever), and in conceiving this fact to consist in the relation between the subject of that judgment and another subject, which has a mode of being that merely consists in the truth of propositions of which the corresponding concrete term is the predicate. Thus, we transform the proposition, "honey is sweet," into "honey possesses sweetness." "Sweetness" might be called a fictitious thing, in one sense. But since the mode of being attributed to it *consists* in no more than the fact that some things are sweet, and it is not pretended, or imagined, that it has any other mode of being, there is, after all, no fiction.

> alguma substância primária nesse sentido podemos deixar para os metafísicos discutirem.
>
> Por uma substância *mais* primária quero dizer uma cujo ser não depende de tudo que o ser de uma substância menos primária [é], mas apenas de uma parte dele.[22] (PEIRCE, 1976, p. 162, 1903)

Há portanto uma ordenação possível em termos de primariedade, e as abstrações são menos primárias, ou seja, seu ser consiste na verdade de alguma proposição a respeito de uma substância cujo ser não depende da verdade do todo da abstração. Assim, a abstração "doçura" depende das verdades do mel e do açúcar serem doces (mais especificamente, depende das proposições "mel é doce" e "açúcar é doce" serem verdadeiras), mas o mel e o açúcar, embora possuam doçura (ou seja, as proposições "mel possui doçura" e "açúcar possui doçura" são verdadeiras), não dependem da abstração "doçura" para serem doces. Mel e açúcar são mais primários que a doçura.

3.4 Hábito e mudança de hábito

A visão peirciana sobre como os hábitos se formam e mudam baseia-se nos conhecimentos da época a respeito da fisiologia do sistema nervoso. Num texto introdutório ao seu trabalho sobre lógica algébrica (CP 3.154–161, 1880), Peirce associa o surgimento da lógica ao funcionamento do sistema nervoso, que embute em si a noção de hábito. A ideia é que quando um grupo de nervos é estimulado, "gânglios mais intimamente conectados ao grupo de nervos são levados a um estado ativo", o que em geral tem como resultado movimento corporal.[23] Se o estímulo continua, mais e mais gânglios são ativados; em seguida as partes que primeiro foram excitadas começam a mostrar fadiga. Ambos os efeitos resultam em mudanças nos movimentos do corpo. Se o estímulo inicial sobre os nervos é removido, a excitação diminui. E o resultado disso é que, quando um nervo é afetado, a ação reflexa, se não remove a irritação no primeiro momento, vai sendo alterada até que a irritação seja removida e a ação, então, cesse. Uma vez que todos os processos vitais se tornam mais fáceis com a repetição, em se repetindo o estímulo irritante, os movimentos que não levaram à eliminação do estímulo podem ou não se repetir; mas os que o removem certamente se repetirão, porque a irritação continua até que justamente esses movimentos sejam feitos. Estes últimos movimentos, portanto, se repetem sempre, tornando-se mais prováveis por causa da repetição mais frequente. Tornam-se, após algum tempo, o modo habitual de eliminar a irritação: um hábito.

[22] An abstraction is a substance whose being consists in the truth of some proposition concerning a more primary substance.
By a primary substance I mean a substance whose being is independent of what may be true of anything else. Whether there is any primary substance in this sense or not we may leave the metaphysicians to wrangle about. By a *more* primary substance I mean one whose being does not depend upon all that the being of the less primary substance [does] but only upon a part thereof.

[23] When a group of nerves are stimulated, the ganglions with which the group is most intimately connected on the whole are thrown into an active state, which in turn usually occasions movements of the body. (CP 3.156, 1880)

Analisemos a relação entre hábitos e interpretantes, principalmente um texto sobre interpretantes lógicos (CP 5.470–493, 1907). Peirce inicia lembrando que todo homem habita dois mundos, o interno e o externo; as diferenças entre eles sendo:

> . . . o Mundo Interno exerce uma compulsão comparativamente leve sobre nós, embora possamos mudá-lo enormemente através de esforços diretos tão suaves a ponto de serem difíceis de notar, criando e destruindo objetos existentes nele; ao passo que o outro mundo, o Mundo Externo, é pleno de compulsões irresistíveis sobre nós, e não podemos modificá-lo nem um pouco, exceto através de um tipo peculiar de esforço, o esforço muscular, e ainda assim muito ligeiramente dessa maneira.[24] (CP 5.474, 1907)

Peirce prossegue, perguntando-se qual é o significado (*meaning*) de um conceito intelectual. Afirma que isso só pode ser resolvido através do estudo dos interpretantes, ou "proper significant effects", "efeitos significativos próprios". Há três classes gerais:

> O primeiro efeito significativo próprio [ou seja, interpretante] é uma sensação [*feeling*] produzida por ele. Há quase sempre uma sensação que passamos a interpretar como evidência de que compreendemos o efeito próprio [interpretante] do signo, embora o fundamento de verdade nisso seja frequentemente muito tênue. Esse "interpretante emocional", como o chamo, pode importar em muito mais que essa sensação de reconhecimento; e em alguns casos, é o único efeito significativo próprio [interpretante] que o signo produz.[25] (CP 5.475, 1907)

Um concerto musical, exemplifica, é um signo que transmite as ideias musicais do compositor, que são frequentemente uma série de sensações. Prossegue com a segunda classe de interpretante:

> Se um signo produz algum efeito significativo próprio [interpretante] adicional, ele o faz através da mediação do interpretante emocional, e esse efeito adicional vai sempre envolver um esforço. Eu o chamo interpretante energético. O esforço pode ser muscular, como no caso do comando [militar] para descansar armas; mas ele é muito mais usualmente um esforço sobre o Mundo Interior, um esforço mental. Não pode nunca ser o significado de um conceito intelectual, uma vez que é um ato único, [enquanto] tal conceito é de natureza geral. Mas que tipo de afeito adicional pode haver?[26] (CP 5.475, 1907)

[24] . . . the Inner World, exerts a comparatively slight compulsion upon us, though we can by direct efforts so slight as to be hardly noticeable, change it greatly, creating and destroying existent objects in it; while the other world, the Outer World, is full of irresistible compulsions for us, and we cannot modify it in the least, except by one peculiar kind of effort, muscular effort, and but very slightly even in that way.

[25] The first proper significate effect of a sign is a feeling produced by it. There is almost always a feeling which we come to interpret as evidence that we comprehend the proper effect of the sign, although the foundation of truth in this is frequently very slight. This "emotional interpretant," as I call it, may amount to much more than that feeling of recognition; and in some cases, it is the only proper significate effect that the sign produces.

[26] If a sign produces any further proper significate effect, it will do so through the mediation of the emotional interpretant, and such further effect will always involve an effort. I call it the energetic interpretant. The effort may be a muscular one, as it is in the case of the command to ground arms; but it is much more usually an exertion upon the Inner World, a mental effort. It never can be the meaning of an intellectual concept, since it is a single act, [while] such a concept is of a general nature. But what further kind of effect can there be?

No parágrafo seguinte, Peirce dá a esse efeito adicional o nome de "interpretante lógico", antes mesmo de descrever sua natureza. Tinha algo em mente, afirma, que poderia ou não ir além do significado de um conceito geral, estando, de qualquer modo, relacionado a ele. Poderia, certamente, ser um pensamento, um signo mental. E se o for, como deveria, deve ele mesmo ter um interpretante lógico; então não pode ser o interpretante lógico final (*ultimate*, derradeiro) do conceito. No mesmo parágrafo Peirce afirma que "se pode provar que o único efeito mental que pode assim ser produzido e que não é um signo mas é de aplicação geral é uma mudança de hábito; querendo dizer uma modificação das tendências de uma pessoa em relação à ação, resultante de experiências prévias ou de prévios esforços de sua vontade ou atos, ou por um complexo dos dois tipos de causa"[27] (CP 5.476, 1907). O interpretante lógico derradeiro é, então, uma mudança de hábito.

O texto prossegue apresentando características dos hábitos e suas mudanças, que a seguir resumimos antes de prosseguir:

- Hábitos têm graus de força, da completa dissociação à associação inseparável. A mudança de hábito frequentemente consiste em aumentar ou diminuir sua força.
- Hábitos apresentam também persistência; mudanças de hábito duram até que o tempo ou outra causa mais definida produza novas mudanças de hábito. E a repetição das ações que produzem mudanças de hábito incrementam as mudanças, como seria de se esperar. Mesmo assim, a mudança de hábito pode ocorrer sem que a ação de mudança se repita; e, alguns casos, a repetição puramente mental é suficiente para mudar um hábito.
- Três classes de evento causam mudanças de hábito:
  - Eventos alheios à mente em que a mudança de hábito se dá: "a surpresa é muito eficiente para quebrar associações de ideias"[28] (CP 5.478, 1907). Não obstante, nenhum hábito novo pode ser criado a partir dessas experiências involuntárias. Mas "cada nova instância da experiência que suporta uma indução fortalece a associação de ideias — esse hábito interno — na qual consiste a tendência a acreditar na conclusão indutiva"[29] (CP 5.478, 1907). Há, portanto, hábitos internos à mente que afetam as mudanças de hábito.
  - Em segundo lugar, o evento que causa a mudança de hábito parece ser um esforço muscular; Peirce cita o esforço consciente de pronunciar corretamente algumas

[27] It can be proved that the only mental effect that can be so produced and that is not a sign but is of a general application is a *habit-change*; meaning by a habit-change a modification of a person's tendencies toward action, resulting from previous experiences or from previous exertions of his will or acts, or from a complexus of both kinds of cause.

[28] Thus, surprise is very efficient in breaking up associations of ideas.

[29] On the other hand, each new instance that is brought to the experience that supports an induction goes to strengthen that association of ideas — that inward habit — in which the tendency to believe in the inductive conclusion consists.

palavras, através da repetição da enunciação correta. E prossegue dizendo duvidar que algo como um conceito possa ser adquirido através somente da prática muscular: quando parece ser assim, "não é a ação muscular mas os efeitos internos que a acompanham, os atos da imaginação, que produzem o hábito"[30] (CP 5.479, 1907). Conclui afirmando que não é a repetição muscular em si que altera o hábito, pois o esforço interno da imaginação é suficiente para tanto.

– A aquisição de conceitos mereceu de Peirce um detalhamento maior, escolha que repetiremos.

Aquisição de conceitos

"Todo conceito, sem dúvida, primeiro surge quando sobre um forte — mas mais ou menos vago — senso de necessidade se sobrepõe a alguma experiência involuntária de natureza sugestiva; aquele modo sugestivo de ser que tem uma certa relação oculta com a estrutura da mente"[31] (CP 5.480, 1907). Essa frase, curiosamente, descreve seu próprio efeito quando lida pela primeira vez. É bastante sugestiva de como Peirce pensa a estrutura da mente: há um senso de necessidade ao qual se sobrepõe uma experiência involuntária. Mas não uma experiência qualquer: uma experiência sugestiva que tem relação com a estrutura da mente. Prosseguindo no mesmo parágrafo, Peirce assume que tal ocorre também com as ideias instintivas dos animais, que entretanto não progridem. Com pessoas, esses primeiros conceitos (primeiros em ordem de desenvolvimento, mas ocorrendo em todos os estágios da vida mental) tomam a forma de conjeturas, embora não sejam reconhecidos como tais. E vai além:

> Todo conceito, toda proposição geral do grande edifício da ciência nos veio primeiro como conjetura. Essas ideias são os *primeiros interpretantes lógicos* dos fenômenos que as sugerem, e que, por sugeri-las, são signos dos quais elas são os interpretantes (conjeturais de fato). Mas que elas não sejam mais do que isso é evidentemente um pensamento posterior, a pitada de dúvida fria que desperta o julgamento sensato do pensador. Enquanto isso, não se esqueça de que toda conjetura é equivalente a, ou expressiva de, um hábito tal como se ter um certo desejo e talvez realizá-lo se se pudesse levar a cabo uma certa ação. Assim, ao homem primitivo deve ter sido perguntado por seu filho se o sol que levantou de manhã era o mesmo que se pôs na tarde anterior; e ele pode ter replicado, "eu não sei, meu menino; mas penso que se pudesse colocar minha marca no sol da tarde, seria capaz de vê-la no sol da manhã novamente; e uma vez conheci um velho que conseguia olhar para o sol embora não pudesse ver quase nada além; e ele me disse que uma vez viu uma mancha com um formato peculiar no sol; e que inequivocamente a reconheceu por vários dias". Prontidão para agir de certo modo sob certas circunstâncias e quando impulsionado por

[30] [...] it is not the muscular action but the accompanying inward efforts, the acts of imagination, that produce the habit.

[31] Every concept, doubtless, first arises when upon a strong, but more or less vague, sense of need is superinduced some involuntary experience of a suggestive nature; that being suggestive which has a certain occult relation to the build of the mind.

> um dado motivo é um hábito; e um hábito deliberado, ou autocontrolado, é precisamente uma crença.[32] (CP 5.480, 1907)

Depreende-se do parágrafo como um todo que existem ideias instintivas, interpretantes lógicos primeiros de fenômenos observados; essas ideias instintivas os animais também podem ter. Em humanos, elas progridem — esse processo será detalhado mais adiante. Aqui, o exemplo no parágrafo ilustra o processo de formação dos conceitos. Há uma conjetura: o sol que levantou de manhã é o mesmo que se pôs na tarde anterior. Uma conjetura que deriva do próprio ciclo diário do sol. Essa conjetura expressa um vago, mas forte, senso de necessidade, que equivale a: se pudesse colocar minha marca no sol da tarde, a veria no sol da manhã novamente. Uma experiência involuntária, mas sugestiva, sugere a formação do conceito: um velho não vê nada além do sol, e nele, certa feita, viu marcas que se mantiveram iguais durante vários dias. Portanto, aparentemente o sol que se põe à tarde é, de fato, o mesmo do dia seguinte. Um conceito, uma primeira ideia que pode ser desenvolvida. Peirce continua nos parágrafos seguintes:

> No próximo passo do pensamento esses primeiros interpretantes lógicos nos estimulam a várias performances no mundo interior. Imagina-mo-nos em várias situações e animados por vários motivos; e prosseguimos a traçar as linhas alternativas de conduta que as conjeturas nos abriram. Somos, além disso, levados, pela mesma atividade interna, a observar diferentes modos pelos quais nossas conjeturas poderiam ser ligeiramente modificadas. O interpretante lógico precisa, portanto, estar num momento relativamente futuro.
>
> A isso pode se acrescentar a consideração de que não são todos os signos que têm interpretantes lógicos, mas apenas os conceitos intelectuais e afins; e esses são todos ou gerais ou intimamente conectados a gerais, assim me parece. Isso mostra que o modo do tempo futuro do interpretante lógico é o condicional, o "seria". [33] (CP 5.481–482, 1907)

[32] Every concept, every general proposition of the great edifice of science, first came to us as a conjecture. These ideas are the *first logical interpretants* of the phenomena that suggest them, and which, as suggesting them, are signs, of which they are the (really conjectural) interpretants. But that they are no more than that is evidently an after-thought, the dash of cold doubt that awakens the sane judgment of the muser. Meantime, do not forget that every conjecture is equivalent to, or is expressive of, such a habit that having a certain desire one might accomplish it if one could perform a certain act. Thus, the primitive man must have been sometimes asked by his son whether the sun that rose in the morning was the same as the one that set the previous evening; and he may have replied, "I do not know, my boy; but I think that if I could put my brand on the evening sun, I should be able to see it on the morning sun again; and I once knew an old man who could look at the sun though he could hardly see anything else; and he told me that he had once seen a peculiarly shaped spot on the sun; and that it was to be recognized quite unmistakably for several days." [Readiness] to act in a certain way under given circumstances and when actuated by a given motive is a habit; and a deliberate, or self-controlled, habit is precisely a belief.

[33] In the next step of thought, those first logical interpretants stimulate us to various voluntary performances in the inner world. We imagine ourselves in various situations and animated by various motives; and we proceed to trace out the alternative lines of conduct which the conjectures would leave open to us. We are, moreover, led, by the same inward activity, to remark different ways in which our conjectures could be slightly modified. The logical interpretant must, therefore, be in a relatively future tense.
To this may be added the consideration that it is not all signs that have logical interpretants, but only intellectual concepts and the like; and these are all either general or intimately connected with generals, as it seems to me. This shows that the species of future tense of the logical interpretant is that of the conditional mood, the "would-be."

Aqui insinua-se a ideia de que, de fato, a criação de novos conceitos decorre de um equilíbrio interno de forças; e que o desenvolvimento desses conceitos depende da proposição e modificação de diagramas mentais que se aproximam, sucessivamente, desse novo conceito que, uma vez admitido como verdadeiro, leva à mudança de hábito. Essa aproximação sucessiva não se dá por acaso. Como já vimos, ela depende do autocontrole, do desejo de se chegar ao conceito buscado. O papel do desejo no processo de inferência aparece também em outro texto de Peirce, "The First Rule of Logic" ("A primeira regra da lógica"), de 1898 (PEIRCE, 1998, p. 42–56). Nele, o autor ressalta que o raciocínio tende a se corrigir com o tempo, expondo exemplo de como isso ocorre nas diversas formas de inferência (que no texto em questão são dedução, indução e retrodução, o nome que à época dava à abdução). A primeira, e num certo sentido a única, regra da lógica seria o desejo de aprender, a vontade de conhecer a verdade. Esse desejo garante que a investigação prossiga, se autocorrigindo à medida em que se aproxima da verdade. E um corolário dessa regra da lógica é a famosa máxima: "não bloqueie o caminho da investigação"[34] (PEIRCE, 1998, p. 48).

Essa importância do desejo no processo de investigação é exemplificado no texto sobre interpretantes lógicos que estamos analisando. Ele continua com um exemplo que não reproduziremos inteiramente aqui, mas que evidencia essas forças em ação na busca por um conceito geral: no exemplo, Peirce descreve uma pessoa que, querendo melhorar um seu invento, precisa encontrar uma solução geral para o problema da coloração de mapas em um bitoro[35], um problema que não tem solução conhecida até hoje. Nesse exemplo o texto peirciano descreve as diversas ações que teriam lugar na tentativa de encontrar a solução; mas chama a atenção que todo o esforço está subordinado a um desejo de ação sobre o mundo: a melhoria do invento. É isso que está por trás do trecho que diz "o intérprete terá formado o hábito de agir de certa maneira quando quer que deseje um certo tipo de resultado" no trecho que reproduzimos a seguir, e que descreve em linhas gerais essa jornada de descoberta:

> Em todos os casos, após algum preâmbulo, a atividade toma a forma de experimentação no mundo interior; e a conclusão (se se chega a uma conclusão definida) é que sob dadas condições o intérprete terá formado o hábito de agir de certa maneira quando quer que deseje um certo tipo de resultado. A conclusão lógica e real é esse hábito; a formulação verbal apenas o expressa. Não nego que um conceito, proposição ou argumento possa ser um interpretante lógico. Apenas insisto que não pode ser o interpretante lógico final, pela razão de que [o conceito, proposição ou argumento] é ele próprio um signo do tipo que tem um interpretante lógico. O hábito em si, embora possa ser um signo de algum outro modo, não é um signo no mesmo modo no qual o signo do qual é um interpretante lógico é signo. O hábito conjugado com o motivo e as condições tem a ação como seu interpretante energético; mas ação não pode ser um interpretante lógico, porque lhe falta generalidade. O conceito que é um interpretante lógico o é apenas imperfeitamente. Ele partilha algo da natureza de

[34] Do not block the way of inquiry.

[35] Uma superfície que se assemelha à de duas rosquinhas coladas lado a lado, de modo que vistas de cima se parecem com o símbolo do infinito (∞).

> uma definição verbal, e é tão inferior ao hábito, e do mesmo modo, quanto uma definição verbal é inferior à definição real. O hábito formado deliberadamente, auto-analisado — auto-analisado porque formado com ajuda da análise dos exercícios que o sustentaram — é a definição viva, o verdadeiro e final interpretante lógico. Consequentemente, a mais perfeita descrição, transmissível por palavras, de um conceito consistirá numa descrição do hábito que o conceito é calculado a produzir. Mas de que outra forma um hábito pode ser descrito além da descrição do tipo de ação à qual dá origem, com a especificação das condições e do motivo?[36] (CP 5.491, 1907)

Fica claro, então, que a aquisição de novos conceitos decorre de um equilíbrio de forças que envolvem um forte (porém vago) desejo, um fenômeno que desperta a criação de um novo conceito, um conjunto de experimentos mentais que frequentemente envolvem experimentação sobre o mundo físico; um processo cuja derradeira consequência é uma mudança de hábitos.

3.5 Agentes interagindo com o entorno

A interação de um agente qualquer com o mundo pode ser extraordinariamente complexa, especialmente se tomarmos agentes compostos por vários indivíduos: um formigueiro ou uma organização social, por exemplo. Uma maneira de compreender essa interação é levar em conta a biossemiótica. Para isso, vamos introduzir o trabalho de Uexküll na Seção 3.5.1; em seguida, faremos uma necessária análise do raciocínio instintivo em Peirce, na Seção 3.5.2.

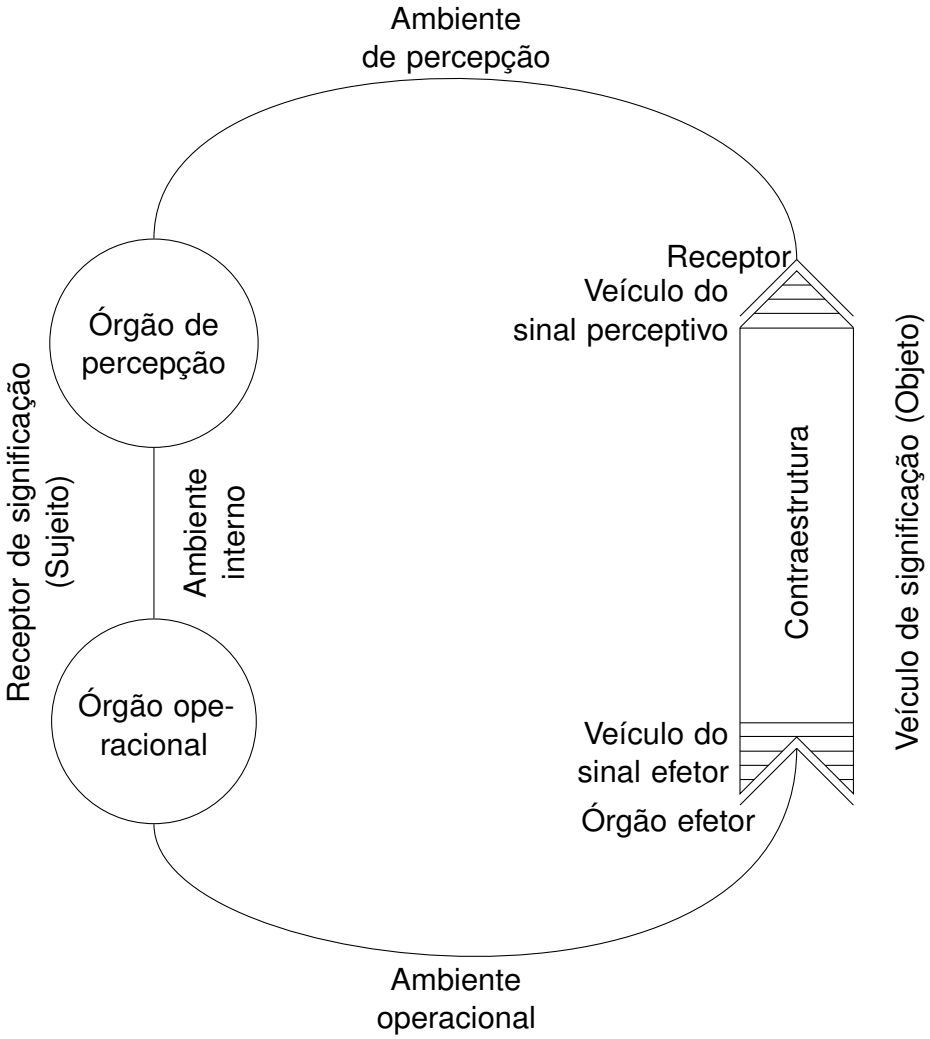
3.5.1 Biossemiótica e Uexküll

A biossemiótica é um assunto extenso, a respeito do qual Nöth (2019) faz um apanhado geral que orienta o texto a seguir. Uexküll adotou o termo *Umwelt* (UEXKÜLL, 1982) como conceito-chave de sua biologia teórica, cuja tradução literal é "meio ambiente". Essa tradução, entretanto, carrega hoje em dia um viés que convém evitar. *Welt*, lembra-nos Nöth, significa "mundo", sendo Umwelt o "mundo circundante".

[36] In every case, after some preliminaries, the activity takes the form of experimentation in the inner world; and the conclusion (if it comes to a definite conclusion), is that under given conditions, the interpreter will have formed the habit of acting in a given way whenever he may desire a given kind of result. The real and living logical conclusion is that habit; the verbal formulation merely expresses it. I do not deny that a concept, proposition, or argument may be a logical interpretant. I only insist that it cannot be the final logical interpretant, for the reason that it is itself a sign of that very kind that has itself a logical interpretant. The habit alone, which though it may be a sign in some other way, is not a sign in that way in which that sign of which it is the logical interpretant is the sign. The habit conjoined with the motive and the conditions has the action for its energetic interpretant; but action cannot be a logical interpretant, because it lacks generality. The concept which is a logical interpretant is only imperfectly so. It somewhat partakes of the nature of a verbal definition, and is as inferior to the habit, and much in the same way, as a verbal definition is inferior to the real definition. The deliberately formed, self-analyzing habit — self-analyzing because formed by the aid of analysis of the exercises that nourished it — is the living definition, the veritable and final logical interpretant. Consequently, the most perfect account of a concept that words can convey will consist in a description of the habit which that concept is calculated to produce. But how otherwise can a habit be described than by a description of the kind of action to which it gives rise, with the specification of the conditions and of the motive?

O Umwelt de um organismo não se constitui de fatos objetivamente dados, mas é marcado pela estrutura interna, ou *ambiente interno*, do organismo. O organismo e seu ambiente formam uma mesma unidade, que é composta de um *ambiente de percepção* e de um *ambiente operacional*, que formam o que Uexküll chamou de *círculo funcional*, representado na Figura 2. O ambiente de percepção se limita a percepções necessárias à sobrevivência do organismo, o ambiente operacional está restrito às operações necessárias para sua a sobrevivência. O caule de uma margarida não é para uma formiga o mesmo que para uma vaca. Não é que ambas as espécies percebem o mesmo objeto de modos diferentes. No mundo das formigas, e no mundo das vacas, não existe o objeto que chamamos "caule de margarida". Os próprios conceitos de "objeto" e "significado" estão em xeque: o modo como a mente de um ser vivo funciona está atrelado ao seu nicho ecológico, pela própria construção do corpo da espécie através da evolução natural.

Figura 2 – O círculo funcional de J. von Uexküll, conforme Nöth (2019).



Um exemplo do próprio Uexküll: o carrapato. As interações do pequeno parasita com os mamíferos são fragmentadas: o carrapato prende-se a um arbusto; o cheiro típico dos mamíferos provoca nele a reação de separar as patas, o que o faz desprender-se do arbusto e, com sorte, cair

sobre seu futuro hospedeiro. Uma vez lá, o calor do animal o atrai, fazendo com que se desloque por entre os pelos até chegar à pele. O cheiro do mamífero provoca uma reação, o calor, outra reação totalmente desconexa da primeira. Infere-se que a mente do carrapato pode prescindir do conceito de um hospedeiro como um objeto único. Não há razão para que o carrapato reúna esses conjuntos díspares de interações numa só entidade; o que chamamos de mamífero, dentro de nosso Umwelt, provavelmente corresponde a dois objetos diferentes na mente do carrapato.

A visão mostrada na Figura 2 pode dar margem a uma interpretação que convém evitarmos: a ideia de que a cognição em si pressupõe uma relação sujeito-objeto. Uma razão para adotarmos o esquema de Uexküll é que ele delimita com propriedade as relações entre a mente e o mundo através de um corpo. Talvez por destacar esses dois extremos fiquemos com a impressão de que neste diagrama a ação mental pressupõe a relação sujeito-objeto.

O circulo funcional é um conceito que complementa e completa as ideias que Peirce desenvolveu a respeito do raciocínio instintivo, que exporemos a seguir.

3.5.2 Percepção, instinto e abdução: o raciocínio instintivo

Peirce debruçou-se sobre os assuntos da percepção, abdução e raciocínio instintivo em diversas ocasiões. Nossa análise baseia-se principalmente no texto intitulado "Instinct and Abduction" ("Instinto e Abdução") (CP 5.171–174, 1903), e o texto "Universes and Predicaments" ("Universos e Predicamentos") (CP 4.539–551, 1906), parte dos "Prolegomena to an Apology for Pragamaticism" ("Prolegômenos para uma Apologia do Pragmaticismo") (CP 4.530–572, 1906).[37]

Em "Instinto e Abdução", Peirce (cf. CP 5.171-174, 1903) se pergunta se as hipóteses explanatórias, resultados da abdução, seriam formadas por acaso. Refuta essa ideia em favor da ideia de que nossa capacidade de gerar hipóteses resulta da evolução. Lembrando-nos que todo o conhecimento que temos sobre a natureza foi, em algum momento, uma mera hipótese abdutiva, Peirce argumenta:

> Qualquer que seja o modo pelo qual o Homem adquiriu suas faculdades divinatórias sobre os caminhos da Natureza, certamente não foi através da lógica autocontrolada e crítica. Mesmo agora ele não é capaz de dar nenhuma razão exata para seus melhores palpites. Parece-me que a afirmação mais clara que podemos fazer sobre a situação do ponto de vista lógico — a mais livre de qualquer mistura questionável — é dizer que o Homem tem uma certa Vidência, não forte o suficiente para estar certa com mais frequência que errada mas forte o suficiente para não estar errada esmagadoramente mais vezes que certa, [uma Vidência] sobre a Terceiridade, os elementos gerais, da Natureza. Uma Vidência, eu a chamo, porque é para ser remetida à mesma classe geral de

[37] Há outro texto em que Peirce detalhou melhor o processo da percepção, introduzindo os conceitos de *percipuum*, *antecipuum* e *ponecipuum* (cf. CP 7.642–681, 1903). Tal detalhamento não é necessário para nossa discussão.

> operações à qual pertencem os Julgamentos Perceptivos. Essa Faculdade é ao mesmo tempo da natureza geral do Instinto, lembrando os instintos dos animais na imensa superação dos poderes gerais da nossa razão e por nos direcionar como se possuíssemos fatos que estão inteiramente fora do alcance dos nossos sentidos. Lembra o instinto também em sua pequena sujeição ao erro; pois embora erre mais frequentemente do que acerte, ainda assim a frequência relativa na qual está certa é, no todo, a coisa mais maravilhosa em nossa constituição.[38] (CP 5.173, 1903)

Ou seja, se fossem obra do acaso, as hipóteses abdutivas deveriam ser esmagadoramente mais erradas do que certas. Sabemos que são mais erradas do que certas, mas, ainda assim, Peirce se maravilha com a grande proporção de hipóteses corretas que nos surgem. Note-se também que, no texto, Peirce coloca esse poder instintivo de gerar hipóteses explanatórias na mesma categoria dos julgamentos perceptivos.

No trecho a seguir, parte dos "Prolegômenos para uma Apologia do Pragmaticismo" (CP 4.530–4.572, 1906), vemos que para Peirce o pensamento natural ou instintivo é importante para a percepção. Convém antes esclarecer que esse trecho é precedido de um parágrafo explicativo dos termos "sema", "fema" e "deloma", que de certo modo correspondem, respectivamente, aos elementos do trio lógico "termo", "proposição" e "argumento", sendo entretanto mais abrangentes do que estes. Sema é "qualquer coisa que sirva a qualquer propósito como substituto de um objeto do qual seja, em algum sentido, um representante ou Signo"[39] (CP 4.538, 1906). Fema é "um signo que é equivalente a uma sentença gramatical, quer interrogativa, imperativa ou assertiva", desde que tenha um efeito compulsivo sobre seu intérprete[40] (CP 4.538, 1906). Deloma é equivalente ao que entendemos por Argumento. Vejamos agora a relação entre instinto e percepção:

> O Objeto Imediato de todo conhecimento e todo pensamento é, em última análise, o Percepto. Essa doutrina de modo algum conflita com o Pragmaticismo, que assegura que o correto Interpretante Imediato[41] de todo pensamento é a

[38] However man may have acquired his faculty of divining the ways of Nature, it has certainly not been by a self-controlled and critical logic. Even now he cannot give any exact reason for his best guesses. It appears to me that the clearest statement we can make of the logical situation — the freest from all questionable admixture — is to say that man has a certain Insight, not strong enough to be oftener right than wrong, but strong enough not to be overwhelmingly more often wrong than right, into the Thirdnesses, the general elements, of Nature. An Insight, I call it, because it is to be referred to the same general class of operations to which Perceptive Judgments belong. This Faculty is at the same time of the general nature of Instinct, resembling the instincts of the animals in its so far surpassing the general powers of our reason and for its directing us as if we were in possession of facts that are entirely beyond the reach of our senses. It resembles instinct too in its small liability to error; for though it goes wrong oftener than right, yet the relative frequency with which it is right is on the whole the most wonderful thing in our constitution.

[39] [...] anything which serves for any purpose as a substitute for an object of which it is, in some sense, a representative or Sign.

[40] By a *Pheme* I mean a Sign which is equivalent to a grammatical sentence, whether it be Interrogative, Imperative, or Assertory. In any case, such a Sign is intended to have some sort of compulsive effect on the Interpreter of it.

[41] Relembrando brevemente o que vimos no capítulo 2, o Interpretante Imediato é o efeito não analisado, a qualidade da impressão que um signo é apto a produzir; o Interpretante Dinâmico é o efeito que efetivamente

> Conduta. Nada é mais indispensável para uma epistemologia sólida do que uma clara distinção entre o Objeto e o Interpretante do conhecimento; [...]. Que somos conscientes dos nossos Perceptos é uma teoria que me parece estar além de disputa; mas isso não é um fato de Percepção Imediata. Um fato da Percepção Imediata não é um Percepto, nem parte de um Percepto; um Percepto é um Sema, enquanto um fato da Percepção Imediata ou ainda o Julgamento Perceptivo do qual tal fato é o Interpretante Imediato é um Fema que é o Interpretante Dinâmico direto do Percepto, e do qual o Percepto é o Objeto Dinâmico, e que é com considerável dificuldade (como a teoria da psicologia mostra) distinguido do Objeto Imediato, embora a distinção seja altamente significativa. Mas para não interromper nossa linha de pensamento, vamos notar que embora o Objeto Imediato de um Percepto seja excessivamente vago, o pensamento natural compensa essa falta (de modo quase equivalente) como segue. Um Interpretante Dinâmico tardio de todo o complexo de Perceptos é o Sema de um Universo Perceptual que é representado no pensamento instintivo como determinando o Objeto Imediato original de cada Percepto. Evidentemente, preciso ser entendido como falando não de psicologia, mas da lógica das operações mentais. Interpretantes subsequentes fornecem novos Semas de Universos resultantes de várias adjunções ao universo perceptual. Eles são, de qualquer modo, todos eles, Interpretantes de Perceptos.[42] (CP 4.539, 1906)

Aqui Peirce articula diversos conceitos importantes. O papel do pensamento instintivo, ou pensamento natural, é central nessa articulação: ele é o responsável por preencher a vagueza do objeto imediato de um percepto. A que vagueza Peirce se refere? O percepto, sendo um sema, determina apenas possibilidade. Aí está a vagueza. Já o julgamento perceptivo é um fema. Trata-se de um distinção extremamente perspicaz: a percepção, livre de qualquer raciocínio sobre ela, não traz em si nenhuma asserção sobre o que percebemos. Apenas percebemos. Esse é o percepto. Há, entretanto, um pensamento natural que transforma essa percepção em asserção — mais precisamente, em um fema. Peirce assim permite a distinção entre a observação irrefletida de, digamos, uma forma avermelhada e uma forma amarelada, das asserções: a cortina é vermelha e a parede é amarela. E reconhece que para passar da observação à asserção está pressuposto um universo perceptual, um sema, que nos situará: o mundo, um quadro, um filme,

produz em alguma mente; e o Interpretante Final não é o efeito do signo sobre alguma mente, mas o efeito que provocaria se a consideração a seu respeito fosse tão longe a ponto de atingir seu derradeiro efeito (cf. CP 8.184, CP 8.314–315, 1909).

[42] The Immediate Object of all knowledge and all thought is, in the last analysis, the Percept. This doctrine in no wise conflicts with Pragmaticism, which holds that the Immediate Interpretant of all thought proper is Conduct. Nothing is more indispensable to a sound epistemology than a crystal-clear discrimination between the Object and the Interpretant of knowledge; [...]. That we are conscious of our Percepts is a theory that seems to me to be beyond dispute; but it is not a fact of Immediate Perception. A fact of Immediate Perception is not a Percept, nor any part of a Percept; a Percept is a Seme, while a fact of Immediate Perception or rather the Perceptual Judgment of which such fact is the Immediate Interpretant, is a Pheme that is the direct Dynamical Interpretant of the Percept, and of which the Percept is the Dynamical Object, and is with some considerable difficulty (as the history of psychology shows), distinguished from the Immediate Object, though the distinction is highly significant. But not to interrupt our train of thought, let us go on to note that while the Immediate Object of a Percept is excessively vague, yet natural thought makes up for that lack (as it almost amounts to), as follows. A late Dynamical Interpretant of the whole complex of Percepts is the Seme of a Perceptual Universe that is represented in instinctive thought as determining the original Immediate Object of every Percept. Of course, I must be understood as talking not psychology, but the logic of mental operations. Subsequent Interpretants furnish new Semes of Universes resulting from various adjunctions to the Perceptual Universe. They are, however, all of them, Interpretants of Percepts.

o que seja, donde podemos extrair do observado a conclusão de que essa forma vermelha é uma cortina, mesmo não sendo, de fato, uma cortina, mas (digamos) a imagem de uma cortina pintada num quadro. E como vimos no trecho anterior (p. 60), o julgamento perceptivo pertence à mesma classe de operações da abdução.

Essa articulação de conceitos para a formação do julgamento perceptivo guarda alguns problemas que Peirce aponta nos parágrafos subsequentes:

> Dito isso, voltemos e perguntemos: como é que o Percepto, que é um Sema, tem como Interpretante Dinâmico direto o Julgamento Perceptivo, que é um Fema? Isso certamente não é o usual dos Semas. [...] Meu raciocínio, então, breve e resumidamente, é que seria *ilógico* para um Ícone puro ter um Fema por Interpretante, e sustento ser impossível para pensamentos não sujeitos ao autocontrole, caso manifesto do Julgamento Perceptual, serem ilógicos. [...] Mas, embora um Interpretante não seja necessariamente uma Conclusão, ainda assim uma Conclusão é necessariamente um Interpretante. De forma que se um Interpretante não está sujeito às regras das Conclusões não há nada monstruoso em eu pensar que esteja sujeito a alguma generalização dessas regras. Para qualquer evolução do pensamento, quer leve a uma Conclusão ou não, há um certo curso normal [...]; e embora concorde inteiramente, em oposição a distintos lógicos, que a normalidade não pode ser critério para o que chamo raciocínio racionalista, o único admissível em ciência, ainda assim [a normalidade] é exatamente o critério do raciocínio instintivo, ou senso-comum, o qual, em seu próprio campo, é muito mais digno de confiança que o raciocínio racionalista. Em minha opinião, é o autocontrole que torna possível qualquer curso de pensamento que não o normal, assim como nada mais [além do autocontrole] torna possível qualquer curso de ação que não o normal; e assim como é precisamente ele [o autocontrole] que dá espaço para um dever-ser de conduta, ou Moralidade, ele igualmente dá espaço para um dever-ser de pensamento, ou Raciocínio Correto; e onde não há autocontrole, nada além do normal é possível.[43] (CP 4.540, 1906)

Peirce aqui introduz as diferenças entre o que chamou de "raciocínio racionalista", próprio do pensamento científico, e o raciocínio instintivo. O raciocínio instintivo, o senso-comum, além de ser mais digno de confiança em seu próprio campo, não segue exatamente as regras das conclusões, mas uma generalização delas. Uma importante diferença entre o raciocínio

[43] That said, let us go back and ask this question: How is it that the Percept, which is a Seme, has for its direct Dynamical Interpretant the Perceptual Judgment, which is a Pheme? For that is not the usual way with Semes, certainly. [...] My reason, then, briefly stated and abridged, is that it would be *illogical* for a pure Icon to have a Pheme for its Interpretant, and I hold it to be impossible for thought not subject to self-control, as a Perceptual Judgment manifestly is not, to be illogical. [...] But though an Interpretant is not necessarily a Conclusion, yet a Conclusion is necessarily an Interpretant. So that if an Interpretant is not subject to the rules of Conclusions there is nothing monstrous in my thinking it is subject to some generalization of such rules. For any evolution of thought, whether it leads to a Conclusion or not, there is a certain normal course [...]; and while I entirely agree, in opposition to distinguished logicians, that normality can be no criterion for what I call rationalistic reasoning, such as alone is admissible in science, yet it is precisely the criterion of instinctive or common-sense reasoning, which, within its own field, is much more trustworthy than rationalistic reasoning. In my opinion, it is self-control which makes any other than the normal course of thought possible, just as nothing else makes any other than the normal course of action possible; and just as it is precisely that that gives room for an ought-to-be of conduct, I mean Morality, so it equally gives room for an ought-to-be of thought, which is Right Reason; and where there is no self-control, nothing but the normal is possible.

racionalista e o senso-comum é o autocontrole, presente no primeiro e ausente no segundo. A ausência de autocontrole faz com que o raciocínio instintivo siga um curso "normal". E lembremos que o caráter de primeiridade do raciocínio abdutivo o habilita a surgir como independente de autocontrole; mas antes de entrar nessas considerações, vejamos que todavia o *insight* abdutivo difere do julgamento perceptivo num importante fundamento, que é justamente o caráter compulsivo do segundo, completamente ausente no primeiro. No parágrafo subsequente:

> Mas supondo que eu esteja certo, como provavelmente estarei na opinião de *alguns* leitores, como se explica então um Julgamento Perceptivo? Como réplica, observo que um Percepto não pode ser ignorado à vontade, mesmo na memória. Muito menos uma pessoa pode evitar de perceber o que, como se diz, está na sua cara. Além disso, há esmagadora evidência de que o observador está ciente dessa compulsão sobre ele; e se não posso dizer com certeza como esse conhecimento lhe chega não é porque não possa conceber como isso lhe vem, mas porque, havendo diversas maneiras pelas quais isso pode ocorrer, é difícil dizer qual dessas maneiras de fato ocorre. Mas essa discussão pertence à psicologia; e não entrarei nela. Basta dizer que o observador está ciente de ser compelido a perceber o que percebe. Agora, existência significa precisamente o exercício da compulsão. Consequentemente, qualquer característica do percepto é aliviada por alguma associação e assim ganha uma posição lógica equivalente à de premissa observacional de uma Abdução explicativa, a atribuição de Existência a ela [a característica do percepto] no Julgamento Perceptivo sendo virtualmente e num sentido estendido, uma Inferência lógica Abdutiva que quase se aproxima da inferência necessária. Mas meu próximo artigo lançará luz sobre a afiliação lógica da Proposição, e do Fema em geral, à coerção.[44] (CP 4.541, 1906)

O trecho exibe as forças que agem para a geração do julgamento perceptivo, mais precisamente seu interpretante dinâmico: temos a compulsão do percepto sobre a mente (sendo o exercício de tal compulsão o próprio significado da existência, segundo o trecho); essa compulsão age sobre a mente, que é aliviada através de uma associação que atribui existência à característica percebida. Tal atribuição é uma "Inferência lógica Abdutiva que quase se aproxima da inferência necessária". Esse mecanismo de compulsão e alívio sobre a mente, subjacente ao mecanismo através do qual o percepto resulta em julgamentos perceptivos num universo perceptual, encontra eco na teoria peirciana da formação e mudança de hábito.

[44] But supposing that I am right, as I probably shall be in the opinions of *some* readers, how then is the Perceptual Judgment to be explained? In reply, I note that a Percept cannot be dismissed at will, even from memory. Much less can a person prevent himself from perceiving that which, as we say, stares him in the face. Moreover, the evidence is overwhelming that the perceiver is aware of this compulsion upon him; and if I cannot say for certain how this knowledge comes to him, it is not that I cannot conceive how it could come to him, but that, there being several ways in which this might happen, it is difficult to say which of those ways actually is followed. But that discussion belongs to psychology; and I will not enter upon it. Suffice it to say that the perceiver is aware of being compelled to perceive what he perceives. Now existence means precisely the exercise of compulsion. Consequently, whatever feature of the percept is brought into relief by some association and thus attains a logical position like that of the observational premiss of an explaining Abduction, the attribution of Existence to it in the Perceptual Judgment is virtually and in an extended sense, a logical Abductive Inference nearly approximating to necessary inference. But my next paper will throw a flood of light upon the logical affiliation of the Proposition, and the Pheme generally, to coercion.

Cabe aqui abrir um parêntese: embora Peirce aparentemente nunca tenha escrito o artigo que cita no parágrafo, não encontraremos dificuldade em encontrar correspondência entre a coerção e a proposição, em particular, e ao fema, em geral. A coerção aparece na proposição de duas formas. A primeira forma decorre do modo como Peirce decompõe a proposição, ou qualquer dicissigno, em um índice — que tem o papel de sujeito proposicional — e um ícone, o predicado da proposição. Ora, o índice só se dá por coerção: é sua direta ligação ao objeto, pela coerção da secundidade, que o torna índice. A segunda forma de coerção da proposição tem a ver com a própria natureza do dicissigno: ele é verdadeiro ou falso, mas não apresenta as razões para tal. Sua ação é, portanto a de compulsão sobre a mente, no sentido de que a esta não resta nada senão registrar a asserção feita pelo dicissigno, sem necessariamente aceitá-la pelo seu valor de face: uma clara reação bruta a uma ação bruta (cf. CP 2.309–314, 1903). Fecha-se o parêntese.

Resumindo os conceitos acima: o percepto traz em si uma vagueza intrínseca que é preenchida pelo julgamento perceptivo; este se dá por meio de um tipo de raciocínio que (1) leva em conta o universo perceptual mas (2) não se submete ao autocontrole do raciocínio "racionalista", sendo, portanto, instintivo. Trata-se do senso-comum, eficiente em sua normalidade. No texto, o raciocínio é construído nas seguintes etapas:

1. O percepto é apresentado como o objeto imediato de todo pensamento e todo conhecimento. Somos conscientes do percepto, que é um sema.
2. Um fato de percepção imediata é um fema; esse fato de percepção imediata é o interpretante dinâmico de um julgamento perceptivo (um signo) cujo objeto dinâmico é o percepto. Isso se dá graças ao "pensamento natural" (ou instintivo), que leva em conta um outro sema, o sema do universo perceptual, para tal. Esse sema do universo perceptual é o interpretante dinâmico tardio do complexo de perceptos que, no pensamento instintivo, determina o objeto imediato original de cada percepto. Ou seja, sendo o sema do percepto uma forma vermelha e o fema a frase "essa cortina é vermelha", a passagem do sema para o fema precisa levar em conta o fato de o observador estar vendo a cortina ao vivo, numa fotografia ou num quadro.
3. Na doutrina peirciana não se pode, do ponto de vista lógico, partir de um sema para chegar a um fema. O primeiro traz em si apenas uma possibilidade, o segundo se impõe, podendo ser uma asserção. Essa passagem do fema a partir do sema é ilógica para Peirce.
    a) Mas o julgamento perceptivo não pode ser ilógico: não é submetido ao autocontrole.
    b) Portanto, é razoável admitir que há interpretantes que não seguem as regras da lógica ("regras da Conclusão", no texto), mas uma generalização dessas regras. Há um curso normal da evolução do pensamento, quer ele leve a uma conclusão (sujeita ao autocontrole) ou não. E o critério de normalidade, embora não seja admissível na ciência, é admissível no raciocínio instintivo ou senso-comum.

   c) É o autocontrole (presente, por exemplo, no raciocínio científico) que possibilita um raciocínio cujo curso não seja o normal. O autocontrole dá espaço tanto para um dever-ser de conduta (moralidade) quanto para um dever-ser de pensamento (raciocínio correto). Onde não há autocontrole (por exemplo, no raciocínio instintivo) apenas a normalidade é possível.

4. Como se dá o julgamento perceptivo?

   a) O percepto exerce uma compulsão sobre a mente; não pode ser ignorado.
   b) Essa compulsão é aliviada por uma associação. Isso coloca o percepto como uma premissa observacional de uma abdução explicativa, que vai gerar essa associação.
   c) Assim, a existência do percepto (ou outro fema a respeito dele) é inferida num julgamento perceptivo: uma inferência lógica abdutiva de um tipo especial, que se aproxima, num certo sentido, de uma inferência necessária.

Ballabio (2018) apresenta também uma explanação para o componente abdutivo da percepção através do raciocínio instintivo; em sua análise diferencia a percepção da experiência. A segunda é produto de uma inferência abdutiva a respeito de mudanças na primeira. Assim, dois eventos que em si não trazem nenhuma asserção a respeito do mundo, quando colocados lado a lado suscitam uma hipótese abdutiva a respeito de como são as coisas.

Essas considerações a respeito do raciocínio instintivo, abdução e percepção complementam a visão de Uexküll, permitindo que se vislumbre como os ambientes de percepção e operacional do círculo funcional atuam sobre a mente na moldura da filosofia peirciana: os órgãos sensórios, embora não sejam capazes de produzir asserções de natureza mental, geram semas que persistem compulsivamente sobre a mente. Esta reage através de uma associação de ideias de natureza abdutiva que parte dos semas e os transforma em femas, que são mais tratáveis, do ponto de vista lógico, pela mente. Essa passagem leva em conta um universo de discurso provavelmente porque essa é uma contextualização útil, sendo preservada na evolução do organismo: graças a esse recurso é possível à mente exercitar sua ação utilizando somente a imaginação, ao invés de experimentar diretamente o mundo.

3.6 Considerações adicionais a respeito da abdução

O raciocínio abdutivo parece guardar em si um paradoxo: embora possa gerar ideias novas sem a necessidade de nenhuma regra e independente de qualquer compulsão, seus resultados não surgem do nada. Pois se surgissem do nada equivaleriam a pensamentos desconectados de quaisquer outros, uma impossibilidade segundo Peirce (cf. CP 5.284, 1868).

Santaella (2004, p. 77–122) detalha a evolução do conceito de raciocínio abdutivo na obra de Peirce e apresenta uma visão extremamente coerente com o todo da obra do filósofo

americano. Com base nesse trabalho, tentaremos aqui apresentar uma conjetura que traz o Umwelt, o círculo funcional de Uexküll e os universos perceptuais como panos de fundo.

A abdução instintiva, que complementa a vagueza do percepto no julgamento perceptivo à luz de um universo perceptual, é uma resposta à persistência do percepto sobre a mente do organismo. Atribuí-la à evolução equivale, de certa forma, a atribuir à evolução o círculo funcional do organismo em seu Umwelt. Essa abdução instintiva não surge do nada: decorre da persistência do percepto sobre hábitos mentais que complementam a vagueza desse mesmo percepto com base em um universo perceptual. A questão que surge é: ao raciocinar sobre algo que não está diretamente ligado ao percepto, ao raciocinar sobre conjuntos infinitos, por exemplo, através do raciocínio diagramático, o modelo peirciano preconiza que algumas hipóteses serão geradas por abdução. Como se dá, então, essa abdução dissociada do percepto?

Santaella traz à tona uma distinção entre "instintos verdadeiramente herdados e os que são adquiridos por convenção, estando ambos, não obstante a diferença, irremediavelmente fora do nosso controle" (SANTAELLA, op. cit., p. 113), tendo ambos o caráter de hábitos. Mais adiante (ibid., p. 121) retoma esse caráter habitual para descrever importantes características da abdução:

> É na noção de hábito inconsciente de inferência que se encontra o ponto de convergência para a compreensão do caráter ao mesmo tempo inferencial e originário da abdução. Embora sejam inferências, por serem inconscientes, elas entram em nosso pensamento como se fossem originárias, primeiras, como se o mundo estivesse começando nelas. (SANTAELLA, 2004, p. 121)

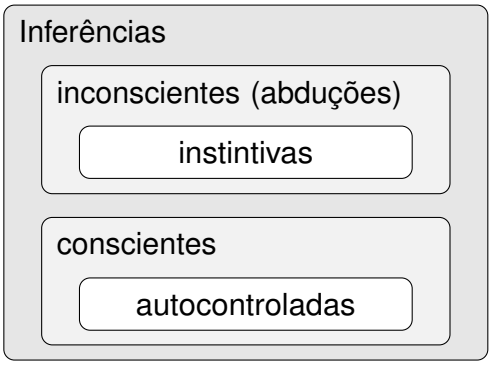
Temos, portanto, um caminho possível para explicar a abdução que não é consequência do percepto: é resultado de um hábito inconsciente adquirido, de certa maneira, um instinto adquirido.

Finalmente, o fato de a abdução não ser esmagadoramente mais errada do que correta — que é o que constitui, de fato, o caráter instintivo desta (p. 104) — é uma faculdade que "a espécie humana desenvolveu provavelmente no curso do crescimento evolutivo de sua constituição física e mental" (p. 106). Santaella cita Peirce:

> 'Não pode haver nenhuma dúvida razoável de que a mente humana, tendo se desenvolvido sob a influência das leis naturais, pensa naturalmente, por essa razão, de um modo similar aos padrões da natureza', afirmou Peirce (CP 7.39 [1907]). Sendo parte da natureza, a mente emergiu do mesmo processo evolutivo que perpassa a biosfera. Há, consequentemente, uma conaturalidade entre a mente e o cosmos, o que significa que o homem tem uma afinidade com a natureza, está em sintonia com ela, e possui uma adaptação natural para imaginar teorias e ideias que traduzem essa sintonia. (SANTAELLA, 2004, p. 106)

A Figura 3 mostra as relações entre as inferências conscientes e inconscientes, levando em conta que algumas inferências inconscientes são instintivas, e que algumas inferências conscientes são autocontroladas.

Figura 3 – Inferências conscientes e inconscientes.



Insinua-se aqui uma relação adicional entre abdução não perceptiva e o círculo funcional de Uexküll. Uma relação que, entre outras coisas, poderia lançar alguma luz sobre como nossa capacidade instintiva de adivinhar as leis da natureza funciona. Vimos (Seção 3.4) que o processo de aprendizado passa por experimentações na imaginação, que são de natureza diagramática (Seção 3.3). Sendo diagramáticas, passam pela construção e observação de diagramas, que são ícones. Nossa conjetura é que tanto a construção de alguns desses diagramas quanto sua observação — ao menos na infância do raciocínio, quando começamos a refletir sobre um tema — guardam semelhança com a forma de ação e atuação do organismo no seu círculo funcional. Ou seja, ao pensar um diagrama a respeito de um problema novo, o ícone que se forma guarda semelhança com elementos que poderiam ser percebidos, ou manipulados, no círculo funcional do agente pensante. Uma evidência a favor dessa conjetura é a dificuldade de raciocinar a respeito de teorias cujos enfoques estão fora do alcance da nossa percepção, como a geometria quadridimensional e a mecânica quântica. Ambas são teorias bem conhecidas, uma puramente matemática, a outra ancorada em experimentos. Apesar de ambas as teorias serem campos de pesquisa em constante evolução, são difíceis de ensinar e de entender.

Embora possa ser verdadeira, essa conjetura se aplica somente aos raciocínios iniciais da investigação; não resta dúvida de que, à medida em que a pesquisa avança, se abre espaço para a formação de diagramas abstratos totalmente desconectados dos perceptos do organismo. Caso contrário, não seríamos capazes de investigar a geometria em quatro dimensões pelo simples fato de não percebermos nenhuma forma quadridimensional.

3.7 Diagramas

Tomemos os conceitos expostos até o momento para delinear os elementos que fazem parte do processo de raciocinar de um modo que possa ser aplicado também aos computadores.

3.7.1 Raciocínio

Para criar uma nova proposição, por dedução, indução ou abdução, um agente precisa ser capaz de construir e observar abstrações e diagramas que guardem uma relação de similaridade com o que quer que esteja sendo pensado. Isso é o que, grosso modo, permite encontrar novas hipóteses por abdução, conclusões necessárias por dedução, ou inferir a validade de alguma proposição por indução. Mas a atuação do raciocínio lógico está subordinada a elementos que não fazem parte da própria lógica. Estamos falando de elementos de natureza ética e estética.

O pensamento controlado permite que novas proposições estejam de acordo, ainda que indiretamente, com algum propósito que é consequência de decisões éticas, por sua vez suportadas por critérios estéticos. Pode-se ver uma ordem de precedência aqui: de princípios estéticos seguem-se regras éticas que definem propósitos tangíveis. Tais propósitos sugerem o que deveria ser pensado e conduzem à construção de abstrações e diagramas apropriados, que levarão a novas proposições através de inferência abdutiva, dedutiva ou indutiva.

Se o agente tiver autonomia suficiente, as proposições às quais ele chega através do raciocínio lógico podem afetar seus hábitos estéticos e éticos. Já agentes com menor autonomia têm hábitos estéticos e éticos, e portanto propósitos, determinados por ações que iniciaram em outros agentes: um mestre, um treinador, um programador.

É importante notar que embora ética e estética sejam pressupostos necessários para o raciocínio, a habilidade de modificar os próprios hábitos éticos e estéticos não é. Mesmo que o agente não seja capaz de mudar seus hábitos éticos ou estéticos, basta possuí-los para que possa propriamente raciocinar.

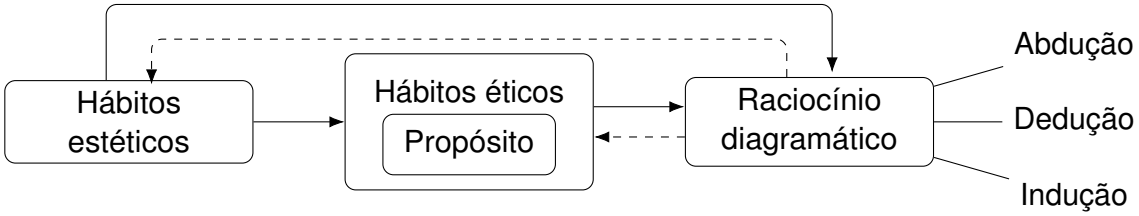
Hábitos éticos incorporam o propósito do raciocínio que está sendo efetuado; tal propósito guia a busca por conclusões. Podemos generalizar e dizer que algum tipo de hábito ético e estético pode ser encontrado em toda ação semiótica, independente do agente. Esses hábitos, no raciocínio humano, se identificam com aquilo que é desejável por si, nos hábitos estéticos, e hábitos de ação — ética. Nas pessoas, a reflexão a respeito do que é desejável por si e a respeito dos princípios desejáveis de conduta pode levar à modificação dos hábitos de conduta, e ao refinamento do conceituação e expressão daquilo que é bom por si.

Como vimos, os objetos de certa forma possuem regras de conduta: são as leis da natureza que determinam seu comportamento. Nos computadores, essas regras se estendem às regras do programa que executam: esses, mais as leis da natureza, são seus hábitos. A habilidade de um agente alterar seus próprios hábitos éticos e estéticos depende do seu grau de autocontrole, e, portanto, de sua autonomia.

A precedência entre ética, estética e lógica pode ser representada no diagrama mostrado

na Figura 4. Nele as setas cheias representam que o raciocínio (que pode ser dedutivo, abdutivo ou indutivo, como se vê no diagrama) é determinado por hábitos estéticos, éticos e por um propósito, que também é determinado pela ética. Num computador, grosseiramente, os hábitos éticos que

Figura 4 – Um diagrama do raciocínio



afetam o raciocínio estão no programa, e o propósito do raciocínio é determinado pelo propósito desejado pelo programador usuário. Tanto o programa quanto o propósito do programador fazem parte da regra de ação, ou hábitos éticos, do computador. Espera-se, naturalmente, que o computador autônomo seja capaz de atuar diretamente nas regras do programa, mas não (diretamente) no propósito do programador, daí o destaque ao propósito no diagrama. As setas pontilhadas mostram que os resultados dos raciocínios lógicos podem, direta ou indiretamente, afetar o modo pelo qual hábitos estéticos, éticos e propósito atuam sobre o raciocínio, desde que o agente possua autonomia para tanto. O papel de cada elemento é bem definido na filosofia de Peirce. O modelo mostra componentes dos agentes semióticos sem entretanto explicitar nem a interação dos agentes com o mundo à sua volta, nem como esses agentes evoluem e aprendem.

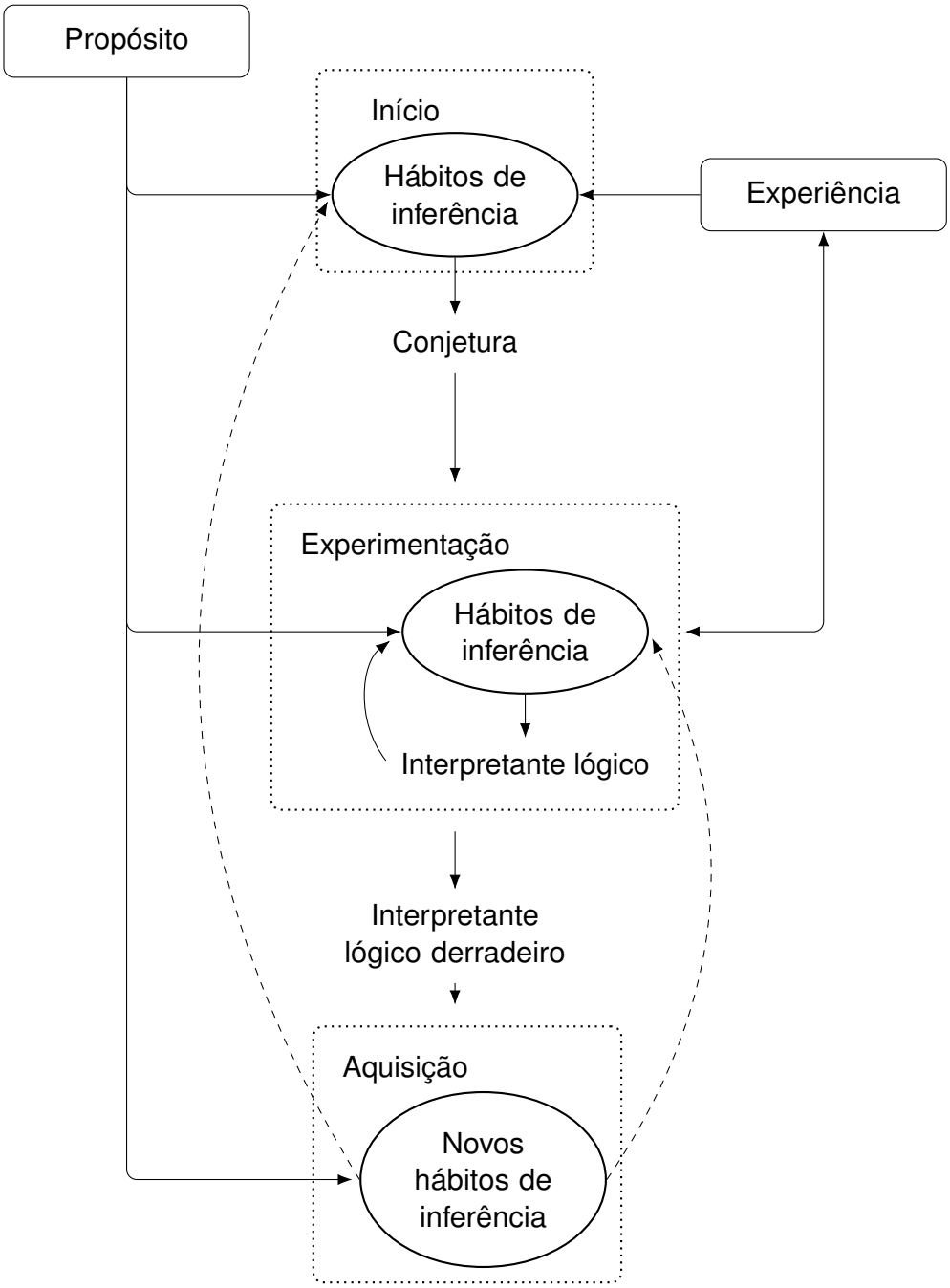
3.7.2 Formação e mudança de hábito

Como vimos, o conceito de aprendizado em Peirce está ligado à ideia de hábito: aprender é mudar de hábito. A Figura 5 ilustra o processo. Nela, vemos os hábitos de inferência em três momentos: quando iniciam, ou seja, é criada uma conjetura; a fase de experimentação, quando são gerados interpretantes lógicos intermediários; e a fase de aquisição do novo conceito, que implica numa mudança dos hábitos; essa mudança é, de fato, o derradeiro interpretante lógico do novo conceito adquirido. Esses novos hábitos então participarão de processos subsequentes. Fica claro como novos hábitos são adquiridos, e como esses hábitos são incorporados aos hábitos éticos do agente, nos casos em que este possui tal autonomia. Como se vê, a experimentação participa desse processo, donde a importância do estudo da maneira pela qual os agentes interagem com seu entorno.

3.8 Outros elementos da filosofia de Peirce

Há muitos outros elementos da filosofia peirciana que não foram detalhados e que podem lançar luz sobre a agência semiótica, inclusive em computadores. Os que apresentamos até aqui — raciocínio diagramático, círculo funcional e aprendizado — constituem um mínimo. A seguir

Figura 5 – A aquisição de conceitos



veremos como se dá, segundo Peirce, a comunicação entre agentes — Seção 3.8.1 — e conceitos essenciais para compreensão das linguagens naturais: indeterminação e vagueza, na Seção 3.8.2.

3.8.1 O modelo peirciano de comunicação

Nöth (2013) nos dá um panorama geral sobre as diferentes teorias semióticas da comunicação humana. No texto o autor conclui que os diversos modelos semióticos de comunicação mostram divergências consideráveis, não sendo sequer possível encontrar um denominador comum entre eles. Ao destacar o modelo de comunicação de Peirce, aponta que, diferente dos

demais — em que os agentes assumem papéis como os de emissor e receptor —, nele a agência também está no próprio signo.

O processo em si, do ponto de vista das pessoas participantes dele, segue a descrição de raciocínio dada acima; sendo a comunicação um tipo especial de raciocínio, envolve diagramas. Vejamos, então, como Peirce descreve o processo de comunicação entre mentes antes de o examinarmos à luz de sua filosofia:

> Signos funcionam principalmente entre duas mentes ou teatros de consciência, dos quais um é o agente que *emite* o signo (seja acusticamente, opticamente ou de outra forma), enquanto o outro é a mente *paciente* que *interpreta* o signo (PEIRCE, 1998, p 403).

Nöth (op. cit.) afirma que a descrição das mentes como "teatros de consciência" é mais que mera metáfora: assim como atores no palco agem em nome de um terceiro agente (o personagem ao qual emprestam corpo e voz) os signos nas mentes envolvidas agem em nome de terceiros: são determinados por seus objetos em processos de semiose nos quais são interpretantes de outros signos, numa recursão que pode ser levada ao infinito. Uma consequência deste modo de ver as coisas é a constatação de que os participantes do processo de comunicação estão restritos nas escolhas que podem fazer dos signos utilizados; lembra-nos o autor que:

> Não é o emissor que decide se a mensagem é certa ou errada, verossímil ou inverossímil, etc. É a realidade representada pelo signo que o faz. Portanto, a realidade determina os signos que a representam. O objeto do signo assim termina sendo um co-agente semiótico na comunicação. (NÖTH, op. cit.)

Percebe-se então que a ação própria do signo explicita um processo de comunicação que é mais complexo do que aquele que preconiza os signos como meros instrumentos dos interlocutores.

Avançando, Nöth também pondera a questão do consenso na comunicação, que para alguns autores é uma meta desejável e para outros, impossível. Informa-nos que Peirce também endereçou essa questão em múltiplas ocasiões. De modo geral, Peirce destaca a importância de haver um conhecimento comum entre os participantes do processo de comunicação, de modo a permitir que os signos utilizados indiquem seus objetos com base na familiaridade que cada participante tem com estes últimos. Segundo Nöth:

> Conhecimento sobre o que o outro quer dizer é sempre fragmentário, e os pedaços de evidência do que se quer dizer são meras "cópias de um rascunho arrancados da vida do outro". (NÖTH, op. cit.)

Assim, os intérpretes podem apenas tentar suplementar esses fragmentos com suas próprias ideias, tentando arranjá-los em seu próprio panorama.

O processo de comunicação

Numa moldura peirciana, o processo de comunicação entre agentes é um processo em que participam dois diagramas mentais: o emissor cria em sua mente um diagrama do estado em que se encontra o receptor, e imagina o estado em que deseja que este se encontre através de sua interlocução. A partir daí, procura os signos que, em sua opinião, serão capazes de levar o receptor ao estado que deseja. Apresenta esses fragmentos de seu teatro mental ao receptor que, por sua vez, os coloca em seu teatro próprio mental, num outro diagrama que inclui o emissor. O receptor tenta preencher o que não sabe com suas próprias ideias; eventualmente formula novas hipóteses, criando novas abstrações. E, no caso de um diálogo, escolhe quais signos levarão o interlocutor ao estado que deseja. Mas, mesmo não se tratando de um diálogo, sendo ambos pessoas, ambos têm familiaridade com o processo que se passa com o outro, e levam isso em conta ao lidar com o que pode ou não ser comum a ambos.

Um ponto importante a observar nessa descrição do processo é que, sob essa moldura, é possível levar em conta a autonomia dos signos que são apresentados na interlocução. Ou seja: o signo não é um instrumento escolhido para expressão do que o emissor pensa. É, antes, um agente autônomo que é apresentado pelo emissor em função do efeito que este acha que o signo pode causar no receptor.

3.8.2 Vagueza e indeterminação em Peirce

Peirce critica os lógicos de sua época por não se dedicarem ao estudo da vagueza (cf. CP 5.446, 1905; CP 5.505, 1905), que para ele é "o análogo antitético da generalidade"[45] (CP 5.505, 1905). Nöth e Santaella (2017) estudaram essa incerteza intrínseca, que se resumiremos tomando dois exemplos que encontramos em Peirce:

> Um signo é objetivamente *geral* na medida em que, deixando sua efetiva interpretação indeterminada, entrega ao intérprete o direito de completar a determinação por si mesmo. "O homem é mortal". "Qual homem?" "Qualquer homem que você queira." Um signo é objetivamente *vago* na medida em que, deixando sua interpretação mais ou menos indeterminada, reserva a algum outro signo ou experiência possível a função de completar a determinação. "Neste mês", diz o oráculo do almanaque, "um grande evento está para acontecer". "Que evento?" "Oh, veremos. O almanaque não diz."[46] (CP 5.505, 1905)

[45] It [vagueness] is the antithetical analogue of generality.

[46] A sign is objectively *general*, in so far as, leaving its effective interpretation indeterminate, it surrenders to the interpreter the right of completing the determination for himself. "Man is mortal." "What man?" "Any man you like." A sign is objectively *vague*, in so far as, leaving its interpretation more or less indeterminate, it reserves for some other possible sign or experience the function of completing the determination. "This month," says the almanac-oracle, "a great event is to happen." "What event?" "Oh, we shall see. The almanac doesn't tell that."

Vemos, portanto, que há dois modos pelos quais os signos não determinam inteiramente sua interpretação. Um, pela generalidade, que é um fenômeno de preponderante terceiridade; outro, pela vagueza da possibilidade, um fenômeno de preponderante primeiridade. Evidentemente, então, para Peirce, a determinação se caracteriza pela secundidade: o signo que aponta seu objeto por uma ligação direta é um signo cuja interpretação é determinada.

Podemos utilizar essa caracterização para signos como números. Num texto a respeito dos números cardinais (CP 4.153–159, 1897), Peirce, querendo escrutinar o significado (ou a falta de significado) de tais números, lembra-nos que o ato de contar os elementos de uma coleção equivale a estabelecer uma relação entre cada elemento da coleção e um elemento de outra coleção que é ordenada, a coleção dos números naturais. E esse pareamento entre elementos se dá de forma aproximadamente mnemônica, pois lembramos a ordem dos símbolos que representam os números naturais, e quando não lembramos, conhecemos as regras para a formação de novos símbolos que representam números.

Tomando a caracterização da determinação, indeterminação e vagueza à luz das categorias, pode-se afirmar que, em certo sentido, signos como “um”, “dois”, “três”, ou com algarismos, “11”, “5”, têm a característica de determinar claramente uma certa posição numa sequência. Por isso afirmamos que, dentro do escopo dos números naturais, o signo “5” é perfeitamente determinado, pois seu interpretante está diretamente ligado a uma posição bem definida no conjunto; o signo “k”, caracterizado pela equação “$k \in \mathbb{N}$” (que quer dizer “k é um número natural”), é vago pois assume a possibilidade de ser qualquer número natural; e o símbolo “m”, caracterizado pela equação “$m = 2k+1, m, k \in \mathbb{N}$” (que quer dizer que m é ímpar) é indeterminado no escopo dos números naturais, ou seja, não é possivelmente qualquer número natural — caso em que seria vago nesse escopo, como o signo “k” —, mas um elemento indeterminado de um conjunto determinado de números naturais.

3.9 Visão resumida

Vamos utilizar os conceitos apresentados neste capítulo para compreender a agência semiótica nos computadores. Por isso, convém elencar os principais pontos apresentados para fins de referência futura.

- Como se dá o raciocínio: os tipos de raciocínio, o papel dos diagramas, a importância das abstrações e como a semiótica se subordina à ética e à estética — ou, como hábitos de raciocínio se subordinam a hábitos de ação (éticos) e a hábitos existentes por si, sem razão (estéticos).
- Como a mente evolui e aprende através da mudança de hábitos.

- Como o agente, enquanto organismo, se relaciona com o meio ambiente, a importância do raciocínio instintivo e sua natureza abdutiva.
- Esclarece particularidades a respeito do raciocínio abdutivo e lança umas conjetura sobre seu funcionamento, a saber, que o raciocínio diagramático, especialmente no seu início, se apoia nas capacidades pre-existentes no organismo.

Para descrever um agente semiótico, como por exemplo um computador, pode-se descrever os seguintes elementos:

1. Círculo funcional e Umwelt: o modo como ele se relaciona com o meio ambiente, e suas capacidades perceptivas e de atuação.
2. Raciocínio diagramático: o modo como hábitos estéticos e éticos afetam o propósito e a inferência.
    - O modo como hábitos éticos afetam o propósito e a inferência pode ser resumida à descrição do modo de ação do agente.

      Em particular, o modo pelo qual hábitos éticos afetam a inferência de uma máquina contém a descrição de seu mecanismo. Mas só isso pode não ser suficiente para compreender seu modo de ação, pois não contempla o propósito da pessoa que a construiu ou que a utiliza.

      A descrição do propósito pode ser mais ou menos vaga, de acordo com a autonomia do agente. Quanto mais autônomo, mais capaz de alterar seus próprios hábitos; portanto, mais capaz de aprender. Quanto maior a autonomia, menos específico pode ser o propósito, especialmente se pensarmos em máquinas como agentes semióticos.
    - A descrição de como hábitos estéticos atuam, que é a descrição dos fatores que afetam hábitos éticos — e portanto o propósito e a inferência — e que não estão subordinados a nenhuma lei nem são compelidos por nada, pode corresponder à descrição de elementos que não têm justificativa para o agente descrito.

      Por exemplo, numa inferência computacional cujo resultado é um programa de computador, o programador pode limitar o número de instruções desse programa resultante a um certo intervalo. A escolha desse intervalo pode depender de razões que nada têm a ver com a conceituação da agência semiótica, pode depender, por exemplo, do poder de processamento do equipamento à disposição no momento. Esse intervalo, determinado pelo programador, seria visto como uma regra que não tem justificativa pelo computador, caso ele tivesse capacidade de refletir sobre isso. Então, embora não possa ser comparada com um critério estético para a inferência humana, trata-se de um hábito que a máquina segue e para o qual não encontraria justificativa se pudesse refletir sobre isso.
    - A descrição de como realiza a inferência:

  - Novas abstrações.
  - Diagramas e como são observados.
  - Inferência propriamente dita:
    * Abdução.
    * Dedução.
    * Indução.

3. O modo como aprende ou muda seus próprios hábitos de inferência, caso tenha essa autonomia.

Nem todos os computadores, vistos como agentes semióticos, apresentam todos os elementos elencados; como vimos, há diferentes capacidades em termos de autonomia, autocontrole e outros fatores. Podemos agora passar a estudar os computadores para podermos, em seguida, fazer a análise deles como agentes semióticos.

4 Computadores

Podemos observar na evolução da palavra “computador” a ideia de crescimento dos signos (cf. NÖTH, 2014). Em 1936, quando Alan Turing (1912-1954) publicou seu famoso artigo “On computable numbers, with an application to the Entscheidungsproblem” (“Sobre os números computáveis, com uma aplicação ao problema da decisão”) (TURING, 1936), era apenas o nome de uma profissão: uma pessoa dedicada a realizar e conferir cálculos matemáticos. Em seu artigo Turing imaginou uma máquina que executava um conjunto finito de operações simples, seguindo uma tabela também finita de “estados”. Turing reivindicava que tal máquina seria capaz de calcular qualquer número calculável, ou computável, como se dizia então. Funcionaria, portanto, como a mente desse profissional, o computador. Turing, de fato, afirma: “podemos agora construir uma máquina para fazer o trabalho desse computador”[1] (op. cit., p. 251), referindo-se à profissão. Utilizou sua máquina abstrata para realizar algumas importantes demonstrações matemáticas, e determinou um limite teórico para o que poderia ou não ser calculado utilizando tal máquina.

Hoje a palavra “computador” raramente é utilizada no sentido original, apesar de ser o sentido usado por Turing. Pode-se dizer que o trabalho de Turing iniciou um processo que mudou seu significado. Um desses significados designa um tipo de equipamento que implementa uma máquina com uma capacidade, em princípio, semelhante à preconizada por Turing, sendo provavelmente (mas não necessariamente) construída de acordo com uma arquitetura conhecida com arquitetura de von Neumann.[2] Apresenta-se sob inúmeras configurações, para incontáveis finalidades. Descobriram-se limites práticos para a computação, além dos limites teóricos descobertos por Turing; veremos alguns desses limites mais adiante neste capítulo.

Tal abrangência nos obriga a delimitar claramente qual o sentido que estamos dando às palavras “computador”, ou “computador eletrônico”, neste trabalho. Escolhemos, por motivos que ficarão claros mais adiante, uma máquina real. Qualquer um dos modernos equipamentos utilizados profissionalmente por milhões de pessoas no mundo todo para suas tarefas diárias. Para delimitarmos um escopo sem sombra de dúvida, adotaremos como arquétipo o equipamento que ora utilizo, uma máquina construída pela Dell, um Inspiron 5447 i7, tendo como processador central um Intel® Core™ i7-4510U com 4 núcleos de 2.00 GHz, placa gráfica Intel® Haswell Mobile. Ampliei sua memória RAM para 16GB, e seu disco original foi substituído por um com 1TB de capacidade. O sistema operacional é o Ubuntu 16.04 LTS “Xenial Xerus”.

---

1 We may now construct a machine to do the work of this computer.

2 Graças ao trabalho (VON NEUMANN, 1993) publicado em 1945 pelo matemático húngaro-americano John von Neumann (1903-1957).

Esta definição de computador, embora precisa, não explicita capacidades do referido equipamento. Vamos delimitar mais claramente suas limitações a seguir. Por esta razão, neste capítulo exporemos a conceituação matemática de função computável e suas limitações, contextualizando-os numa moldura filosófica. Assim, iniciaremos falando sobre os fundamentos da matemática na Seção 4.1, que nos permitirá situar filosoficamente o alcance das seções seguintes. Em seguida falaremos sobre o teorema de Turing na Seção 4.2, e explicitaremos alguns limites e capacidades de computadores tanto teóricos quanto reais na Seção 4.3.

4.1 Fundamentos da matemática

A presente seção se baseia no livro do professor Newton da Costa, que inicia a sua "Introdução aos Fundamentos da Matemática" afirmando que

> [A] filosofia da matemática deve determinar, entre outras coisas, quais as suposições e as ideias que servem de fundamentos para as verdades matemáticas. Como decorrência desta tarefa, a análise do mecanismo dedutivo, ou seja, da própria lógica formal, pertence ao domínio da filosofia da matemática. (COSTA, 1992, p. 13).

Trata-se de uma visão que difere da de Peirce, para quem a análise do mecanismo dedutivo faz parte da semiótica, uma ciência normativa que explica os raciocínios em geral, inclusive os raciocínios matemáticos. Para Peirce a matemática se ocupa de duas atividades distintas e complementares: (1) determina hipóteses que são independentes de qualquer coisa do mundo real e (2) traça as consequências necessárias dessas hipóteses (CP 3.559, 1898). Mas a visão de Costa coincide com a visão da maioria dos matemáticos, talvez de todos eles. Passamos a expô-la. De acordo com ele, as inúmeras tendências no âmbito da filosofia da matemática variam em torno de três correntes principais: logicismo, intuicionismo e formalismo, que explanaremos a seguir. Costa propõe em seu livro uma quarta interpretação da matemática, que chamou de linguística e que se baseia em teorias da linguagem que, por serem pouco aderentes à semiótica peirciana como a conhecemos hoje, evitaremos comentar.

Logicismo

A tese central do logicismo é, resumidamente, que a matemática reduz-se à lógica; não à lógica no sentido tradicional, mas ao que Newton da Costa (op. cit., p. 19) chamou de *logística*, que também é conhecida como lógica matemática, lógica simbólica ou lógica algorítmica. A logística se desenvolveu enormemente no século XIX através principalmente dos trabalhos de George Boole (1815-1864) e de Giuseppe Peano (1858-1932), e a escola logicista tem seu auge na publicação dos três volumes, por Bertrand Russell (1872-1970) e Alfred North Whitehead (1861-1947), do *Principia Mathematica* (WHITEHEAD; RUSSELL, 1912–1913). A

teoria compõe-se de duas partes: (1) toda ideia matemática pode ser definida por intermédio de conceitos lógicos e (2) todo enunciado matemático verdadeiro pode ser demonstrado a partir de princípios lógicos, mediante raciocínios puramente lógicos.

A visão logicista da matemática, ainda segundo Costa, esbarrou em diversos paradoxos e contradições de natureza lógica, que obrigaram seus apoiadores a adotar conceitos e axiomas que não são considerados puramente lógicos. Além disso, na época de sua elaboração, não se conhecia, como hoje, uma variedade de sistemas lógicos; a continuidade entre a lógica de então e a matemática tinha o apelo de um purismo de raciocínio que não se sustentou com a descoberta de outros sistemas lógicos não triviais. Por essas razões, adicionadas ao fato de alguns conceitos adotados para resolver contradições dizerem respeito ao mundo real (o que contraria a ideia de que a matemática não depende do que se observa no mundo real), vê-se em xeque o purismo da visão logicista da matemática.

Intuicionismo

Essa escola tem como precursor o matemático alemão Leopold Kronecker (1823-1891), que não admitia elementos matemáticos que não pudessem ser efetivamente construídos com base nas noções intuitivas[3] que suportam os números naturais. "É comum, em matemática, inferirmos, por exemplo, que um número x existe, porque sua não existência implicaria contradição", afirma Costa (op. cit., p. 34). A afirmação de que x existe não tinha valor para Kronecker se não mostrasse como x foi construído. Para ele também não se podia admitir que conjuntos infinitos fossem concebidos como realizados; coleções infinitas o seriam somente potencialmente, e somente coleções finitas poderiam ser inteiramente dadas. Isso invalidaria teorias de sua época a respeito dos números reais.

As ideias de Kronecker foram levadas ao extremo pelo matemático holandês Luitzen E. J. Brouwer (1881-1966), considerado o fundador da escola intuicionista. Para ele a matemática não se compõe de "verdades eternas, relativas a objetos intemporais, metafísicos, semelhantes às ideias platônicas" (COSTA, op. cit., p. 36), mas de entidades criadas pelo próprio matemático: a afirmação matemática de que algo existe, para ele, significa somente que esse algo foi construído pela inteligência humana. A matemática, então, pertenceria à categoria das atividades sócio-biológicas, destinada a satisfazer certas exigências vitais da humanidade, e o matemático cria e dá forma aos entes matemáticos. A origem histórica da matemática estaria na experiência, através dos sentidos, mas sua estruturação final seria rigorosa, puramente intuitiva (uma intuição de caráter racional, nada tendo de mística) e baseada na noção de número natural, sendo independente de outras ciências ou da filosofia.[4]

---

[3] Segundo Peirce não temos poder de intuição (cf. CP 5.265, 1868); tal escola de pensamento matemático é destituída de sentido numa moldura peirciana.

[4] Uma posição, nesse preciso ponto, semelhante à que Peirce dá à matemática.

A matemática intuicionista apresenta particularidades que a tornam de difícil trato. Por negar, por exemplo, o valor universal do princípio do terceiro excluído,[5] não se vale das demonstrações por redução ao absurdo. Isso exigiu dos intuicionistas a criação de uma nova lógica aderente a esses princípios. Por exemplo, uma afirmação do tipo "todo número natural x possui a propriedade P" tem um caráter apenas hipotético, já que não se pode averiguá-la para todos os infinitos números naturais. Convém observar que tal lógica não serve como fundamento para a matemática intuicionista. Foi apenas "abstraída" dessa escola (COSTA, op. cit., p. 41).

A escola intuicionista trouxe resultados importantes ao incitar os estudiosos de outras escolas a procurar novos métodos, na esperança de reabilitar as teorias clássicas. Há algumas críticas de caráter filosófico ao intuicionismo, sendo a mais decisiva, do ponto de vista de Costa, o fato de que "se o intuicionismo prevalecesse, a velha ciência matemática seria completamente desfigurada. [...] Uma filosofia correta e apropriada da matemática precisa estar de acordo com o desenvolvimento autêntico da matemática, o que não acontece com o intuicionismo" (COSTA, op. cit., p. 46). Mesmo sendo vista como uma escola que restringe o trabalho do matemático clássico por privá-lo de recursos lógicos usuais, inspirou correntes ainda mais restritivas; por exemplo, a proposta pelo matemático holandês George Griss (1898-1953) pretende excluir da matemática qualquer forma de negação.

Formalismo

Numa frase, a fundamentação formalística da matemática, proposta pelo matemático alemão David Hilbert (1862-1943), consiste em converter o método axiomático em fundamento da matemática. Segundo Costa, para estudar uma teoria pelo método axiomático procede-se assim:

> escolhe-se certo número [de] noções e de proposições primitivas, suficientes para sobre elas edificar a teoria, aceitando-se outras ideias ou outras proposições só mediante, respectivamente, definições e demonstrações; obtém-se, dessa maneira, uma *axiomática material* da teoria dada; deixam-se de lado os significados intuitivos dos conceitos primitivos, considerando-os como termos caracterizados implicitamente pelas proposições primitivas. Procuram-se, então, as consequências do sistema obtido, sem preocupação com a natureza ou com o significado inicial desses termos ou das relações entre eles existentes. Estrutura-se, assim, o que se denomina uma *axiomática abstrata*. (COSTA, op. cit., p. 49)

O poder do método axiomático é conhecido desde a antiguidade, e sua utilização trouxe grande desenvolvimento a importantes campos da matemática, especialmente a partir do século XIX. Hoje é a técnica básica da matemática.

[5] Princípio segundo o qual uma proposição ou é verdadeira ou é falsa, não podendo assumir um terceiro valor.

A concepção formalista da matemática a vê como a ciência da estrutura dos objetos. O matemático só pode estudar as propriedades dos objetos por meio de um sistema adequado de símbolos, que seja privado daquilo que é destituído de importância. Se o sistema é adequado, o matemático não precisa mais se incomodar com o significado dos símbolos, pois pode constatar nos próprios símbolos as propriedades estruturais que o interessam. Daí a relevância das características formais da linguagem matemática — donde o nome "formalismo". A introdução de conceitos e princípios sem conteúdo intuitivo justifica-se, desde que não leve a contradições; a finalidade disso seria simplificar e sistematizar as disciplinas matemáticas, mantendo a validade das leis da lógica clássica.

Para Hilbert, o matemático pode estudar qualquer sistema simbólico que não contenha contradições, e para garantir isso edificou a metamatemática, ou teoria das demonstrações, com a finalidade de provar a consistência das diversas disciplinas matemáticas. Os passos para tal seriam: *axiomatização*, ou a determinação dos conceitos primitivos, *formalização*, ou a substituição de relações e princípios lógicos encontrados nos conceitos primitivos por símbolos e arranjos simbólicos sujeitos a regras bem definidas, que prescindem da significação dos conceitos primitivos, e *demonstração da consistência da axiomática formalizada*, buscando provar a impossibilidade da axiomática levar a contradições. Dessa forma, a axiomatização fundamentaria a matemática. Esse trabalhou avançou: demonstrou-se que certas teorias seriam consistentes se outras teorias também o fossem. Por exemplo, se a geometria euclidiana for consistente, então a geometria plana desenvolvida pelo matemático russo Nikolai Lobatchewsky (1792-1856) também o é. E se a análise matemática for consistente, a geometria comum também o é. Certas versões mais restritas da aritmética "comum" tiveram sua consistência demonstrada.

O programa de Hilbert revelou-se impossível graças ao trabalho do matemático austríaco Kurt Gödel (1906-1978), que em 1931 publicou seus famosos teoremas da incompletude (GÖDEL, 1992). O seu primeiro teorema prova que *toda axiomática da aritmética é incompleta*. Uma axiomática é completa se, dado qualquer enunciado S formulável nessa axiomática, pode-se provar que S é verdadeiro, ou que sua negação é verdadeira. Portanto, o primeiro teorema prova que há enunciados, chamados *indecidíveis*, tais que nem eles nem suas negações podem ser demonstrados na axiomática que se adotar, *qualquer que seja ela*. Note-se que os enunciados indecidíveis, sendo proposições, devem portanto ser ou verdadeiros ou falsos. Não sendo demonstráveis, podem ser incorporados como axiomas ao conjunto inicial de axiomas (o mesmo valendo para suas negações, claro que não no mesmo sistema axiomático). Pois bem: essa nova axiomática, à qual se acrescentou a proposição indecidível como axioma, também é incompleta (cf. NAGEL; NEWMAN, 1986). O segundo teorema de Gödel, que é consequência do primeiro, afirma que *a consistência de qualquer axiomática consistente da aritmética não pode ser demonstrada nessa axiomática*. Ou seja, não é possível provar a consistência da aritmética utilizando a própria aritmética. Esses teoremas valem para qualquer sistema axiomático que englobe a aritmética, valendo, portanto para toda a matemática que conhecemos.

A consequência dos teoremas de Gödel é que o fato de a matemática ser produto do método axiomático não garante sua consistência, como queria Hilbert. Portanto, o método axiomático não basta para fundamentá-la. Isso não tira o valor do método axiomático como técnica, sendo o método mais amplamente utilizado na pesquisa matemática hoje.

Considerações adicionais

Cabem aqui algumas considerações a respeito da prova de Gödel. Uma das dificuldades mais relevantes que superou foi a de expressar dentro da aritmética proposições a respeito da própria aritmética.[6] Para isso criou o que ficou conhecido como *numeração de Gödel*, e que é uma maneira de calcular, a partir de um enunciado aritmético — escrito no formalismo do "Principia Mathematica" (WHITEHEAD; RUSSELL, 1912–1913), obra capital da escola logicista —, um número inteiro único. Esse é o *número de Gödel* do enunciado.

Dado qualquer enunciado aritmético, é possível calcular seu número de Gödel. Dado qualquer número de Gödel, é possível determinar o enunciado que lhe deu origem. Num sistema formal a evolução das demonstrações não depende do significado dos enunciados que as compõem. É possível estabelecer quais mudanças formais correspondem a passos válidos nas demonstrações. Portanto, passagem de um enunciado válido E_1, cujo número de Gödel é $\mathfrak{G}_{E_1}$, para outro enunciado válido E_2 (número de Gödel $\mathfrak{G}_{E_2}$) implica numa relação entre os números $\mathfrak{G}_{E_2}$ e $\mathfrak{G}_{E_1}$. Essa relação é, ela mesma, um enunciado aritmético e tem, portanto, o seu próprio número de Gödel. O cerne do argumento de Gödel está resumido nos seguintes cinco passos (cf. NAGEL; NEWMAN, 1986, p. 85–86):

1. Gödel mostrou como construir uma fórmula aritmética G que representa o enunciado "a fórmula G não é demonstrável".
2. Em seguida provou que a fórmula G é demonstrável se, e somente se, sua negação formal ¬G também for demonstrável. Ora, se a aritmética for consistente, das duas uma: ou G é demonstrável, ou sua negação ¬G é demonstrável, mas não ambas. Então, se a aritmética for consistente, G não é demonstrável (nem sua negação). Portanto G é um enunciado indecidível.
3. Prosseguiu, provando: embora não seja demonstrável, G é um enunciado verdadeiro — intuitivamente vemos isso, já que G afirma que ela mesma não é demonstrável.
4. Sendo G verdadeira mas indecidível, conclui-se que os axiomas da aritmética são incompletos, mesmo que os ampliemos de modo a permitir a derivação de G.
5. Finalmente Gödel mostrou como criar a fórmula A que representa o enunciado "a aritmética é consistente"; e, relacionando A e G, provou que A não pode ser demonstrada.

[6] Essa é uma pré-condição para as provas absolutas de consistências buscadas por Hilbert. Pois, caso a consistência de um sistema **X** fosse provada em outro sistema **Y**, então a consistência de **X** dependeria de haver prova da consistência de **Y** (cf. NAGEL; NEWMAN, 1986, p. 26 e seguintes).

Temos, assim, que a aritmética é essencialmente incompleta e sua consistência não pode ser demonstrada dentro do seu próprio formalismo. Ressalte-se que hoje se entende que os resultados de Gödel se aplicam também à matemática intuicionista.

Os resultados de Gödel foram utilizados por Turing no famoso artigo em que trata do problema da decisão (TURING, 1936), no qual cria a noção de máquina de Turing e demonstra que existe uma máquina de Turing universal, tema a que nos voltamos a seguir.

4.2 Funções computáveis

As máquinas de Turing foram foram criadas por Alan Turing (1936) para uma demonstração matemática. Uma função computável é uma função que pode ser executada por uma máquina de Turing de computar, descrita a seguir. Essas máquinas podem ser vistas como uma formalização do conceito intuitivo de computador — palavra que na época designava uma profissão cujo principal trabalho era realizar e conferir cálculos matemáticos. Com as máquinas de Turing é possível definir algoritmos com rigor matemático. Não se sabe se todos os números calculáveis podem ser calculados por máquinas de Turing. Isso depende, claro, da definição do que seja um número calculável. A hipótese de que todo número calculável pode ser calculado por uma máquina de Turing é conhecida como tese de Church-Turing. Até o momento, todas as formalizações matemáticas do conceito intuitivo de computabilidade geraram um sistema que tem o mesmo poder computacional das máquinas de Turing (cf. CARNIELLI; EPSTEIN, 2006).

Os resultados a que Turing chegou em seu artigo são puramente matemáticos; um dos principais resultados é a prova de que o problema da decisão (conhecido pelo nome em alemão, *Entscheidungsproblem*), proposto por Hilbert em 1928, também não tem solução. O problema consiste em encontrar um algoritmo que, dada uma declaração lógica, determina se essa declaração pode ser derivada a partir de um conjunto de axiomas. Num passo intermediário para provar que não existe tal algoritmo, Turing mostrou que não existe uma máquina de Turing capaz de prever o resultado de qualquer máquina de Turing.[7] Em outras palavras, não há um procedimento definido com rigor matemático que permita determinar o resultado de uma função computável. Em última análise, essa limitação é a razão pela qual não há um método geral para evitar erros em programas de computador.

Em um trecho da prova Turing faz uso do teorema de Gödel de um modo que explicita sua importância:

[7] Para isso usou um procedimento de certo modo semelhante à numeração de Gödel, mostrando que existe uma máquina de Turing universal.

> Se a negação do que Gödel mostrou tivesse sido provada, ou seja, se, para cada $\mathfrak{A}$,[8] ou $\mathfrak{A}$ ou $\neg\mathfrak{A}$ pudesse ser provado, então teríamos uma solução imediata para o Entscheidungsproblem. Pois podemos inventar uma máquina $\mathcal{K}$ que provará consecutivamente todas as fórmulas que podem ser provadas. Cedo ou tarde $\mathcal{K}$ chegará ou a $\mathfrak{A}$ ou a $\neg\mathfrak{A}$. Se chegar a $\mathfrak{A}$, então saberemos que $\mathfrak{A}$ pode ser provada. Se chegar a $\neg\mathfrak{A}$ então, uma vez que **K** é consistente (Hilbert e Ackermann, p. 65),[9] saberemos que $\mathfrak{A}$ não pode ser provada.[10] (TURING, 1936, p. 259)

Esse trecho ilustra o alcance dos teoremas de Gödel e da máquina de Turing. Sigamos uma linha de raciocínio sugerida por Nagel e Newman (1986): vimos que o método axiomático conta com um pequeno número de regras — axiomas e formas de cálculo — que podem ser usadas para construir verdades a respeito do sistema que representam. O exemplo da geometria euclidiana é eloquente: um pequeno número de regras permite deduzir as inúmeras verdades geométricas que conhecemos. Temos a impressão de que todas as verdades geométricas podem ser construídas a partir desse pequeno conjunto de elementos que admitimos verdadeiros. A construção dessas verdades pode, inclusive, ser um tanto quanto mecânica: o próprio Turing imaginava ser possível construir todas as regras deriváveis de um conjunto de elementos axiomáticos primitivos utilizando a máquina que imaginou. O que o teorema de Gödel garante é que sempre haverá alguma verdade fora do alcance dessa construção mecânica a partir do conjunto inicial de elementos primitivos; e que não teremos como provar que essa construção a partir desse conjunto inicial não leva a alguma contradição. Essa importante limitação do método axiomático, bem como as da máquina de Turing, serão analisadas à luz da filosofia peirciana no capítulo 5. Mas antes vamos à descrição máquina de Turing.

Máquinas de Turing

Praticamente qualquer texto introdutório a respeito de computabilidade descreve uma versão da máquina de Turing. Escolhemos das palavras de seu criador:

> Podemos comparar um homem no processo de calcular um número real a uma máquina que é capaz apenas de um número finito de condições $q_1, q_2, \ldots, q_R$ que serão chamadas "configurações-m". A máquina é fornecida com uma "fita" (o análogo ao papel) que corre através dela, e [é] dividida em seções (chamadas "quadrados") cada uma capaz de carregar um "símbolo". A qualquer momento há somente um quadrado, digamos o r-ésimo, carregando o símbolo $\Im(r)$ o qual está "na máquina". Podemos chamar esse quadrado de "quadrado examinado". O símbolo no quadrado examinado pode ser chamado de "símbolo

[8] $\mathfrak{A}$ representa uma afirmação lógica qualquer dentro do sistema axiomático **K** ao qual se aplica o problema da decisão. A negação de $\mathfrak{A}$ é escrita como $\neg\mathfrak{A}$.

[9] Turing refere-se ao "Fundamentos da Lógica Teórica" ("Grundzüge der Theoretischen Logik"), publicado por Hilbert e Ackermann em Berlim no ano de 1931.

[10] If the negation of what Gödel has shown had been proved, i.e. if, for each $\mathfrak{A}$, either $\mathfrak{A}$ or $\neg\mathfrak{A}$ is provable, then we should have an immediate solution of the Entscheidungsproblem. For we can invent a machine $\mathcal{K}$ which will prove consecutively all provable formulae. Sooner or later $\mathcal{K}$ will reach either $\mathfrak{A}$ or $\neg\mathfrak{A}$. If it reaches $\mathfrak{A}$, then we know that $\mathfrak{A}$ is provable. If it reaches $\neg\mathfrak{A}$, then, since **K** is consistent (Hilbert and Ackermann, p. 65), we know that $\mathfrak{A}$ is not provable.

> examinado". O "símbolo examinado" é o único dos quais a máquina está, por assim dizer, "diretamente consciente". De qualquer modo, através da alteração de sua configuração-*m* a máquina pode efetivamente lembrar alguns dos símbolos que ela "viu" (examinou) previamente. O comportamento possível da máquina a qualquer momento é determinado pela configuração-*m* q_n e o símbolo examinado $\mathfrak{S}(r)$. Este par q_n, $\mathfrak{S}(r)$ chamar-se-á "configuração": assim, a configuração determina o possível comportamento da máquina. Em algumas das configurações nas quais o quadrado examinado é branco (i.e. não carrega nenhum símbolo) a máquina escreve um novo símbolo no quadrado examinado: em outras configurações ela apaga o símbolo examinado. A máquina pode também trocar qual quadrado está sendo examinado, mas apenas deslocando-o um lugar para a direita ou para a esquerda. Adicionalmente a qualquer uma dessas operações a configuração-*m* pode ser trocada. Alguns símbolos escritos formarão a sequencia de algarismos [*figures*, em inglês] que são a [representação] decimal do número [*number*, em inglês] que está sendo computado. Os outros são apenas notas de rascunho para "auxiliar a memória". Serão somente essas notas de rascunho que serão suscetíveis de serem apagadas.
>
> Afirmo que essas operações incluem todas as que são usadas no cálculo de um número.[11] (TURING, 1936, p. 231–232)

A cada momento, a máquina se encontra em alguma configuração-*m*, e qual ação irá performar depende da configuração-*m* particular em que se encontra e qual o símbolo examinado. Turing inicialmente representou o conjunto de configurações-*m* de uma máquina que imprime a sequência "010101…" como na Tabela 2. Nela, a letra "D" quer dizer movimento à direita ("E" seria

Tabela 2 – Configurações-*m* duma máquina de Turing que imprime "010101…"

Configuração		Comportamento	
Configuração-*m*	Símbolo examinado	Operações	Configuração-*m* final
$\mathfrak{b}$	Nenhum	I0, D	$\mathfrak{c}$
$\mathfrak{c}$	Nenhum	D	$\mathfrak{e}$
$\mathfrak{e}$	Nenhum	I1, D	$\mathfrak{f}$
$\mathfrak{f}$	Nenhum	D	$\mathfrak{b}$

Fonte: Turing (1936)

[11] We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions $q_1, q_2, \ldots, q_R$ which will be called "*m*-configurations". The machine is supplied with a "tape" (the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the *r*-th, bearing the symbol $\mathfrak{S}(r)$ which is "in the machine". We may call this square the "scanned square". The symbol on the scanned square may be called the "scanned symbol". The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". However, by altering its *m*-configuration the machine can effectively remember some of the symbols which it has "seen" (scanned) previously. The possible behaviour of the machine at any moment is determined by the *m*-configuration q_n and the scanned symbol $\mathfrak{S}(r)$. This pair q_n, $\mathfrak{S}(r)$ will be called the "configuration": thus the configuration determines the possible behaviour of the machine. In some of the configurations in which the scanned square is blank (i.e. bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or left. In addition to any of these operations the *m*-configuration may be changed. Some of the symbols written down will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to "assist the memory". It will only be these rough notes which will be liable to erasure.
It is my contention that these operations include all those which are used in the computation of a number.

à esquerda), "I" quer dizer imprimir ("I0", portanto, "imprimir 0") e "A" significar "apagar" o símbolo examinado. Turing destaca que a máquina inicia na configuração-*m* 𝔟 com uma fita em branco. Aqui pode-se observar que é possível que a lista de configurações de uma máquina pode levar a situações nas quais a máquina para de imprimir, mas não encerra seus movimentos. Por exemplo, se a configuração-*m* final da configuração-*m* 𝔠 fosse o próprio 𝔠 ao invés de 𝔢, teríamos uma máquina que imprime o número 0 uma vez e continua avançando indefinidamente para a direita sem imprimir mais nada.

Em seu artigo, Turing imaginou alguns tipos de máquina e as classificou conforme seu comportamento geral. Antes introduziu o conceito de "configuração completa". Em qualquer estágio do movimento da máquina, a configuração completa é descrita pelo número do quadrado examinado, pela sequência completa de todos os símbolos na fita, e pela configuração-*m* em que a máquina se encontra. As mudanças na máquina e na fita entre duas configurações completas sucessivas são chamados "movimentos" da máquina.

- Uma "máquina automática", ou "máquina-*a*", é uma máquina em que, a cada estágio, o movimento é completamente determinado pela configuração.
- A "máquina-*c*" (*choice machine*, ou máquina de escolha) tem o movimento apenas parcialmente determinado pela configuração. Quando chega a uma configuração ambígua, ela para de se mover até que alguma escolha arbitrária seja feita por um operador externo, que não precisa ser necessariamente um ser humano, podendo ser, por exemplo, outra máquina de Turing.
- A "máquina de computar" é uma máquina automática que imprime dois tipos de símbolos, dos quais o primeiro tipo (chamados figuras ou caracteres) consistem inteiramente de 0 e 1, os outros sendo chamados de "símbolos do segundo tipo"; suprida com uma fita em branco e colocada em movimento, começando da configuração-*m* correta, a sequência de símbolos impressas por ela que são do primeiro tipo chamar-se-á "sequência computada pela máquina". O número real cuja expressão binária é dada pela introdução de um ponto decimal no início da sequência é chamado "número computado pela máquina".
- Máquinas de computar "circulares", que chegam a configurações a partir das quais não há movimento possível, ou se o movimento continua ela possivelmente continua imprimindo símbolos do segundo tipo mas não do primeiro tipo.
- Máquinas de computar "livres de circularidade": máquinas automáticas de computar que não se enquadram como circulares. "Sequência computável" é aquela que pode ser computada por uma máquina de computar livre de circularidade; e "número computável" é aquele que difere por um inteiro do número computável por uma máquina de computar livre de circularidade.
- Finalmente, a "máquina de Turing de computar universal", ou "máquina de Turing universal", é uma máquina de computar cujas configurações-*m* são tais que permitem à máquina

universal comportar-se como qualquer outra máquina de Turing cujas configurações-*m* estejam apropriadamente descritas na própria fita da máquina universal.

A máquina de Turing universal inaugura um novo conceito. Até então o funcionamento das máquinas em geral era inteiramente determinado somente pelo seu mecanismo e pelas leis naturais que regem o funcionamento do mecanismo. Para alterar o funcionamento da máquina, normalmente era preciso alterar fisicamente o seu mecanismo; alterações de configuração, como por exemplo, trocar a marcha numa caixa de câmbio (que em última análise altera o funcionamento da máquina sem necessidade de troca de peças), poderiam ser previstas, mas tinham um limite físico. Nas máquinas de Turing, esse mecanismo "físico" é representado pelo conjunto das configurações-*m* da máquina. Na máquina universal as configurações-*m* são tais que ela funciona de acordo com o que está escrito na fita. Para alterar o funcionamento desse tipo de máquina não é preciso alterar seu mecanismo "físico": basta alterar o conteúdo da fita. E, uma vez que qualquer máquina de Turing pode também escrever na fita, a máquina de Turing universal pode, teoricamente, mudar seu próprio comportamento, mudando a parte da fita que descreve seu comportamento. Essa mudança de comportamento não tem limite teórico: qualquer outra máquina de Turing pode ser descrita na fita. De fato, isso acontece nos modernos computadores: são mapas, tocadores de música, aparelhos telefônicos, sem que se precise alterar nenhuma configuração física. Uma capacidade que nenhuma máquina até então apresentou, mais assombrosa ainda se pensarmos que a fita alterada pela máquina pode gerar uma outra máquina que também pode modificar-se — desde que tenha sido programada para isso —, num processo recursivo. Quais são os limites dessa máquina? O que ela pode, de fato, fazer? Talvez essas considerações tenham levado Turing a imaginar que os computadores poderiam ter comportamento que pudesse ser considerado inteligente, como expôs em seu artigo de 1950 (TURING, 1950). Vejamos então alguns dos limites conhecidos dos computadores.

4.3 Limites dos computadores

A presente seção expõe alguns limites conhecidos dos computadores teóricos (Seção 4.3.1) e reais (Seção 4.3.2). O objetivo é embasar uma análise semiótica dos computadores e das capacidades e limitações encontradas para, a partir daí, determinar as capacidades dos computadores reais como agentes semióticos e, assim, apontar caminhos para novas formas de programação nas quais as máquinas tenham maior autonomia.

4.3.1 Limites dos computadores teóricos

Já conhecemos um limite muito bem definido para as máquinas de Turing: não pode ser construída uma máquina de Turing capaz de prever o funcionamento de uma máquina de Turing qualquer. Essa é uma limitação matemática, ou seja, não existe nenhum conjunto finito de

configurações-*m* que descrevam uma máquina de Turing $\mathcal{T}$ capaz de ter como entrada a descrição de uma máquina de Turing $\mathcal{X}$ e como saída alguma informação sobre o resultado da máquina $\mathcal{X}$, qualquer que seja a informação: se ela é circular ou não, ou mesmo se ela imprime o algarismo 1 ou não (cf. TURING, 1936, p. 248). Nessa demonstração utilizou-se do fato de a máquina de Turing universal ser capaz de simular qualquer máquina de Turing a partir do que se encontra em sua fita, num procedimento que lembra o da numeração de Gödel.

No artigo de 1936 Turing apenas apresenta o conceito de máquina de escolha (máquina-*c*). O conceito não foi desenvolvido: o artigo trata somente de máquinas automáticas, em particular as de computar universais. Em sua tese de doutorado (TURING, 1939) introduziu um outro tipo de máquina, a "máquina-*o*", (*oracle machine*, ou máquina-oráculo), que possui uma configuração adicional na qual a máquina faz uma "pergunta ao oráculo". Esse oráculo é capaz de dar qualquer resposta não computável[12] à máquina. Difere da máquina de escolha: esta espera o auxílio de um operador externo que determina o seu próximo passo, mas essa escolha não tem nenhuma restrição, podendo ser resultado de um procedimento computável. Já a máquina-oráculo só continua graças a uma decisão não computável que é, não obstante, correta. Uma máquina-oráculo é, portanto, capaz de dar informações corretas a respeito do resultado de uma máquina de Turing automática qualquer.

A máquina-oráculo é utilizada no estudo de problemas de decidibilidade e computabilidade. Por exemplo, demonstra-se que uma máquina-oráculo que é capaz de determinar se uma máquina de Turing automática é ou não livre de circularidade, essa máquina-oráculo não é capaz de determinar se uma máquina-oráculo igual a si própria é ou não livre de circularidade (cf. TURING, 1939, p. 172–173). Daí, haverá uma outra máquina-oráculo que é oráculo para a primeira máquina-oráculo, e assim sucessivamente, gerando uma espécie de "hierarquia" de máquinas-oráculo. O teorema de Post (KLEENE, 1952) determina uma relação entre o grau de insolubilidade (ou grau de Turing, que é, aproximadamente, o "nível" da máquina-oráculo na "hierarquia" que intuímos) e a complexidade da definição de conjuntos de números naturais. De uma maneira simplificada, vale dizer que o problema de determinar se um número pertence a determinado conjunto é tanto mais difícil, no sentido de exigir uma máquina-oráculo de hierarquia mais alta, quanto mais complexa for sua definição.

Um limite de computabilidade que nos interessa e é relacionado à capacidade de computação de máquinas-oráculo é o apresentado por Davis (2006, p. 128). Citando os trabalhos de Gold (1965) e Putnam (1965), dois artigos contíguos publicados no mesmo jornal, Davis propõe uma computação que chamou de "tentativa e erro". Trata-se de um procedimento que produz resultados de tempos em tempos, sem limitação de prazo temporal. Não há também garantia de que o último resultado apresentado seja o correto, sabe-se somente que, se for correto, então ele será repetido nas saídas subsequentes. Uma situação peculiar, porque não há como saber, a

[12] Ou seja, que não pode ser calculada por uma máquina de Turing de computar.

cada momento, se o último resultado mostrado é, de fato, correto ou não, mesmo que ele seja a repetição dos *n* últimos resultados. Nos artigos originais os autores mostram sua motivação. Gold (p. 30) destaca as limitações inerentes à teoria dos autômatos para resolver um problema que vem da inteligência artificial: as funções que estudou seriam um modelo preliminar do problema que o pensador tem para identificar o universo. Putnam (1965) procurou estudar um tipo de procedimento que pode "mudar de opinião" e apresentar resultado diferente do que apresentou antes, num processo que vai se repetir até que o procedimento não "mude" mais de opinião, atingindo um estado equivalente à certeza; mas não há como saber se isso de fato aconteceu ou não. O resultado a que chegaram, que é explorado por Davis, é que tais procedimentos são equivalentes aos procedimentos computáveis por uma máquina-oráculo como a apresentada por Turing em sua tese.

Os resultados dos parágrafos anteriores são relevantes, porque uma alternativa para a limitação decorrente dos teoremas de Gödel, a saber, a impossibilidade de determinar de forma computável todas as declarações verdadeiras dentro de um sistema axiomático razoavelmente expressivo[13] é, justamente, a tentativa e erro por sorteio: simplificadamente, a escolha de uma solução sem nenhum critério pré-definido e posterior verificação de suas consequências. Esse procedimento não é computável por uma máquina de Turing.[14] Proporemos no presente trabalho maneiras de utilizar procedimentos não computáveis desse tipo para encontrar algumas soluções; se correspondem, ou não, a máquinas-oráculo, deve ser alvo de futuras investigações. Os procedimentos não computáveis que proporemos têm base nas capacidades e limitações dos computadores reais.

4.3.2 Limites dos computadores reais

Os computadores reais — e aqui entendemos que o computador arquetípico apresentado na página 77 é o caso de computador real a que nos referimos — se aproximam mais das máquinas de Turing de escolha do que das máquinas automáticas. São equipamentos que, quando ligados, performam uma série de ações até que, em dado momento, parecem parar e esperar uma ação do usuário. Assim, a modelagem matemática do artigo de Turing (1936) não vale para o conjunto computador–usuário humano, já que não formam uma máquina de Turing; nem o computador real funciona exatamente como uma máquina de Turing automática, mas como uma máquina de Turing de escolha, abstração definida por Turing no artigo, mas não estudada por ele. Não obstante, a diferença efetiva entre a máquina de escolha e a máquina automática é que nas primeiras há estados em que a máquina para e espera, e nas segundas, não. Nos estados intermediários entre os estados de espera da máquina de escolha ela se comporta

[13] Por "razoavelmente expressivo" queremos dizer "capaz de expressar pelo menos a aritmética básica".

[14] Falando rigorosamente, há dois limites aqui. Um, escolher uma solução sem nenhum critério: "escolher sem nenhum critério" não é um procedimento computável. Outro, escolhida a solução, saber se é demonstrável no sistema axiomático escolhido. É justamente esse último o limite demonstrado por Turing em seu artigo.

exatamente como uma máquina automática, o que permite modelar, utilizando as abstrações de Turing, seu comportamento entre dois estados de espera.

Os computadores reais apresentam várias características que os diferenciam das máquinas de Turing automáticas, pelo simples fato de que estas últimas são imaginárias. A mais óbvia é que computadores reais não dispõem de uma fita infinita. Uma outra diferença é de ordem prática: possuem uma velocidade finita de processamento, o que implica dizer que as movimentações entre diferentes configurações-*m* tomam tempo real. Isso é particularmente importante quando se tenta resolver problemas do tipo não polinomial (conhecidos como problemas NP), nos quais a quantidade de movimentos aumenta exponencialmente com o tamanho do problema a resolver. Não fora essa segunda limitação, muitos perguntas teriam resposta quase imediata, como o problema de vencer sempre no xadrez ou o determinar a melhor rede neural para um determinado conjunto de dados.

Finalmente, os computadores reais têm uma capacidade que as máquinas de Turing teóricas não têm: eles são capazes de "acumular entropia" de seus ambientes e utilizá-la para produzir números aleatórios. Isso possibilita implementar nelas procedimentos que fazem escolhas sem nenhum critério, uma impossibilidade para máquinas de Turing automáticas. A importância dessa capacidade ficará mais clara quando quisermos aumentar a autonomia dos computadores como agentes semióticos.

5 Computadores agentes semióticos

No presente capítulo vamos tentar enxergar os computadores, da maneira que foram definidos no Capítulo 4, como agentes semióticos, descritos no Capítulo 3. Nosso objetivo é conceituar os principais pontos antes de esboçar, no próximo capítulo, uma proposta de implementação que contempla os elementos expostos até aqui.

O capítulo inicia com uma visão peirciana do formalismo matemático e das máquinas de Turing na Seção 5.1. Prossegue analisando como a vagueza e a indeterminação aparecem na matemática, nas máquinas de Turing e nos computadores eletrônicos, na Seção 5.2. Na Seção 5.3 são expostos elementos básicos para uma interação do tipo dialógica entre usuário e programa, com foco no próposito dessa interlocução. Elementos essenciais para tratar o outro interlocutor desse diálogo, que é o programa, estão na Seção 5.4.

5.1 Visão peirciana do formalismo matemático

Para Peirce, a matemática se ocupa de duas atividades distintas e complementares: (1) determina hipóteses que são independentes de qualquer coisa do mundo real e (2) traça as consequências necessárias dessas hipóteses (CP 3.559, 1898). Essa visão é mais abrangente que qualquer das visões filosóficas de todas as escolas matemáticas que vimos e se aplica a qualquer uma delas: todas determinam hipóteses independentes do mundo real (algumas incompatíveis com o pensamento peirciano) e traçam as consequências necessárias delas. Vamos prosseguir analisando a escola formalista porque essa escola é a mais abrangente e a mais adotada pelos matemáticos em geral.

A principal característica da escola formalista da matemática é a necessidade de eliminar os significados intuitivos dos conceitos e estabelecer um sistema de símbolos cujas relações refletem as relações entre os elementos dos conceitos iniciais. Só assim, de acordo com o pensamento dessa escola, a veracidade de todas as demonstrações estará atrelada à veracidade dos elementos iniciais. Kauffman (2001, p. 85) nos lembra que uma das razões para extrair da estrutura tudo o que não é essencial está em que o padrão remanescente torna-se pequeno e, não obstante, poderoso: encaixa-se em muitos contextos, torna-se uma “chave para muitas portas”.Tal multiplicidade de usos é uma das características que torna a matemática uma ferramenta poderosa.

O que vemos na construção da axiomática abstrata na escola formalista é a construção de um diagrama, composto de elementos primitivos representados por símbolos, e de regras para a construção de novos símbolos. A principal característica desse diagrama, que constitui o

conjunto de hipóteses matemáticas das quais serão derivadas as necessárias consequências, é que as relações entre os símbolos e as regras: (1) são formais, ou seja, independentes do significado, e (2) captam a essência das relações entre os elementos hipotéticos que a geraram, a axiomática material. Os elementos primitivos da axiomática abstrata formam o legissigno que determina quais inferências são válidas dentro desse corpo de conhecimento.

Mas o processo de investigação matemática não é decorrente da atividade de seguir regras determinadas em uma axiomática abstrata. A investigação matemática, cujo resultado é por excelência a demonstração de um teorema, é um processo de raciocínio que envolve abdução, dedução e indução, e cujo efeito derradeiro, quando se chega a algum, é uma mudança de hábito, ainda que somente um hábito de inferência. Se as razões para essa mudança de hábito têm valor matemático, então devem poder ser explicitadas na forma de um raciocínio dedutivo no produto final da pesquisa matemática, a demonstração do teorema.

A demonstração de Gödel faz uso da característica formal do método axiomático abstrato e, com a numeração de Gödel, cria condições para submeter o processo de inferência da aritmética às próprias regras determinadas para a aritmética. Os enunciados ganham o poder de se referenciar a si mesmos, lembra-nos Kauffman (2001, p. 107). As relações entre os passos das inferências passam a ser relações entre números de Gödel. Gödel conclui com isso que as regras da axiomática abstrata da aritmética não são suficientes para determinar todas as verdades que podem ser expressas em enunciados formados de acordo com as regras de formação de proposições da aritmética.

À luz da moldura filosófica de Peirce, a escola formalista chama a atenção inicialmente pela eliminação, nas demonstrações, da "dependência do significado intuitivo" dos conceitos utilizados. O que isso quer dizer, traduzido em termos peircianos? Começamos por lembrar que não pode haver eliminação do significado. As demonstrações são compostas por símbolos no papel, e esses símbolos dão origem, cada um, a seu próprio interpretante. Uma característica desses interpretantes é que podem ser inteiramente definidos à luz dos axiomas e das regras de cálculo do sistema. Ou seja, o único "significado intuitivo" a ser levado em conta é o dos axiomas. Todos os demais conceitos podem ser construídos a partir da aplicação das regras de cálculo aos axiomas, regras de cálculo cuja aplicação não depende nem sequer da compreensão dos significados dos axiomas. É isso que quer dizer eliminar a dependência do significado intuitivo nas demonstrações. Vale dizer que, definidos os símbolos à luz dos axiomas, a verificação da correção das demonstrações não depende da compreensão do significado intuitivo dos símbolos, mas somente da compreensão da aplicação das regras de cálculo ao texto — o representamen — dos símbolos.

Por se basearem apenas nos textos dos próprios símbolos, nessas demonstrações não há espaço para a vagueza, porque nelas não interessa o único elemento que poderia ser vago,

o elemento que não está presente no texto da demonstração, que é o significado intuitivo dos símbolos. Essas demonstrações são sempre deduções, ou seja, nelas as conclusões já estão necessariamente embutidas nas premissas, cabendo às demonstrações apenas explicitar essas relações. Essa eliminação do significado intuitivo da demonstração resulta na desconexão do processo de investigação matemática do seu produto final, que é a demonstração.

Como vimos, em Peirce o processo de investigação conta com os três tipos de raciocínio. Na investigação matemática o raciocínio abdutivo possibilita o surgimento de conjeturas matemáticas. Por exemplo: "todo número natural ímpar pode ser escrito como a subtração de dois quadrados perfeitos de números naturais". Essas conjeturas muitas vezes são encontradas por acaso, e fortalecidas por um método que é indutivo, no qual se busca, principalmente, um contra-exemplo. No nosso caso,

$$\begin{aligned} 1 &= 1^2 - 0^2 = 1 - 0 \\ 3 &= 2^2 - 1^2 = 4 - 1 \\ 5 &= 3^2 - 2^2 = 9 - 4 \\ 7 &= 4^2 - 3^2 = 16 - 9 \\ &\vdots \end{aligned} \tag{5.1}$$

O raciocínio dedutivo tem papel preponderante na construção da demonstração, mas não só porque a demonstração é uma dedução. Alguns caminhos a percorrer para obter a demonstração, por exemplo, podem ser deduzidos a partir de outras demonstrações, ou a partir de características dos próprios exemplos obtidos. No nosso caso vemos, indutivamente, que os números ímpares são resultados da subtração de quadrados de números consecutivos. Podemos tentar deduzir as características desse tipo de subtração. Se k é um número natural qualquer, seu sucessor é dado por $k + 1$. Estudemos a subtração de seus quadrados:

$$\begin{aligned} (k+1)^2 - k^2 &= \left(k^2 + 2k + 1\right) - k^2 \\ &= 2k + 1 \end{aligned} \tag{5.2}$$

Assim, sabemos que a subtração do quadrado de dois números consecutivos sempre resulta num número ímpar, já que o número $(2k + 1)$ é obrigatoriamente ímpar. Resta saber: será que todos os números ímpares podem resultar da subtração de dois quadrados? Observando a dedução feita em 5.2, descobrimos que dado um número natural ímpar, x, para encontrar os quadrados perfeitos a^2 e b^2 que subtraídos um do outro resultam x basta resolver a equação

$$\begin{aligned} x &= 2k + 1 \Rightarrow \\ k &= \frac{x-1}{2} \\ b &= k = \frac{x-1}{2} \\ a &= b + 1 = \frac{x-1}{2} + 1 \\ a^2 - b^2 &= x \end{aligned} \tag{5.3}$$

o que se demonstra calculando o valor de $a^2 - b^2$:

$$\begin{aligned} a^2 - b^2 &= \left(\frac{x-1}{2} + 1\right)^2 - \left(\frac{x-1}{2}\right)^2 \\ &= \left(\frac{x-1}{2}\right)^2 + 2\left(\frac{x-1}{2}\right) + 1 - \left(\frac{x-1}{2}\right)^2 \\ &= \cancel{\left(\frac{x-1}{2}\right)^2} + \cancel{2}\left(\frac{x-1}{\cancel{2}}\right) + 1 - \cancel{\left(\frac{x-1}{2}\right)^2} \\ &= (x-1) + 1 \\ &= x \end{aligned} \tag{5.4}$$

Assim, demonstra-se que qualquer número ímpar pode ser obtido a partir da subtração de dois quadrados perfeitos. As Equações 5.3 mostram como calcular os números a e b que, elevados ao quadrado e subtraídos um do outro (o maior do menor), resultam em um número natural ímpar x dado. E as Equações 5.4 demonstram que os valores de a e b calculados pelas Equações 5.3 realmente resultam no número ímpar inicial. Isso demonstra que qualquer número natural ímpar pode ser calculado como a subtração de dois quadrados perfeitos. Essa demonstração pode ser escrita utilizando uma linguagem inteiramente formal na qual cada símbolo pode ser interpretado à luz dos axiomas e do cálculo da aritmética.

O resultado pode ser testado calculando dois números cujos quadrados subtraídos um do outro resultam num número ímpar dado. Seja o número ímpar 11. De acordo com nossas deduções, os números a e b cujos quadrados subtraídos resultam 11 são dados por:

$$\begin{aligned} x &= 11 \\ b &= \frac{x-1}{2} = \frac{11-1}{2} = 5 \\ a &= b + 1 = 6 \\ a^2 - b^2 &= 6^2 - 5^2 \\ &= 36 - 25 \\ &= 11 \end{aligned}$$

Na demonstração, que é o produto final da investigação, não aparece o processo que inicia na conjuntura, nem a experimentação mental subsequente. É natural, já que o esforço da escola formalista é justamente o de extirpar a influência do significado das demonstrações. Do ponto de vista peirciano, o formalismo matemático minimiza ou elimina da demonstração os fenômenos de primeiridade e terceiridade que levaram a ela. A influência da terceiridade é minimizada: ela aparece somente nas leis que regem as transformações formais dos símbolos. A influência da primeiridade, que foi essencial no levantamento das conjeturas iniciais (que resultaram nas Equações 5.1) e das conjeturas a respeito dos caminhos possíveis para a evolução da demonstração (que nos levou à Equação 5.2, que é uma dedução), também desaparece no produto final que é a demonstração.

Pode-se conjeturar que os resultados apontados pelos teoremas de Gödel sejam uma consequência desse procedimento de extirpar o significado intuitivo do processo de demonstração. De fato, a numeração de Gödel só pode ser considerada válida do ponto de vista filosófico porque atribuir um número único a uma equação lógica, e em seguida afirmar que as operações sobre a equação original correspondem a transformações numéricas sobre o número, só é possível se tais transformações da equação dependerem somente da forma da equação, tal como o número associado a ela. Foi isso que permitiu a criação de algo que funciona com uma frase que se referencia a si mesma dentro do formalismo da matemática. Uma consequência disso é que, embora o primeiro passo da dedução de Gödel (que vimos na p. 82) seja criar a sentença G que representa o enunciado "a fórmula G não é demonstrável", o fato de que G não é demonstrável só pode ser enxergado a partir de um ponto de vista fora da axiomática. Dentro da axiomática, trata-se da descrição de uma relação entre números inteiros.

Máquinas de Turing

Vimos na Seção 4.2 algumas semelhanças que há entre a demonstração de Turing e a prova de Gödel. As máquinas de Turing automáticas são tais que seus mecanismos podem ser descritos nos dados que estão à sua disposição, numa forma de recursão, nas máquinas de Turing universais. Do ponto de vista semiótico, seu funcionamento é semelhante ao de qualquer axiomática abstrata: baseia-se na secundidade determinada pelas regras contidas na tabela de configurações-*m* e no que chamamos de configuração completa (conteúdo da fita, quadrado e estado atuais etc). Essa descrição determina qual o próximo movimento, se há um próximo movimento, de maneira compulsiva: não pode ser diferente disso. Sua ação é, por assim dizer, compelida diretamente pelas suas regras de ação.

Do mesmo modo que nas axiomáticas abstratas, o funcionamento da máquina de Turing automática, universal ou não, não depende de interpretação nem de apelo ao significado intuitivo do que quer que seja. Nela também a ação de primeiridade é eliminada, e a da terceiridade minimizada ao conjunto de regras definidas pelas configurações-*m* da máquina. Na demonstração de Gödel, a sentença G só é vista como não demonstrável a partir de um ponto fora da aritmética; o fato de que uma máquina de Turing é universal também só pode ser visto a partir de um ponto fora da máquina: a máquina universal em si apenas segue um conjunto de configurações-*m*.

Essa característica que é a independência de primeiridade e determinação estrita das leis de funcionamento é que permitem que as máquinas de Turing automáticas universais inspirem a criação de objetos como os computadores eletrônicos. O fato de a máquina de Turing ser capaz de alterar sua fita torna o computador eletrônico uma máquina cujo comportamento muda sem que mude o seu mecanismo. Essa mudança de hábito independente do mecanismo tem se mostrado útil no computador — é o que, em última análise, permite que seu comportamento mude conforme o programa que executa.

É importante lembrar que essas características das máquinas de Turing automáticas não se aplicam nem às máquinas de Turing de escolha, nem às máquinas-oráculo. Tais máquinas exibem comportamento igual ao das máquinas automáticas somente enquanto não encontram alguma configuração-*m* na qual aguardam uma ação externa, quer de um oráculo, quer de um outro operador. Podemos explorar essas características específicas das máquinas de escolha e das máquinas-oráculo nos computadores eletrônicos.

5.2 Irresolução em computadores eletrônicos

Na Seção 3.8.2 vimos uma conceituação peirciana de determinação, indeterminação e vagueza. Grosseiramente, signos cujo interpretante é determinado são indiciais, são signos de secundidade; a indeterminação corresponde à generalidade, na terceiridade, e a vagueza à possibilidade, na primeiridade. Vamos introduzir o conceito de *irresolução* como o oposto de determinação, ou seja, irresolução correspondendo a vagueza ou indeterminação. Nesse sentido, parece estranho falar em irresolução em computadores eletrônicos. Vejamos como ela aparece, começando pela irresolução na própria axiomática abstrata da aritmética.

5.2.1 Irresolução dentro da axiomática da aritmética

Na axiomática abstrata da aritmética é possível encontrar diferentes graus de indeterminação. Tomemos parte das Equações 5.1 e 5.2 como exemplos:

$$2^2 - 1^2 \tag{5.5}$$

$$(k+1)^2 - k^2 \tag{5.6}$$

A Equação 5.5, tomada da lista de Equações 5.1, designa uma operação bem definida entre dois números bem definidos. Podemos apontar claramente qual número natural corresponde a ela: o número 3, que é um conceito cuja compreensão deriva inteiramente da compreensão dos axiomas da aritmética. No caso, tomando os axiomas de Peano[1] para a aritmética, é o sucessor do sucessor do sucessor do número 0. Já a Equação 5.6 não corresponde a um único número; ela permite, do modo mostrado no conjunto de Equações 5.2, a seguinte derivação, inteiramente baseada na axiomática abstrata:

$$x = (k+1)^2 - k^2 \Rightarrow x = 2k + 1 \tag{5.7}$$

Trata-se, rigorosamente falando, de uma transformação algébrica que revela de forma mais evidente que a subtração de dois quadrados de números consecutivos resulta num número ímpar. Essa transformação algébrica é uma dedução que não traz uma ideia nova, apenas explicita uma ideia que não é evidente no formato da Equação 5.6. O conceito de número ímpar também tem

[1] Giuseppe Peano (1858-1932), matemático italiano.

uma definição precisa a partir dos axiomas: um número ímpar é o sucessor de um número par, que por sua vez ou é o número 0, ou é o sucessor do sucessor de um número par.

Assim, as Equações 5.5 e 5.6, apesar de corresponderem a elementos perfeitamente definíveis em termos dos axiomas da aritmética, apresentam diferentes irresoluções. Uma corresponde a um único número natural, sendo determinada, pois aponta uma única posição definida na sequência de números. A outra corresponde a um conjunto infinito de números naturais, os ímpares, sendo indeterminada, pois não aponta um número específico, mas um conjunto geral definido por uma lei.

Há uma outra forma de irresolução dentro da matemática: as conjeturas. Podem ser expressas na linguagem lógica formal, mas talvez não possam ser derivadas a partir da axiomática abstrata, em mais de um sentido. Toda conjetura pode se encaixar em três situações: é verdadeira, é falsa, ou é indecidível. Se pode ser derivada da axiomática abstrata, é verdadeira, se sua negação pode ser derivada dessa axiomática, é falsa. Se não é possível derivá-la, nem sua negação, da axiomática abstrata, então ela é indecidível, caso em que ou a conjetura ou sua negação podem ser tomadas como axioma, estendendo assim a axiomática. Vale ressaltar que, no caso de ser indecidível, o fato de não poder ser derivada da axiomática tem de ser derivado da axiomática, embora a conjetura em si não o possa ser. Ou seja, a conjetura só pode ser considerada indecidível se houver uma prova matemática de sua indecidibilidade. O propósito de formalizar a situação das conjeturas à luz da axiomática é um dos fatores impulsionam o desenvolvimento da matemática.

Tomemos então uma conjetura. Abaixo, o signo M designa um conjunto de números representados pelo signo m conforme definido pelas equações; nelas, a Equação 5.8 define o significado de "$P(x)$", que quer dizer "x é um número primo":

$$(\forall x \in \mathbb{N})\Big(P(x) \Rightarrow (\nexists y, z \in \mathbb{N})(x = yz \wedge y < x \wedge z < x)\Big) \tag{5.8}$$

$$M = \Big\{m \in \mathbb{N} \mid (\exists k \in \mathbb{N})(\nexists n, q \in \mathbb{N})\Big(m = 2k \wedge P(n) \wedge P(q) \wedge m = n + q\Big)\Big\} \stackrel{?}{=} \varnothing \tag{5.9}$$

Na Equação 5.9, M define o conjunto de todos os números naturais pares que não são a soma de dois números primos. A conjetura, designada pelo símbolo "$\stackrel{?}{=}$", de que tal conjunto é vazio (símbolo "$\varnothing$") é conhecida como conjetura de Goldbach.

Há uma diferença importante entre o tipo de indeterminação da Equação 5.6 e o da Equação 5.9. A Equação 5.6 define um conjunto conhecido de números. Isso não ocorre na Equação 5.9, pois até o momento não sabemos nada a respeito do conjunto M; ou, colocando de outra forma, não sabemos se a Equação 5.9 pode ser derivada da axiomática abstrata da aritmética. Há na aritmética, e na matemática em geral, um tipo de irresolução que está associada à indeterminação do número, e uma irresolução que está associada à derivabilidade da equação; podemos afirmar que, nos termos do apresentado na Seção 3.8.2, os elementos do conjunto M,

na Equação 5.9, estão vagamente definidos dentro de nosso conhecimento atual, ao passo que x, na Equação 5.7, é indeterminado.

5.2.2 Irresolução nas máquinas de Turing

Há extensos estudos no campo da computabilidade efetiva. Esses estudos tiveram início com os trabalhos de Gödel (1992, publicado em 1931), Turing (1936, 1939), Church (1936) e outros, prosseguindo nas décadas seguintes (cf. ROGERS, 1967, p. vii–viii). O estudo da computabilidade efetiva traz importantes reflexões a respeito do que pode, ou não, ser calculado utilizando um procedimento efetivo de cálculo, ou seja, do que pode, ou não, ser realizado por uma máquina de Turing automática. Uma dessas reflexões associa a complexidade da descrição matemática de um conjunto à capacidade computacional necessária para enumerar seus elementos.

Há uma forma de classificar os conjuntos de números naturais conforme a complexidade (formalmente definida) das fórmulas que os definem, conhecida como hierarquia aritmética (cf. SOARE, 1987, p. 60). Há uma relação entre a hierarquia aritmética de um conjunto e a capacidade computacional necessária para determinar se um número qualquer pertence ou não ao conjunto; essa relação é estabelecida pelo teorema de Post (cf. SOARE, 1987, p. 64). Deste teorema se depreende, resumidamente, que máquinas de Turing automáticas não são capazes de determinar o pertencimento de um número a conjuntos cuja descrição possui uma complexidade maior que um certo grau. Para essa determinação seriam precisas máquinas-oráculo (definidas na p. 88). Lembrando que há uma hierarquia de máquinas-oráculo — uma máquina-oráculo não pode ser oráculo de si própria, obrigando portanto que imaginemos outra máquina-oráculo "superior" para decidir seu resultado —, temos que essa hierarquia acompanha a hierarquia aritmética: quanto mais complexa a descrição do conjunto, mais poderosa deve ser a máquina-oráculo necessária para definir se um dado número pertence a ele.

Portanto, os procedimentos efetivos para determinar certas verdades matemáticas a respeito de certos conjuntos, cuja descrição é progressivamente mais complexa, precisam contar com máquinas-oráculo: máquinas de Turing que possuem uma certa configuração-m cuja ação não pode ser efetivamente determinada, pois trata-se da resposta de um oráculo. Essa é uma resposta correta para uma pergunta que não pode ser resolvida através da utilização de uma máquina automática. Essa é uma irresolução que não é da máquina de Turing automática, mas da máquina-oráculo (que também é uma máquina de Turing). As máquinas-oráculo não são nem sequer conjeturas. Não se espera que elas existam. Seu papel é evidenciar a necessidade de um procedimento não computável que encaminha uma questão que o procedimento "mecânico" da máquina automática não é capaz. Sob a óptica peirciana, fica claro que há descrições de conjuntos, um fenômeno de terceiridade, que não podem ser alcançados pela atuação mecânica, de secundidade, das máquinas de Turing automáticas. O estudo da computabilidade efetiva pre-

encheu essa lacuna com uma máquina de Turing, a máquina-oráculo, que realiza um movimento cuja natureza, em princípio, ignoramos.

Do ponto de vista de uma máquina automática, uma máquina-oraculo não é diferente de uma máquina de escolha. Em ambas, tudo acontece como se uma máquina automática chegasse a uma configuração-*m* na qual aguarda uma intervenção. Que a intervenção levará a computação a uma resposta correta é um fato que só pode ser observado de fora da máquina automática — do mesmo modo que o fato de que sentença G na prova de Gödel não ser demonstrável só pode ser enxergado a partir de um ponto fora da axiomática abstrata da aritmética.

Uma vez que pretendemos, mais adiante, sugerir caminhos para estender a forma de uso dos computadores eletrônicos, de modo a permitir que lidem de forma mais autônoma com essas irresoluções, e, como vimos na Seção 4.3.2, os computadores eletrônicos se comportam como máquinas de escolha, vamos agora estudar as irresoluções nas máquinas de Turing tomando o paradigma das máquinas de escolha. Nesse modelo a tabela de configurações-*m* em funcionamento automático só ocorrem entre duas interações com o usuário. De fato, podemos visualizar uma simplificação dos eventos numa máquina de escolha como aproximadamente obedecendo a sequência:

$$a_0 \to c_0 \to a_1 \to \ldots \to a_i \to c_i \to a_{i+1} \to c_{i+1} \to \ldots \to a_{n-1} \to c_{n-1} \to a_n \tag{5.10}$$

Nela, as letras a indicam as ações automáticas, performáveis por uma máquina de Turing automática, sendo a_0 a inicialização da máquina de escolha e a_n sua finalização, e as letras c indicam as escolhas feitas pelo usuário. Trata-se de uma simplificação: de fato, a escolha c_0 pode determinar qual ação será executada no lugar de a_1, e essa escolha, como qualquer outra escolha, pode alterar toda as ações e escolhas subsequentes. Além disso, uma mesma escolha pode disparar mais de uma ação; e, adicionalmente, uma ação possível é justamente carregar e executar uma outra máquina de escolha, que tem suas próprias ações e escolhas. Vê-se que a árvore de possibilidades de interação antes que essas ocorram é mais complexa que a sequência mostrada na Equação 5.10.

Não obstante, depois de definidas todas as escolhas entre duas ações numa máquina de escolha — uma definição que pode ser feita durante a interação com a máquina —, esse específico conjunto de escolhas e ações pode ser executado por uma máquina de Turing automática. Para isso, basta que cada configuração-*m* que aguarde a interação do usuário na máquina de escolha seja substituída por uma configuração-*m* na máquina automática cuja ação seja realizar a escolha feita. Esquematicamente, depois que as escolhas c_i foram feitas, é possível recriar a interação numa máquina de Turing inteiramente automática cujas configurações-*m* são as da máquina de escolha acrescidas das configurações-*m* a_{c_i}, que são as configurações cuja ação é realizar a escolha c_i:

$$a_0 \to a_{c_0} \to a_1 \to \ldots \to a_i \to a_{c_i} \to a_{i+1} \to a_{c_{i+1}} \to \ldots \to a_{n-1} \to a_{c_{n-1}} \to a_n \tag{5.11}$$

A Equação 5.11 mostra e sequência de ações da máquina de Turing automática que performa as mesmas ações da máquina de escolha da Equação 5.10, desde que todas as escolhas c_i tenham sido determinadas. Cumpre notar que, embora seja sempre possível, em teoria, realizar essa substituição depois que se sabem todas as escolhas, nem sempre é possível conhecer quais foram as escolhas. Muitos programas interativos, como por exemplo alguns jogos eletrônicos, utilizam informação verdadeiramente aleatória para realizar escolhas não computáveis. Nesses casos pode ser muito difícil rastrear as escolhas feitas aleatoriamente.

Doravante procuraremos não mais nos referir a tabelas de configurações-*m*, ou máquinas de escolha, mas a programas de computador compostos de ações e escolhas; são conceitos equivalentes, mas os últimos soam mais familiares nos exemplos a seguir. De modo geral pode-se tanto programar interativamente uma máquina de escolha, ou seja, programar um programa — tomando "programar" com o sentido de mudar hábitos — quanto executar, interativamente ou não, um programa de computador.

Vimos na seção 5.2.1 que a descrição dos componentes de um conjunto pode guardar em si uma irresolução. Esta é caracterizada como a indeterminação da generalidade na Equação 5.7, um fenômeno de proeminente terceiridade, ou a vagueza da possibilidade na Equação 5.9, um fenômeno de proeminente primeiridade. Assim como há uma espécia de progressão na irresolução na axiomática da aritmética, a partir da Equação 5.5, passando pela Equação 5.6 até chegar à Equação 5.9, há também uma gradação de irresolução nos programas de computador.

Assim como a Equação 5.5 determina um único número, pode-se programar um computador para realizar esse cálculo e mostrar o número. Esse programa (chamemo-lo P_1) não possui nenhuma indeterminação, e pode ser inteiramente implementado numa máquina de Turing automática. No nosso modelo abstrato de programação e interação, vale dizer que pode ser programada interativamente, mas sua execução prescinde de interação. Já a Equação 5.6 corresponde a um programa cujo resultado é indeterminado até que se decida um valor para o que corresponde à variável k na equação. Falando rigorosamente, há uma máquina de Turing automática que realiza essa conta. Essa máquina automática funciona tomando um valor escrito na fita da máquina antes do início do movimento como sendo o valor k. Já numa máquina de escolha podemos, por exemplo, implementar um programa (chamamos P_2) que interage com o usuário, aguardando que este defina o valor que deseja para k, que nesse caso é chamado de *parâmetro* do programa. Uma vez definido esse valor, o programa realiza o cálculo, mostrando o resultado. Esse programa pode realizar as operações como definidas na Equação 5.6 — como a subtração de dois quadrados de números consecutivos —, mas o programador se pode valer da derivação exposta na Equação 5.7 (que é detalhada na Equação 5.2) e implementar a equação $(2k + 1)$, obtendo os mesmos resultados. Curiosamente, uma vez que a derivação da Equação 5.2 é inteiramente formal, é também possível criar um programa que a realiza. Por exemplo, WolframAlpha (WOLFRAM RESEARCH, 2019), disponibilizado online, é um programa no

qual o usuário determina uma equação matemática, e o programa retorna diversos elementos de interesse matemático a respeito da equação, entre elas sua forma simplificada.

Essa característica formal das equações matemáticas permite programas interativos com maior grau de irresolução do número calculado. As planilhas eletrônicas (cf. SPREADSHEET, 2019), por exemplo, apresentam ao usuário uma interface que é uma matriz de células, cada uma com um endereço dado pela coluna e linha em que se encontram. Cada célula pode conter textos, valores numéricos e fórmulas: equações matemáticas, lógicas etc. Essas equações referenciam outras células na planilha, permitindo que o usuário, num mesmo programa interativo, determine quais funções matemáticas serão utilizadas no cálculo e quais dados servirão como parâmetro de entrada para as funções. As planilhas eletrônicas são programas com maior grau de irresolução que os programas que implementam as operações das Equações 5.5 (P_1) e 5.6 (P_2). Efetivamente, são programas nos quais os usuários gozam de grande liberdade para determinar o que querem.

Como se vê, os programas expostos até o momento apresentam uma gradação de irresolução quanto aos números que resultam deles. Percebe-se também diferentes papéis na determinação do número resultante dos programas. O que o programa P_1 faz é inteiramente determinado pelo programador; em P_2 o programador escreve o programa contendo a função, e um usuário que não necessariamente conhece programação pode calcular o valor da equação determinando o valor de k. Nas planilhas eletrônicas, o programador programa uma planilha sem nenhum objetivo específico além do de disponibilizar a planilha para o usuário, que por sua vez determina tanto as equações quanto os parâmetros de cálculo. Aqui ficam evidentes dois papéis: o do programador e o do usuário. Esses papéis podem ser assumidos pela mesma pessoa ou por pessoas diferentes. De um modo geral, o programador, que é o agente que determina o programa em si, determina também quais graus de liberdade o usuário terá à disposição. Quanto mais graus de liberdade, menos determinado é o número gerado (ou a função calculada) pelo programa. Nesse contexto, o programa pode ser visto como um intermediário entre o programador e o usuário, o que leva alguns autores a estudar os programas de computador como meios de comunicação entre programadores e usuários (cf. SOUZA, 2005).

Nos exemplos acima, o programador e o usuário sempre fornecem aos programas construtos formais — números, equações, comandos — que não dependem de interpretação, nem de apelo ao significado intuitivo do que quer que seja, para permitir o avanço da computação, exatamente como nas axiomáticas abstratas e nas máquinas de Turing automáticas. Isso pode ser, de certa forma, depreendido das Equações 5.10 e 5.11. Há, entretanto, técnicas, conhecidas como aprendizado de máquina, que permitem determinar as ações de máquinas de escolha sem a necessidade de fornecer ao programa construtos formais, independentes de interpretação, para determinar a ação da máquina (cf. MACHINE LEARNING, 2019). Por exemplo, na técnica conhecida como rede neural artificial (cf. ARTIFICIAL NEURAL NETWORK, 2019), doravante simplesmente rede neural, a ideia é justamente utilizar a interpretação humana sobre um conjunto

vagamente determinado de dados para capacitar a máquina a realizar interpretações semelhantes, com grau de precisão semelhante à humana. Isso é feito sem que seja preciso fornecer à máquina somente construtos formais independentes de interpretação.

Simplificadamente, as redes neurais funcionam em dois momentos: o aprendizado e a utilização. No aprendizado, milhares de exemplos são apresentados ao programa. Na utilização, o programa avalia um caso novo, ao qual nunca foi apresentado, à luz do que aprendeu durante o aprendizado. Um exemplo: no reconhecimento de imagens, no aprendizado apresenta-se ao programa imagens, em algumas das quais aparece, digamos, um gato, noutras, não. Cada imagem está acompanhada da informação "contém um gato" ou "não contém um gato". Depois do aprendizado, na utilização, o programa está pronto para ser apresentado a uma imagem que nunca "viu" antes e afirmar, com precisão semelhante à humana, se esta contém ou não um gato.

A novidade aqui está na escolha dos dados para aprendizado e nas técnicas utilizadas para tal. É um novo degrau na irresolução do número produzido pela máquina, no qual a construção do processo que calcula o número conta com elementos que não são determinados com exatidão, mas dependem de interpretação. Tudo acontece como se a máquina tomasse emprestada, por assim dizer, a capacidade de discernimento humana.

Na moldura peirciana, as imagens contendo (ou não) gatos constituem ícones, cuja ligação com o objeto se dá por semelhança, um critério que não tem formalização. Por isso, é importante que a associação, por semelhança, entre a imagem e a informação se ela contém ou não um gato seja feita por uma pessoa.

O que o programa faz, durante o aprendizado, é calcular uma grande quantidade de parâmetros. A cada imagem apresentada esses parâmetros são ajustados de modo a maximizar a probabilidade de acerto durante a fase de utilização. Os métodos para esse ajuste são cuidadosamente escolhidos pelas pessoas responsáveis pelo treinamento, e a precisão da utilização é periodicamente aferida durante o processo. A ideia é que esse processo consiga transformar a capacidade humana de identificar gatos em elementos formalmente bem definidos — os parâmetros da rede neural.

Assim, mesmo que não sejam apresentados elementos formais para o programa, esses elementos são transformados, pelo próprio programa, de acordo com um procedimento acompanhado e aferido por pessoas, em parâmetros que serão utilizados nas ações a_i na etapa de utilização.

Apesar de incorporar uma grande irresolução, o programa que implementa a rede neural é, ainda, uma máquina de escolha, na qual continuam valendo as considerações explicitadas nas Equações 5.10 e 5.11. O esquema de aprendizado e utilização da rede neural também pode ser

esquematizado como uma sequência de ações e escolhas como a ilustrada a seguir:

$$a_{ini} \rightarrow c_{Ldata} \rightarrow a_{learn} \rightarrow c_{case_0} \rightarrow a_{use_0} \rightarrow \ldots \rightarrow c_{case_i} \rightarrow a_{use_i} \rightarrow \ldots \rightarrow a_{end} \tag{5.12}$$

Onde:

a_{ini} é a inicialização do programa
c_{Ldata} representa os dados escolhidos para aprendizado
a_{learn} representa a ação de parendizado
c_{case_i} representa um caso de utilização
a_{use_i} representa as ações de reconhecimento do caso de utilzação e
a_{end} são as ações de finalização

Temos que, uma vez determinados todos os elementos de escolha na Equação 5.12, os resultados podem também ser reproduzidos por uma máquina de Turing automática que implementa as mesmas escolhas.

De uma forma geral, vemos que, apesar do alto grau de irresolução na utilização das máquinas de Turing de escolha, essa irresolução tem sido tratada pelo elemento externo à máquina. Embora não seja possível determinar antecipadamente os resultados da execução de um programa interativo com intervenção do usuário pode-se, uma vez determinadas as escolhas, implementar uma máquina de Turing automática que gera os mesmos resultados. O presente trabalho pretende propor modos de ultrapassar essa barreira e incorporar ao computador eletrônico recursos para lidar mais autonomamente com a indeterminação e a vagueza.

5.3 Conceituação da interlocução

Podemos começar agora a utilizar os conceitos expostos até aqui para delinear a implementação de um comportamento autônomo nos computadores eletrônicos. A ideia é permitir que o programa tome por si algumas decisões que seriam tomadas pelo usuário.

Esquematicamente, podemos descrever o processo de decisão do usuário à luz do seu propósito. Durante a execução de um programa interativo espera-se que, em algum momento decisivo c_d surja, no usuário, o propósito de que o programa execute a ação a_p. Dentro do modelo que estamos utilizando teríamos:

$$a_{ini} \rightarrow \ldots \rightarrow a_{d-1} \rightarrow c_d \rightarrow a_{d_0} \rightarrow c_{d_0} \rightarrow \ldots \rightarrow a_{d_i} \rightarrow c_{d_i} \rightarrow \ldots \rightarrow a_{d_n} \rightarrow c_{d_n} \rightarrow a_p \rightarrow \ldots \tag{5.13}$$

O diagrama mostra a sequência de escolhas e ações do usuário depois que todas as escolhas foram conhecidas, daí ser linear. As ações a_{d_i} e as escolhas c_{d_i} (das quais, convencionemos, a escolha c_d faz parte) representam o que ocorreu desde o momento da decisão c_d até o atingimento da ação a_p desejada, supondo que o usuário logrou chegar a ela.

Entretanto, no momento de escolha c_d, no qual decide a ação a_p a que quer chegar, essa sequência apresenta um certo grau de irresolução. O usuário pode estar, por exemplo, usando uma planilha eletrônica e ter decidido que deseja calcular um certo valor. A ação a_p seria, digamos, calcular o desvio padrão de uma amostra de valores digitados na planilha. Se o usuário não está familiarizado a essa operação, é razoável que faça tentativas de diferentes naturezas: por exemplo, consultar o sistema de ajuda da planilha procurando por uma fórmula pronta que realize o cálculo, ou utilizar recursos que já conhece para calcular valores intermediários que serão utilizados na equação final.

Dar ao computador autonomia na tomada de decisões significa delegar ao computador quais escolhas levarão a computação da decisão em c_d à ação a_p; esquematicamente corresponde a substituir a ação c_d por uma ação c'_d que vai disparar uma ação autônoma a'_{d_0}, com o objetivo de performar a ação a_p desejada, ou ao menos aproximar a computação dessa ação, num processo que passa a ser dialógico entre usuário e programa, de acordo com o diagrama a seguir.

$$a_{ini} \to \ldots \to a_{d-1} \to c'_d \to a'_{d_0} \to c'_{d_0} \to \ldots \to a'_{d_i} \to c'_{d_i} \to \ldots \to a_p \to \ldots \tag{5.14}$$

Na Equação 5.14 a escolha c'_d é tal que o resultado da ação a'_{d_0} disparada faz parte de um processo de diálogo, como veremos a seguir. Claro fica que as ações a'_i têm como característica incorporar decisões que não são, necessariamente, determinadas:

$$a'_i \Leftrightarrow a_0 \to c_0 \to \ldots \to a_i \to c_i \to \ldots \to c_{n-1} \to a_n \tag{5.15}$$

Uma ação a'_i corresponde a uma sequência de escolhas c_i do usuário e ações a_i automáticas do programa.

5.3.1 Elementos para o diálogo

Como vimos na teoria peirciana de comunicação, exposta na Seção 3.8.1, no diálogo um interlocutor cria um diagrama mental que contém o outro interlocutor, e calcula que efeito deseja imprimir nesse interlocutor. Escolhe então o signo que julga que vá exercer esse efeito e o apresenta ao outro. Então, para que o diálogo ocorra precisamos conhecer tanto o interlocutor quanto o efeito que desejamos imprimir nele. O propósito do usuário no momento c_d é performar a ação a_p. Então, no nosso caso, o interlocutor é o programa, e o efeito desejado é ação a_p. Falando rigorosamente, numa moldura peirciana qualquer processo de programação ou de utilização dos computadores eletrônicos tem características dialógicas. No momento c_d se inicia o diálogo, e o conjunto de escolhas c_{d_i}, que levarão de c_d a a_p, depende tanto da ação a_p quanto do conhecimento que o usuário tem do programa, o interlocutor. Isso ocorre quer o programa tenha ou não autonomia. Numa moldura na qual o computador eletrônico tem alguma atuação autônoma, as escolhas c'_{d_i} são tais que levam em conta essa autonomia. Como a ideia é estender a autonomia do computador eletrônico, é razoável que dessa extensão decorra um nova forma de interlocução, que exemplificamos a seguir.

A familiaridade do usuário com a ação a_p pode variar. Pode variar também sua familiaridade com o interlocutor, o programa. O conhecimento do propósito e do programa como agente semiótico autônomo afeta o próximo passo do diálogo em mais de uma maneira: a ação a_p pode ser conhecida do usuário, que quer explorar as capacidades do programa. Ou o contrário pode ser o caso: o usuário quer usar as capacidades do programa autônomo para investigar uma ação a_p, que conhece pouco. Além disso, é possível que o usuário esteja equivocado quanto ao que julga conhecer. Isso levará o diálogo para caminhos que não espera. Como em qualquer diálogo entre agentes autônomos, não se pode esperar saber já no primeiro momento como a interação se desdobrará.

Dos elementos do diálogo, o mais complexo é o conhecimento do interlocutor; iniciaremos, portanto, pela menos complexa conceituação de como o propósito pode participar deste diálogo para em seguida nos aprofundarmos na agência semiótica autônoma no computador eletrônico.

Determinando o propósito

O propósito, um fenômeno de proeminente terceiridade, se corporifica na secundidade: as marcas no papel (secundidade) que representam uma palavra não são a palavra (terceiridade). Apagar as marcas não retira a palavra do dicionário. Entretanto, temos conhecimento da palavra ou pelas marcas no papel ou pelo som, fenômenos de proeminente secundidade. Vamos nos valer dessa característica para possibilitar que o computador, como agente autônomo, passe a agir como se tivesse o propósito que é, em última análise, do usuário. A familiaridade do usuário com a ação a_p que deseja performar é muito importante para essa determinação. Seguem alguns exemplos que não formam uma lista exaustiva, mas que ilustram, inspirados nas categorias peircianas, como se pode imaginar uma estratégia de diálogo de acordo com o conhecimento que se tem da ação a_p. Um diálogo real pode contar com uma combinação das estratégias abaixo.

Ação e escolhas determinadas. A ação a_p pode ser, para o usuário, inteiramente determinada, bem como o caminho para chegar a ela. Ou seja, ele não só conhece a ação em si, mas também quais escolhas c_{d_i} devem ser feitas para atingi-la. É um caso em que poderia ele mesmo performar tais escolhas; as razões para delegá-las ao computador são de outra ordem que não o desconhecimento: pode querer economizar seu tempo, ou quer conhecer melhor as habilidades do programa. O início diálogo aqui depende mais do conhecimento que o usuário tem do programa interlocutor que do propósito a_p.

Escolhas indeterminadas. Por outro lado, ação a_p pode ser conhecida pelo usuário, que, mesmo sabendo que é possível performá-la, não sabe quais escolhas c_{d_i} devem ser feitas para chegar a ela. É um caso em que se busca um caminho que chegue a a_p numa situação em que é possível determinar um critério de parada para a investigação. Uma vez que a ação a_p é conhecida, seus efeitos também o são. Portanto, é possível dar ao programa uma

forma de testar se a ação a_p foi ou não performada durante a investigação. Por exemplo, se a ação a_p é "somar dois números", um critério de parada seria "pare ao chegar a uma ação que, dados a ela os números 3 e 5, retorne o número 8". Esse critério não carrega nenhuma irresolução, é um critério inteiramente determinado de parada. A investigação busca quais escolhas c_{d_i} levam a a_p. O que se está investigando é qual caminho percorrer para performar uma ação conhecida; aqui entra o conhecimento que o usuário tem das capacidades do programa, mas de uma maneira diferente da exposta acima, onde todos os elementos necessários para chegar à ação a_p são conhecidos e o programa em si talvez não o seja. O conhecimento que o usuário tem do programa, aqui, pode influenciar, por exemplo, na otimização da solução. Poderia escolher um critério para o programa iniciar a investigação: busca exaustiva, ou tentativas aleatórias de combinações das funções disponíveis existentes. Ao fazê-lo, podemos dizer que está determinando a forma da ação do programa, ou seja, determinando qual hábito ético o programa deve seguir, deixando o raciocínio a cargo do programa. O programa "saberá" quando tiver chegado à inferência procurada (a ação a_p) porque tem um critério inteiramente determinado para julgar quando isso ocorre.

Conhecimento vago. A ação a_p pode ser desconhecida para o usuário, que talvez não saiba sequer se pode ser performada ou não. Um exemplo de ação desse tipo é encontrar um número par que não seja a soma de dois números primos, provando falsa a conjetura de Goldbach, descrita na Equação 5.9. A ideia aqui é utilizar a capacidade autônoma do programa para encontrar um caminho que leve à ação a_p, ou encontrar as razões pelas quais a ação a_p não pode ser performada. Em ambos os casos, sendo a ação apenas uma conjetura, será preciso determinar uma estratégia de investigação. A estratégia pode ser determinada pelo usuário, ou este pode aproveitar a autonomia da máquina para buscar a estratégia. Ao invés de escolher qual tipo de inferência — busca exaustiva, combinação aleatória — e determinar um critério de parada da busca, o usuário poderia direcionar a investigação de modo que o critério de parada não esteja ligado à ação a_p em si, mas a algo que tenha relação com o próprio processo investigativo.

Por exemplo, no caso de investigar a conjetura que diz que todo número impar pode ser escrito como a subtração de dois quadrados, apresentada nas Equações de 5.1 a 5.4, a lista apresentada na Equação 5.1 poderia ser obtida através de uma busca exaustiva cujo critério de parada seria dado pelo número de exemplos. Um caso em que o usuário determina a parada não por um critério ligado à ação procurada, mas por um critério que o leva a um passo intermediário. Não é um critério ligado diretamente à ação procurada, mas a uma conveniência do usuário que, do ponto de vista do programa, não tem razão de ser: é estética. Já a dedução na Equação 5.2 poderia ser obtida através de uma inferência dedutiva, na qual se busca a simplificação da subtração dos dois quadrados. Claramente um hábito de ação. Assim, o usuário que busca uma estratégia de investigação tem à disposição

tanto hábitos estéticos a respeito do resultado da ação — a quantidade de equações — quanto critérios de ação propriamente dita — a inferência de um simplificação. Os graus de liberdade que o usuário tem, ou a vagueza com a qual pode propor uma ação autônoma a'_i, depende da autonomia do interlocutor.

Uma generalização possível para a determinação de um propósito investigativo poderia ser tirada de um texto de Peirce, "The First Rule of Logic" ("A primeira regra da lógica"), de 1898 (PEIRCE, 1998, p. 42–56): o desejo de conhecer a verdade. Num contexto filosófico peirciano, podemos entender a ideia de verdade como um interpretante final. A investigação científica permite que nos aproximemos cada vez mais desse interpretante final, sem nunca entretanto atingi-lo. Paradoxalmente, quanto mais signos se interpõem entre nós e o mundo — ou seja, quanto mais detalhada nossa descrição do mundo — mais distantes estamos do mundo em si. Chegar ao interpretante final é uma tarefa que não tem conclusão: aproximamo-nos dele assintoticamente (cf. CP 8.12, 1871). Mas, quanto maior a semelhança entre os interpretantes dos signos que descrevem o mundo, gerados pela investigação, e o próprio mundo, mais próxima a busca está da verdade. Isso nos dá uma métrica que poderia orientar um programa numa tal busca.

Depreende-se dos exemplos acima que o conhecimento do propósito determina apenas uma parte do processo de diálogo com o computador eletrônico como agente autônomo. Outra parte desse processo depende do conhecimento do próprio interlocutor como agente autônomo, a que nos dedicamos a seguir.

5.4 Conceitos essenciais para compreensão do interlocutor

Vamos conceituar como alguns dos diferentes elementos da agência semiótica apresentados no Capítulo 3 afetam a atuação autônoma de computadores eletrônicos. Aqui, pretendemos destacar pontos importantes para a compreensão da implantação esboçada no capitulo a seguir.

Para que o computador tenha um comportamento autônomo ele mesmo precisa ser capaz de alterar seus próprios hábitos. Embora a máquina de Turing universal seja configurada de uma maneira que possibilita que altere seus próprios hábitos, isso não é suficiente. Como vimos, Turing imaginou o comportamento dessa máquina como o de uma máquina automática: inteiramente determinado, num sentido peirciano. Mesmo uma máquina de Turing universal implementada como uma máquina de escolha depende de escolhas externas c_i para lidar com a irresolução; feitas as escolhas, o comportamento das ações a_i é automático. Autonomia implica que a máquina *altere* seus hábitos de acordo com algum propósito ou critério que não seja inteiramente determinado pelo usuário. Ou seja, este faz escolhas c'_i que disparam ações a'_i que por sua vez atuam como sequências de escolhas c_i e ações a_i determinadas por elas.

Uma das principais questões que surge quando se trata de pensar na autonomia de um computador é como livrá-lo do determinismo imposto pelo seu próprio mecanismo. Como evitar esse determinismo? Em termos semióticos, equivale a perguntar: como uma ideia nova, ou seja, uma inferência abdutiva (única forma de raciocínio capaz de iniciar uma nova ideia, conforme visto na Seção 3.3.1), poderá ocorrer em circunstâncias nas quais todas as inferências são dedutivas?

Pensando um computador como um agente imerso em um círculo funcional, surgiu a ideia de que o único resultado possível de uma abdução do computador é uma sequência aleatória de bits, que pode ser interpretada como uma sequência de números aleatórios. Essa ideia evoluiu e foi implementada num programa que criava programas (os programas-produto) tais que resolviam problemas vagamente definidos pelo programador original, que entretanto não tinha como saber quais programas-produto seriam produzidos pela programa (cf. GAZONI, 2016). Para isso o programa original utilizava o sorteio pseudoaleatório. Se desejado, esse sorteio poderia ser substituído por sorteio efetivamente aleatório, disponibilizado pelo coletor de entropia dos modernos computadores, ou por um gerador de números aleatórios baseado em hardware, garantindo que as soluções geradas não decorrem de um procedimento determinado.

A solução proposta acima pode ser filosoficamente questionada. Como vimos, para Peirce não há pensamento desconectado de outro; e um número aleatório num computador é, teoricamente, desconectado de qualquer coisa. Aparentemente não há, no sorteio de um número aleatório, algo como um instinto abdutivo, ou como uma introjeção inconsciente de hábitos que levam a uma inferência abdutiva, ambos mecanismos que levariam à abdução ligada a pensamentos anteriores, como vimos na Seção 3.6. Portanto, uma sequência aleatória de bits não seria resultado um raciocínio abdutivo genuíno, embora possa ser considerada como resultado de uma primeiridade genuína.

Poder-se-ia contrapor a esse questionamento considerações do tipo: há o fato de ser um *número* aleatório (e não um evento aleatório qualquer), o que ocorre por estar no círculo funcional do Umwelt do computador (que só lida com sequências de bits, interpretadas como números), podendo por isso ser considerado "instintivo", dado pelo "organismo" do agente. Poder-se-ia adicionar que outras capacidades — que serão implementadas como parte do círculo funcional do agente que estaremos programando — modulariam esse número aleatório de modo que seu resultado seria semelhante ao resultado do raciocínio abdutivo tal como estamos habituados a imaginá-lo.

E poderíamos, ao invés de contrapor o questionamento, nos contentar com aceitar que o computador, como agente, é essencialmente diferente dos seres vivos. Essas são questões filosóficas que poderão ser desenvolvidas em estudos futuros. No momento procuramos determinar o modo pelo qual um agente semiótico autônomo implementado num computador

escapa do determinismo mecânico do meio em que existe, e propomos que isso se dê através do uso de números aleatórios. Como vimos, a geração de números genuinamente aleatórios não é uma função computável, não podendo ser executada por uma máquina de Turing automática de computar. Daí, a utilização de números aleatórios só poder fazer parte do repertório dos computadores eletrônicos, reais, pois se baseiam em processos físicos e não em abstrações. Antes de prosseguirmos, entretanto, convém fazer algumas considerações a respeito do raciocínio abdutivo em computadores eletrônicos.

5.4.1 Abdução em computadores hoje

Dentro da inteligência artificial existe, hoje, um método de inferência computacional ao qual chamam "abdução". Embora de inspiração peirciana, não corresponde, essencialmente, ao raciocínio abdutivo a que nos referimos no presente trabalho. O raciocínio dedutivo é um fenômeno da secundidade, a categoria da necessidade mecânica. Trata-se da categoria característica de programas de computador que implementam funções computáveis. A abdução, por ter a primeiridade como categoria característica, não pode ser implementada em computador por funções computáveis. Na moldura conceitual da filosofia de Peirce nenhuma função computável, nem nenhum fenômeno de secundidade pura, podem implementar abdução. Ou seja, caso desejemos implementar abdução estritamente peirciana numa máquina, será preciso contar com recursos que estão fora da máquina para dispor dos fenômenos das categorias necessárias. Não obstante, dentro da pesquisa em inteligência artificial existe esse método, aparentemente consagrado, de inferência computacional que afirma implementar o raciocínio abdutivo. Analisaremos esse método, que chamaremos *abdução computacional*, a seguir.

Abdução computacional

Uma busca rápida em artigos publicados entre Janeiro e Agosto de 2018 nos mostra aplicações em inteligência artificial que afirmam utilizar a inferência abdutiva para inúmeros fins, tais como visão computacional (SUCHAN *et al.*, 2018) e otimização de aprendizado de máquina (YAMAMOTO; ONISHI; TSURUOKA, 2018). Outros trabalhos propõem estender a inferência abdutiva: Juba, Li e Miller (2018) estendem a proposta de Juba (2016) para o aprendizado da plausibilidade das hipóteses propostas por inferência abdutiva; já Aiguier *et al.* (2018) propõem novos "operadores de raciocínio abdutivo" em lógicas arbitrárias, um trabalho teórico, como é também o texto de Bloch *et al.* (2018), que estudam a morfologia da dinâmica do conhecimento e derivam operadores não só para abdução, mas também para outros aspectos da cognição como a revisão de crenças e a fusão de crenças. A abdução também aparece em propostas de novas arquiteturas de inferência artificial, como no trabalho de Dai *et al.* (2018), que combinam redes neurais com inferência abdutiva para criar uma máquina neuro-lógica (*the Neural-Logical Machine*). Não somente a abundância de trabalhos, mas a extensão de sua utilização leva a crer que a abdução peirciana é extensivamente utilizada computacionalmente, mas não é bem assim.

A abdução computacional é uma área da inteligência artificial que se encarrega de encontrar o melhor conjunto de hipóteses explanatórias para uma observação. Essa é, de fato, uma das características da abdução em Peirce, mas não é a única. Uma definição formal da abdução computacional é apresentada por Kakas, Toni e Kowalski (1993), reproduzida aqui:

> Dado um conjunto T de sentenças (a apresentação de uma teoria), e uma sentença G (observação), numa primeira aproximação a tarefa abdutiva pode ser caracterizada como o problema de encontrar um conjunto Δ de sentenças (explicação abdutiva para G) tal que:
> (1) $T \bigcup \Delta \vDash G$,
> (2) $T \bigcup \Delta$ é consistente.
> Essa caracterização da abdução é independente da linguagem na qual T, G e Δ são formulados.[2] (KAKAS; TONI; KOWALSKI, 1993, p. 721)

Segundo essa definição formal a abdução computacional consiste em encontrar um conjunto de sentenças (Δ) que, adicionadas ao conjunto de sentenças de uma teoria T já existente sem com isso criar inconsistência, formam um conjunto de sentenças que é uma explicação para o fenômeno observado G.

Os autores admitem que essas noções não capturam de fato a noção peirciana (KAKAS; TONI; KOWALSKI, 1993, p. 721). Além disso, segundo eles, as explicações que farão parte do conjunto Δ "são frequentemente restritas a uma classe pré-determinada de sentenças específicas de cada domínio, chamadas de *abdutíveis*".[3] As razões para isso são melhor explicadas por Flach e Kakas (2000, p. 19):

> É importante aqui enfatizar que a restrição da hipótese de abdução aos predicados abdutíveis não é incidental nem computacional, mas possui uma razão representacional mais profunda. Ela reflete a abrangência relativa do conhecimento do domínio do problema contido em T. Os predicados abdutíveis e as fórmulas abdutivas permitidas assumem o papel de 'portadores-de-respostas' para os objetivos-problema que queremos definir para nossa teoria. Nesse sentido eles tomam o lugar da variável lógica como portadora de respostas no caso em que o raciocínio dedutivo é usado para resolução de problemas. Como resultado isso significa que a forma da hipótese abdutiva depende fortemente da teoria T em particular à mão, e o modo que escolhemos representar nela o domínio do nosso problema.
>
> Tipicamente, as fórmulas abdutíveis permitidas são adicionalmente restritas a formas lógicas simples tais como conjunções de literais abdutíveis básicos ou existencialmente quantificados. Embora essas restrições adicionais possam

[2] Given a set of sentences T (a theory presentation), and a sentence G (observation), to a first approximation, the abductive task can be characterised as the problem of finding a set of sentences Δ (abductive explanation for G) such that:
(1) $T \bigcup \Delta \vDash G$,
(2) $T \bigcup \Delta$ is consistent.
This characterisation of abduction is independent of the language in which T, G and Δ are formulated.

[3] [...] explanations are often restricted to belong to a special pre-specified, domain-specific class of sentences called *abducible*.

> ser em parte motivadas por considerações computacionais, é novamente importante apontar que elas são possibilitadas apenas pela abrangência relativa da representação do domínio de nosso problema na teoria T. Assim, o caso da abdução simples — na qual as hipóteses abdutíveis são fatos básicos — ocorre exatamente porque a representação do domínio do problema em T é suficientemente completa para o permitir.[4]

Apesar dessas limitações, dez anos depois do artigo de Kakas, Toni e Kowalski — do qual extraímos a definição formal da abdução computacional —, Denecker e Kakas (2002) registram o notável desenvolvimento do campo da abdução computacional assim caracterizada, apontando os desafios de então.

Ocorre que tal caracterização da abdução não corresponde inteiramente à ideia peirciana. Uma característica importante do raciocínio abdutivo peirciano é que ele produz uma ideia nova, ainda que, como vimos em seções anteriores, a abdução de certa forma participe do instinto (Seção 3.5.2) ou decorra de uma introjeção inconsciente de ideias (Seção 3.6). Já o raciocínio abdutivo computacional, como caracterizado aqui, embora de fato resulte numa explicação para uma observação à luz de uma teoria — uma das características do genuíno raciocínio abdutivo — não é capaz de propor uma explicação nova. Essa abdução computacional consiste em extrair de um conjunto já determinado de predicados abdutíveis aqueles que formarão a melhor hipótese para explicar uma certa observação à luz de uma dada teoria. Não é problema trivial do ponto de vista computacional: Bylander *et al.* (1991) já apontam esse problema como intratável computacionalmente, daí o enorme interesse em procurar uma solução computacionalmente útil. Entretanto, a natureza do problema que resolve é tal que é possível propor algo que, em princípio, parece uma contradição em termos: um algoritmo computável para abdução. O que só é possível porque não se trata da abdução vislumbrada por Peirce.

Dois fatores podem ter induzido essa abordagem algo equivocada: o progressivo desvelamento da filosofia peirciana, que segue incompleto, e a própria evolução do conceito de abdução em Peirce. Santaella (2004, p. 85) destaca o caráter evolutivo dos conceitos associados aos tipos de raciocínio na obra de Peirce. Em 1892, os tipos eram classificados como dedução, indução e hipótese; a hipótese e a indução eram ambas classificadas como "inferências prováveis" (CP

[4] It is important here to emphasise that the restriction of the hypothesis of abduction to abducible predicates is not incidental or computational, but has a deeper representational reason. It reflects the relative comprehensiveness of knowledge of the problem domain contained in T. The abducible predicates and the allowed abductive formulae take the role of 'answer-holders' for the problem goals that we want to set to our theory. In this respect they take the place of the logical variable as the answer-holder when deductive reasoning is used for problem solving. As a result this means that the form of the abductive hypothesis depends heavily on the particular theory T at hand, and the way that we have chosen to represent in this our problem domain.
Typically, the allowed abducible formulae are further restricted to simple logical forms such as ground or existentially quantified conjunctions of abducible literals. Although these further restrictions may be partly motivated by computational considerations, it is again important to point out that they are only made possible by the relative comprehensiveness of the particular representation of our problem domain in the theory T. Thus, the case of simple abduction — where the abducible hypothesis are ground facts — occurs exactly because the representation of the problem domain in T is sufficiently complete to allow this.

6.144–147, 1892). Somente dez anos depois (CP 2.96, 1902) os argumentos aparecem na forma de dedução, indução e abdução, com as características descritas no presente trabalho (Seção 3.3.1). Esse caráter evolutivo do conceito em Peirce, associado ao próprio caráter evolutivo da compreensão de sua obra, podem ter levado à escolha de um nome equivocado para uma forma de inferência computacional que procura a melhor explicação para uma observação e tem sido estudada e aperfeiçoada ao longo das décadas.

Outros métodos de abdução em computadores

Cumpre ressalvar que a abdução computacional não é o único método que se apropria do termo "abdução", embora aparentemente a maioria dos trabalhos em abdução por computador se baseie nela. Schurz (2008), por exemplo, propõe diferentes modelos conceituais de abdução. Apesar de citar Peirce em seu artigo, Schurz analisa o conceito de abdução a partir de uma perspectiva filosófica que não é peirciana. Sua classificação se baseia em três dimensões: de acordo com o tipo de hipótese a que se chega, com o tipo de evidência que a abdução pretende explicar e de acordo com as crenças, ou mecanismos cognitivos, que levam à abdução (SCHURZ, 2008, p. 205). Identifica dois grandes grupos de abdução: a abdução *seletiva*, que aproximadamente corresponde ao que aqui chamamos de abdução computacional, e abdução *criativa*, que de fato introduz novos conceitos, e da qual exibe diversos tipos em seu artigo. Um desses tipos tem sido aproveitado para propor modelos de aplicação nos quais se calcula a probabilidade de haver uma nova teoria unificadora de um conjunto pré existente de teorias (cf. FELDBACHER-ESCAMILLA; GEBHARTER, 2018).

Embora estejam distantes da conceituação peirciana, tanto a abdução computacional quanto as que podem se desdobrar a partir do trabalho de Schurz podem ser úteis no modelo dialógico de programação que ora propomos.

5.4.2 Abdução peirciana em computadores

A ideia de propósito incorpora a essência da terceiridade: o propósito é aquilo que, sem afetar diretamente a ação, determina seu curso (cf. CP 1.532, 1903). Nesse sentido deve haver um equilíbrio entre o propósito e a autonomia da máquina. É isso que, juntamente com a criação de novas ideias através do raciocínio abdutivo, tornará possível ação autônoma, como definida na Equação 5.15, no computador eletrônico. Vejamos um exemplo.

Imaginemos um programa de computador, que chamaremos $\mathcal{A}_{lea}$, e que, ao invés de seguir um algoritmo totalmente fixo, escolhe, em algum momento, aleatoriamente, uma ação, ou uma sequência de ações, dentro de um conjunto possível; por exemplo, um conjunto finito, pré-determinado de instruções que podem ser executadas. O programa apresentado por Gazoni (2016) faz exatamente isso: dadas as entradas e saídas esperadas, seleciona aleatoriamente uma sequência de instruções (um sequência de instruções é, por definição, um outro programa, o

programa-produto), aplica sobre elas os dados de entrada e verifica a saída; caso seja igual à saída esperada, exibe a sequência de ações sorteada. Caso contrário, repete o procedimento. O artigo encaminha a questão de como $\mathcal{A}_{lea}$, que é um programa, "sabe" qual a saída de outro programa, o programa-produto, o que aparentemente viola o teorema provado por Turing. Resumidamente, $\mathcal{A}_{lea}$ faz o que um programador humano faria: executa o programa-produto (num outro processo do sistema operacional) e, dentro de uma janela de tempo pré-definida, verifica os resultados. Se houver erro na execução, ou se a janela de tempo é ultrapassada, ou se o resultado não é o esperado, o programa-produto é descartado. O procedimento é repetido até que uma solução satisfatória seja encontrada, ou o número de tentativas ultrapasse um valor fixo. A atuação de $\mathcal{A}_{lea}$ é tal que, alimentado com a entrada e saída desejada, retorna um programa candidato a realizar a operação desejada. Por exemplo, se alimentado com "entrada: 5 e 3; saída: 8", poderia retornar um programa que soma dois números; poderia, também, retornar um programa que imprime "8", sem realizar soma nenhuma, um resultado incorreto que, não obstante, se encaixa no critério de parada da investigação.

O propósito de $\mathcal{A}_{lea}$ é encontrar programas que se encaixem numa especificação das entradas e saídas desejadas. É um caso em que a ação a_p procurada é conhecida — no exemplo a ação é gerar um programa que soma dois números —, mas não é especificada por si mesma, e sim por seus efeitos. O resultado constitui uma abdução peirciana: é totalmente imprevisível, é considerada correta não por si, mas por provocar um efeito desejado — e portanto "parecer" correta —, e por isso é uma hipótese a ser verificada. É importante ressaltar, entretanto, que o fato de ser uma hipótese que parece correta não pode ser enxergado pelo programa $\mathcal{A}_{lea}$. O usuário foi obrigado a tomar uma especificação indeterminada ("um programa que soma dois números") e a partir dela encontrar um critério determinado de parada que, se preenchido, garantiria um resultado que poderia estar correto.

O programa $\mathcal{A}_{lea}$ é claramente um caso em que uma sequência de escolhas e ações do usuário programador, que resultaria em um programa que faz somas, foi substituída por uma única ação na qual o programa, de certa forma, faz escolhas e apresenta o resultado delas ao usuário programador. A autonomia de $\mathcal{A}_{lea}$ é limitada pelo fato de não conseguir alterar seus próprios hábitos. Mas, uma vez que seus produtos também são programas, poderiam potencialmente alterar os hábitos do programa gerador, se os programas-produto resultantes pudessem, em outro contexto, ser incorporados à ação dele.

A real utilidade do programa $\mathcal{A}_{lea}$ é verificar se esse método de inferência abdutiva é possível, e a resposta é positiva. O programa, adicionalmente, ilustra uma maneira pela qual o propósito delineia mas não determina a ação. Segue um algoritmo bem definido cujo resultado não é mecanicamente definido porque executa um passo não computável, a seleção aleatória. De certa forma, o que faremos a seguir é tentar determinar, utilizando a moldura conceitual peirciana, modos de aproveitar esse raciocínio abdutivo computacional delineado (mas não

determinado) por um propósito de uma maneira que permita utilizar o computador atuando como agente autônomo.

5.4.3 Círculo funcional e Umwelt no computador

Investigando a ação semiótica da máquina $\mathcal{A}_{lea}$, exemplificada na Seção 5.4.2, vemos que é bastante simples. Na prática, realiza inferências abdutivas assim: são sorteadas soluções que são verificadas quanto à aderência a um resultado desejado; se a solução não é aderente, é descartada, senão, é apresentada.

Essa mesma ação semiótica pode ser implementada em um agente um pouco mais útil. O novo programa, que chamaremos $\mathcal{M}_{aux}$, apresenta essencialmente a mesma ação semiótica de $\mathcal{A}_{lea}$, num círculo funcional diferente. O programa $\mathcal{M}_{aux}$ é um auxiliar de trabalho. Funciona assim: uma pessoa está utilizando um editor de texto específico, que pode ser qualquer um, desde que exporte arquivos no formato PDF. Esse usuário quer performar uma ação específica, por exemplo, colocar uma palavra do texto em negrito. Mas a pessoa não sabe como fazê-lo nesse editor de texto. Ao invés de procurar por si uma solução, vai utilizar $\mathcal{M}_{aux}$ para isso. Passa como parâmetro para $\mathcal{M}_{aux}$ três arquivos, Arq_{ini}, Arq_{fim} e Arq_{src}. Dois deles, Arq_{ini} e Arq_{fim}, são em formato PDF. O arquivo Arq_{ini} contém o texto como o usuário o tem, Arq_{fim} contém o texto como o usuário deseja que fique. Por exemplo, Arq_{ini} contém uma linha com o texto "Essa palavra em negrito!!!", e Arq_{fim} contém uma linha com o texto "Essa palavra em **negrito**!!!". O arquivo Arq_{fim}, naturalmente, teria de ser gerado por um outro programa no qual o usuário soubesse como colocar uma palavra em negrito. O arquivo Arq_{src} está no formato do editor que está utilizando, e contém a situação inicial do texto. Contém, portanto, uma linha de texto na qual está escrito "Essa palavra em negrito!!!", num formato aceito pelo editor. O arquivo PDF Arq_{ini} foi derivado dele. Após algum tempo, $\mathcal{M}_{aux}$ exibe ao usuário uma sequência de comandos do editor de texto que realiza a transformação procurada, ou seja, uma sequência de comandos do editor que converte o texto "Essa palavra em negrito!!!" no texto "Essa palavra em **negrito**!!!". Como $\mathcal{M}_{aux}$ funciona?

Do mesmo modo que $\mathcal{A}_{lea}$, também sorteia sequências de ações, mas essas ações não são instruções de programas: são comandos disponíveis para o usuário no editor de texto cujo uso auxilia. Deve possuir adicionalmente a capacidade de comparar arquivos PDF. Com isso, $\mathcal{M}_{aux}$ sorteia uma sequência de comandos do editor, e as aplica a uma cópia do arquivo Arq_{src}. Exporta o resultado para o formato PDF e o compara com o arquivo Arq_{fim}. Se forem iguais, mostra a sequência de comandos para o usuário.

As repostas encontradas por $\mathcal{M}_{aux}$, assim como os programas-produto gerados por $\mathcal{A}_{lea}$, não são necessariamente as melhores soluções. Se a melhor solução for a ilustrada abaixo:

selecionar a palavra → clicar 'formatar' → 'caractere' → 'negrito'

A solução apresentada por $\mathcal{M}_{aux}$ pode ser, por exemplo:

clicar 'arquivo' → selecionar a palavra → clicar 'formatar' → 'caractere' → 'negrito'

Não é a melhor solução, mas é uma solução possível.

A diferença entre $\mathcal{A}_{lea}$ e $\mathcal{M}_{aux}$ é, basicamente, o círculo funcional e as capacidades inatas associadas ao círculo funcional. O programa $\mathcal{A}_{lea}$ "percebe" as entradas e saídas esperadas do programa; é capaz de sortear aleatoriamente instruções, e de mostrar as sequências de instruções que geram a saída desejada. Já $\mathcal{M}_{aux}$, por sua vez, recebe como entradas três arquivos; é capaz de comparar arquivos PDF, e gerá-los a partir do editor de texto que auxilia. Além disso, não sorteia sequências de instruções numa linguagem de programação, mas sequências de comandos do editor de texto, e mostra as que se encaixam em determinado critério. As diferenças entre ambos não está no raciocínio. Está nos ambientes de percepção e operacional do seu círculo funcional.

Uma importante característica das capacidades presentes no círculo funcional, como conjeturamos na Seção 3.6, é que os diagramas que o agente cria durante a investigação guardam alguma relação com o círculo funcional do agente, uma suposição que levaremos em conta mais adiante.

5.4.4 Proposições no computador

Evidentemente, uma proposição não é uma frase. Frases expressam proposições, mas não são elas. Proposições são dicissignos que combinam um índice e um ícone no seu um efeito sobre a mente. Por exemplo, as proposições provenientes do percepto ("percebidas") se impõem à mente sem dar as razões para tal. Os possíveis efeitos disso foram apresentados na Seção 3.5.2. Na Seção 3.4, a partir da página 55, vimos que, se há uma predisposição, a mente passa a investigar essas proposições "percebidas" de acordo com algum propósito. Portanto, proposições, no contexto dos programas que buscamos especificar, consistem em elementos que são tratados do mesmo modo que os elementos das proposições são tratados num processo de inferência. É o tratamento dado aos elementos que os caracterizam como sendo pertencentes a uma proposição. Somente a partir dessa base a proposição pode ser posteriormente expressa. Isso adere ao que no Capítulo 2 é o objeto (no caso, a proposição sendo investigada) que determina o signo (a expressão da proposição).

A capacidade de gerar proposições a partir do percepto é instintiva e leva em conta o universo do discurso, como vimos na Seção 3.5.2. Assim, as primeiras proposições geradas pelo programa são produto dessa capacidade, que também deverá fazer parte do seu círculo funcional. Essa capacidade não só determina as primeiras proposições disponíveis para o programa, mas também delimita uma fronteira no agente incorporado no programa; essa fronteira é a que corresponde, para nós, à separação entre o mundo exterior e o mundo interior.

Em particular, as proposições se prestam à criação de abstrações. O raciocínio diagramático consiste em criar e observar diagramas, e novos diagramas podem ser criados a partir de abstrações. Resumindo o que vimos na Seção 3.3.3, toma-se o predicado de uma proposição como abstração. Essa abstração é sujeito de uma proposição cujo predicado queremos descobrir; e as abstrações podem ser ordenadas de acordo com a primariedade das substâncias a que se referem.

6 Estudos de implantação

No presente capítulo esboçaremos a implantação dos conceitos expostos até aqui; o principal objetivo é ilustrar como esses conceitos podem orientar a construção de uma solução na qual o computador eletrônico possua maior autonomia. Os exemplos que mostraremos devem ser vistos com duas ressalvas: primeira, não são exemplos prontos para implantação direta em programas. Muitas vezes, a implantação exigirá um detalhamento não trivial das funções e constituirão objeto de pesquisa futura. Outra ressalva é que o mapeamento apresentado não é necessariamente o único possível. Trata-se de uma visão que, naturalmente, incorpora as limitações do autor. O projeto de um sistema dialógico completo, que reúna todas as possibilidades sugeridas pelo presente trabalho, será tema de pesquisa futura.

O capítulo apresenta um problema que servirá como exemplo de aplicação na Seção 6.1. Uma proposta para implementação de hábitos e aprendizado está na Seção 6.2. Na Seção 6.3 são analisadas algumas formas de implementar os diferentes tipos de raciocínio, que possibilitam uma solução dialógica para o problema, apresentada na Seção 6.4. A Seção 6.5 propõe soluções adicionais para explicitar proposições, abstrações e propósitos de investigação. Possíveis pontos de melhoria são mostrados na Seção 6.6, e a Seção 6.7 elenca algumas possibilidades de implementação em casos reais.

6.1 Problema de aplicação

Vamos usar como exemplo um programa que estende o programa $\mathcal{A}_{lea}$, apresentado na Seção 5.4.2. Esse novo programa, que chamaremos $\mathcal{O}_1$, atua sobre os resultados de $\mathcal{A}_{lea}$. O programa $\mathcal{A}_{lea}$ não gera resultados otimizados: os programas-produto podem conter, por exemplo, instruções redundantes. Na Tabela 3 há alguns exemplos de resultados possíveis de $\mathcal{A}_{lea}$. A linguagem de programação dos programas-produto de $\mathcal{A}_{lea}$ é detalhada no Apêndice B. O objetivo do programa $\mathcal{O}_1$ é eliminar os passos redundantes dos programas-produto de $\mathcal{A}_{lea}$. Como vemos na Tabela 3, as linhas 1 e 2 possuem os mesmos dados de entrada, mas apresentam saídas diferentes. A razão é justamente a presença dos comandos redundantes. A solução sem redundância é "`INP INP + PRT`".

Trata-se de um problema conhecido cuja solução é conhecida. Poderíamos construir um programa que elimina os passos redundantes, ou seja, conhecemos a sequência de escolhas c_i e ações a_i que chegarão ao resultado desejado. E, como conhecemos a solução, poderíamos utilizar um programa semelhante ao programa $\mathcal{A}_{lea}$, tendo a solução como critério de parada. Esse novo programa seria o próprio $\mathcal{A}_{lea}$ estendido com a capacidade de processar dados não

Tabela 3 – Resultados possíveis do programa $\mathcal{A}_{lea}$

	Dados de entrada		
	E_o = Entrada do programa-produto	S_o = Saída do programa-produto	Programa-produto encontrado
1	$\{3,5\}$	8	`LET a + INP INP + PRT`
2	$\{3,5\}$	8	`INP INP + PRT +`
3	$\{3,5\}$	15	`INP INP × PRT`
4	$\{15,5\}$	3	`INP INP ÷ PRT`
5	$\{13\}$	26	`INP 2 × PRT`
6	$\{25,2\}$	13	`13 PRT`
7	$\{96\}$	13	`13 PRT`

numéricos como entrada e saída. O novo programa poderia usar como dados de entrada:

$$\{E_o = \{\texttt{LET a + INP INP + PRT}\}, S_o = \{\texttt{INP INP + PRT}\}\}$$

Além de ser uma alternativa que exige que conheçamos a solução, apresenta problemas adicionais que analisaremos adiante, em outro contexto (Seção 6.3.1).

Antes de esboçar a implementação da investigação propriamente dita, vamos estabelecer sobre que bases ela ocorre. Comecemos por propor, para nosso caso particular, quais estruturas possibilitam hábitos e aprendizado.

6.2 Hábitos e aprendizado

Há várias maneiras de implementar a capacidade de mudar os hábitos do programa; para fins de ilustração propomos uma lista, interna ao programa, que contém todos os procedimentos à disposição dele. Trata-se da lista de hábitos do programa, que doravante chamaremos de lista $\mathbb{H}$, para "hábitos". Idealmente, cada item de $\mathbb{H}$ deveria conter diagramas explicativos para:

- As circunstâncias em que o procedimento é executado; como vimos na Seção 3.2.2, um hábito é um comportamento que ocorre em determinadas as circunstâncias.
- As entradas, que são os dados utilizados na execução do procedimento.
- As saídas, ou os dados produzidos pelo procedimento.

Além dessas informações seria interessante incluir ao menos mais duas: um texto contendo uma palavra ou frase que serve para identificar o procedimento para um usuário externo ao programa; e um (ou mais) comandos de execução, que acionam internamente a execução do procedimento.

Ilustraremos essa ideia mostrando alguns itens possíveis nessa lista de procedimentos. Estão descritos nas Tabelas 4 a 8. Essas descrições não são suficientemente detalhadas para implementação em programas de computador; a ideia é apresentar em linhas gerais uma lista de procedimentos e suas descrições e explorar algumas consequências da lista.

Nas Tabelas 4 e 5, vemos descrições dos procedimentos de $\mathcal{A}_{lea}$ e $\mathcal{M}_{aux}$, respectivamente. Na Tabela 6 está um procedimento que executa outros procedimentos de acordo com algum

Tabela 4 – Procedimento "principal" de $\mathcal{A}_{lea}$

Identificação	"busca de programa-produto"
Circunstâncias de execução	a pedido do usuário
Entradas	entradas do programa-produto (E_o), saída desejada do programa-produto (S_o)
Saídas	o programa-produto
Comando (exemplo)	`gerar-programa-produto(E, S)`

Tabela 5 – Procedimento "principal" de $\mathcal{M}_{aux}$

Identificação	"busca de sequência de comandos"
Circunstâncias de execução	a pedido do usuário
Entradas	os arquivos Arq_{ini}, Arq_{fim} e Arq_{src}
Saídas	a sequência de comandos
Comando (exemplo)	`gerar-sequencia`(Arq_{ini}, Arq_{fim}, Arq_{src})

critério. O procedimento da Tabela 7 é aquele que registra, numa outra lista, as entradas, saídas e

Tabela 6 – Procedimento que executa procedimentos

Identificação	"execução de procedimentos"
Circunstâncias de execução	de acordo com o propósito e hábitos éticos do programa
Entradas	identificação procedimento a executar e suas entradas (E)
Saídas	circunstâncias de execução (data, hora, resultado) e saídas do procedimento executado
Comando (exemplo)	`executar-procedimento`("busca de sequência de comandos", E). Esse exemplo executa o procedimento da Tabela 5.

circunstâncias da execução de procedimentos. Um outro procedimento, que poderia substituir o procedimento arbitrário de execução da Tabela 6 é mostrado na Tabela 8; esse procedimento constitui um hábito ético do programa, pois determina sua forma de atuação. O algoritmo de execução, que chamaremos $\mathcal{E}_{th}$, está mostrado em seguida; trata-se de uma implementação de uma máquina de Turing de escolha: há um passo em que aguarda uma atuação externa.

Tabela 7 – Procedimento que registra execução de procedimentos

Identificação	"registro de execução de procedimentos"
Circunstâncias de execução	de acordo com o propósito e hábitos éticos do programa
Entradas	identificação do procedimento executado (I_d), dados da particular instância de execução (entradas e saídas efetivas, armazenadas em $Exec_{data}$), circunstâncias de execução (data, hora etc, armazenados $Exec_{circ}$) e nome lista na qual registrar a execução do procedimento
Saídas	nenhuma; o procedimento apenas registra numa lista os dados recebidos
Comando (exemplo)	`registrar-execucao`(I_d, $Exec_{data}$, $Exec_{circ}$, "lista-log"). Esse exemplo registra as circunstâncias de execução na lista chamada "lista-log".

Tabela 8 – Um hábito ético: forma de atuação de um programa

Identificação	"forma de atuação"
Circunstâncias de execução	sempre
Entradas	nenhuma
Saídas	nenhuma
Comando	Segue o algoritmo de execução abaixo

1. Recupere da lista de procedimentos o procedimento
   "registro de execução de procedimentos" e armazene-o em r_{exec}.
2. Aguarde que as circunstâncias mudem (esse é o passo em que aguarda que algo mude).
3. As novas circunstâncias são armazenadas no conjunto $C = \{c_1, c_2, \ldots, c_n\}$.
4. Para cada procedimento p_i diferente de "forma de atuação" e de r_{exec} na lista $\mathbb{H}$
   de procedimentos:
5. Se C é semelhante às circunstâncias de execução do procedimento p_i:
6. Transfira os dados de C que são entrada para o procedimento p_i para E_i.
7. Execute o comando de p_i passando como entrada os dados de E_i.
   Registre as saídas em S_i.
8. Execute o comando de r_{exec} com os dados de p_i, E_i, S_i,
   as circunstâncias de execução tendo como lista o registro geral do programa.
9. (Final da instrução 5).
10. (Final da instrução 4).
11. Retorne para o passo 2.

As especificações dos procedimentos acima não são suficientes para criar um programa a partir delas. Por exemplo, o passo 5 do algoritmo da "forma de atuação" compara dois conjuntos por semelhança; tal comparação depende de uma definição de "semelhança" nesse contexto. Poderia ser implementada de várias maneiras — por exemplo, através de redes neurais —, com diferentes consequências para o resultado final. Indo além, pode-se questionar se a avaliação

da semelhança de maneira parecida com a humana é factível em uma máquina. Dreyfus (1992) afirma que qualquer avaliação de uma situação, quando feita por um ser humano, conta com uma priorização hierárquica dos elementos avaliados que não pode ser reproduzida pelo computador. Mesmo assim, basta que a implementação escolhida para essa comparação por semelhança permita a interlocução com o usuário programador, ou seja: deve ser, até certo ponto, conhecida a ponto de permitir que as escolhas do usuário levem a ações pertinentes. No nosso caso, seria suficiente garantir que o conjunto de entradas E_i do procedimento seja um subconjunto do conjunto C das circunstâncias: $E_i \subseteq C$. Isso asseguraria que o passo 6 poderia ser executado.

Uma característica digna de nota nesses procedimentos é que dão ao programa a capacidade de executar outros procedimentos e registrar a execução dos procedimentos. Isso leva a outra observação: não é o fato de incluirmos na lista $\mathbb{H}$ um registro qualquer que o comando de execução incluído junto com o item poderá, de fato, ser executado. A lista construída até agora deve ser entendida como uma enumeração de procedimentos já existentes no programa. E como são capacidades deste, pode ser entendida, se quisermos, como parte do círculo funcional deste. É possível, entretanto, criar uma lista tal que seus procedimentos são suficientes para executar qualquer ação que possa ser executada por uma máquina de Turing automática.

Se houvesse um procedimento capaz de incluir um item na lista $\mathbb{H}$, esse novo item seria executado (desde que possuísse um comando válido de execução) pelo programa, de acordo com o algoritmo $\mathcal{E}_{th}$. Poderíamos, por exemplo, incluir uma versão diferente do procedimento de registro de execução de procedimentos, mostrado na Tabela 7. Esse novo item está exemplificado na Tabela 9. Observemos que o comando nessa tabela é o mesmo da Tabela 7. Ou seja, a inclusão da Tabela 9 em $\mathbb{H}$ resulta, de fato, na disponibilização de um hábito adicional.

Tabela 9 – Registra execução de $\mathcal{A}_{lea}$

Identificação	"registro de execução de $\mathcal{A}_{lea}$"
Circunstâncias de execução	toda vez que se executa $\mathcal{A}_{lea}$
Entradas	identificação do procedimento executado (I_d ="busca de programa-produto"), dados da particular instância de execução (entradas e saídas efetivas, retiradas da lista de circunstâncias C, armazenadas em $Exec_{data_{\mathcal{A}_{lea}}}$), circunstâncias de execução (data, hora etc, armazenados $Exec_{circ_{\mathcal{A}_{lea}}}$) e nome lista na qual registrar a execução do procedimento, que será "log-Alea"
Saídas	nenhuma; o procedimento apenas registra numa lista os dados recebidos
Comando (exemplo)	`registrar-execucao`("busca de sequência de comandos", $Exec_{data_{\mathcal{A}_{lea}}}$, $Exec_{circ_{\mathcal{A}_{lea}}}$, "log-Alea")

Um procedimento no círculo funcional do programa possibilitando a inclusão de um item na lista $\mathbb{H}$ tornaria possível a mudança de hábito do programa; se o próprio programa o

executasse, isso constituiria uma mudança de hábito feita pelo próprio programa. Se o programa utilizasse esse procedimento para incluir o procedimento da Tabela 9 na lista $\mathbb{H}$, passaria a imprimir separadamente os resultados do procedimento $\mathcal{A}_{lea}$. Porém, podemos ir além. O algoritmo $\mathcal{E}_{th}$ apresenta capacidades que não foram incluídas como itens da lista $\mathbb{H}$ de hábitos do programa, mas que poderiam fazer parte dela. São as capacidades de:

- recuperar um item específico de uma lista (passo 1)
- armazenar um item em uma memória nomeada (passo 1)
- aguardar a mudança de circunstâncias e, quando elas mudam, atualizar uma lista C de circunstâncias (passos 2 e 3)
- iterar ao longo de uma lista e, para cada item dela, executar condicionalmente um algoritmo (passo 4)
- comparar coisas (passo 5)
- executar um procedimento de acordo com uma condição resultante de uma comparação (passo 5)
- transferir dados entre listas, o que implica incluir itens em uma lista (passo 6)
- transferir a execução de um algoritmo para um ponto arbitrário no algoritmo (passo 11)

Vamos acrescentar a essas capacidades, presentes no algoritmo $\mathcal{E}_{th}$, a capacidade de excluir itens de uma lista. Chamaremos essa lista de capacidades estendida de $\mathbb{H}_T$. Ela é importante porque, se nos procedimentos de $\mathbb{H}_T$ a ordem das listas manipuladas é preservada quando se exclui ou inclui um item da lista, então esse conjunto de capacidades $\mathbb{H}_T$, podendo ser estruturadas em uma sequência lógica (por exemplo, no comando de execução do procedimento), constituem um dispositivo com capacidade equivalente à de uma máquina de Turing automática, e, em particular, uma máquina de Turing universal.[1]

Dessa forma, a capacidade de mudar o próprio hábito que procuramos para o programa pode ser conseguida se acrescentarmos as capacidades da lista $\mathbb{H}_T$ ao programa e incluirmos a própria lista $\mathbb{H}_T$ na lista $\mathbb{H}$ dos hábitos dele. Qualquer mudança de hábitos que se queira pode ser obtida através da inclusão e exclusão de itens adequadamente escolhidos da lista $\mathbb{H}$. Em particular, um algoritmo que elimina passos supérfluos de uma sequencia sorteada, que é o algoritmo que procuramos para o programa $\mathcal{O}_1$, pode ser incluído na lista $\mathbb{H}$ desde que se saiba qual algoritmo incluir.

A lista de hábitos inicial que propomos para o programa $\mathcal{O}_1$ consiste nos procedimentos listados nas Tabelas 4 a 9 acrescentados de uma lista de hábitos que chamamos $\mathbb{H}_I$. A lista $\mathbb{H}_I$ está detalhada no Apêndice C. Lá aparecem: a sintaxe de cada procedimento, uma descrição do

[1] Não é difícil ter um visão intuitiva disso: as listas fazem o papel da fita, conjuntos adequados de condições e atuações fazem o papel das configurações-*m*. Subsídios para uma demonstração formal, que não daremos, podem ser encontrados no texto de Rogers (1967).

que faz, uma relação, que será melhor entendida quando falarmos de proposições (na Seção 6.5), o retorno (resultado) esperado do procedimento, um exemplo e a explicação do exemplo. Não aparece as circunstâncias de execução de cada hábito.

Assim, o programa $\mathcal{O}_1$, até o momento, tem os recursos necessários para alterar sua própria atuação, como uma máquina de Turing universal. As únicas diferenças essenciais dele para essas máquinas é que em alguns procedimentos de $\mathcal{O}_1$ há o sorteio de números aleatórios (o que o permite executar os procedimentos principais de $\mathcal{M}_{aux}$ e $\mathcal{A}_{lea}$) e ele possui um estado em que aguarda que as circunstâncias mudem, sendo capaz de registrá-las.

Uma vez que $\mathcal{O}_1$ tem a capacidade teórica de alterar seus próprios hábitos surge a tentação de classificar os hábitos de sua lista $\mathbb{H}$ em dois grupos: hábitos mentais, quando o hábito existe por ter sido consequência de uma mudança de hábito do programa, e hábitos pertencentes ao círculo funcional, quando os hábitos são, por assim dizer, inatos, ou seja, já estavam lá antes do programa iniciar. Vamos evitar essa classificação. Na moldura conceitual peirciana, vimos (Seção 3.6) que pode haver hábitos adquiridos que se tornam inconscientes e parecem instintivos. Portanto, não há razão para estabelecer a essa distinção. Doravante utilizaremos "lista de hábitos" e "circulo funcional" conforme o contexto, mas os termos podem ser intercambiados.

Vejamos, a seguir, como se pode se dar o processo investigativo no contexto de nosso estudo.

6.3 Investigação utilizando raciocínio autônomo

O processo de inferência inicia na observação icônica de um diagrama, como vimos na Seção 3.3.2. Consiste em "olhar" para o ícone (imagem, diagrama, metáfora, o que for) e tentar encontrar nele relações que antes não havíamos percebido. Essa possibilidade é uma das características da natureza icônica daquilo que observamos dessa forma. Ao experimentar o sabor do açúcar e do mel reconhecemos que ambos guardam entre si uma certa semelhança, por si indefinível, mas que aplicamos a uma abstração que chamamos doçura. Foi só por causa da capacidade de primeiro perceber o sabor de ambos, depois de reconhecer sua semelhança, que foi possível criar a abstração — o objeto precedendo o signo. Para que um elemento dotado de iconicidade funcione como tal para um agente, é preciso que este tenha o poder de observar esse elemento e julgar semelhanças entre este e outros elementos.

Como conjeturamos na Seção 3.6, os diagramas criados pelo agente na investigação guardam, inicialmente, alguma relação com o seu círculo funcional. No contexto do programa, isso significa que a habilidade de observar um ícone como diagrama deve corresponder a um ou mais procedimentos na lista $\mathbb{H}$. Têm como entrada o elemento a ser observado, e retornam características desse elemento, por sua vez também passíveis de observação e experimentação.

Vamos usar como exemplo exatamente o que nos propusemos a investigar: um resultado do programa $\mathcal{A}_{lea}$. Gostaríamos de eliminar os passos redundantes desses programas-produto. Tomemos como exemplo o resultado "`LET a + INP INP + PRT`" e vejamos alguns procedimentos que podem observá-lo e manipulá-lo.

Separação em caracteres retorna a lista {"`L`", "`E`", "`T`", "` `", "`a`", "` `", "`+`", "` `", "`I`", "`N`", "`P`", "` `", "`I`", "`N`", "`P`", "` `", "`+`", "` `", "`P`", "`R`", "`T`"}. O procedimento oposto, que recebe uma lista de caracteres e retorna o texto, também é possível.

Separação em palavras retorna {"`LET`", "`a`", "`+`", "`INP`", "`INP`", "`+`", "`PRT`"}. Novamente, dada uma lista de palavras, pode-se reconstruir o texto.

Executar como programa executa o texto como um programa escrito na linguagem de $\mathcal{A}_{lea}$ e retorna o resultado.

Esses procedimentos podem ser complementados por outros procedimentos, estes atuando sobre listas e palavras. Por exemplo:

Comprimento da lista recebe uma lista com entrada e retorna o número de elementos nela. Se associado ao resultado do procedimento "separação em caracteres", conta o número de caracteres no predicado; se associado ao resultado do procedimento "separação em palavras", conta o número de palavras. No caso dos exemplos acima, seus resultados seriam respectivamente 21 e 7.

Inclusão em lista inclui um elemento em determinada posição de uma lista dada. O oposto, exclusão da lista, também pode existir.

Permuta duas posições da lista é uma combinação de exclusões e inclusões que resulta na troca de posições de elementos em uma lista.

Gerar combinações de elementos de listas retorna uma lista de listas, cada uma contendo uma combinação possível de elementos de uma lista inicial, incluindo a lista vazia e a própria lista inicial. Isso pode ser feito através da inclusão, exclusão e permuta dos elementos da lista inicial.

Esses dois últimos procedimentos, de permuta de posições e de geração de combinações de elementos, podem ser criados a partir dois procedimentos anteriores, de inclusão e exclusão em lista, mais os procedimentos de $\mathbb{H}_I$. Então, teoricamente, também poderiam ser inferidos pelo programa e incluídos como hábitos.

Esses sete novos procedimentos, propomos, farão parte das capacidades do círculo funcional do programa, estarão incluídos na lista $\mathbb{H}$ de hábitos do programa (e terão correspondência na lista $\mathbb{R}$ das relações geradas por procedimentos, que veremos na Seção 6.5.1). Podemos agora

começar a mostrar um método de investigação baseado na observação do ícone pelo programa $\mathcal{O}_1$.

Queremos chegar a um programa capaz de eliminar os passos redundantes de qualquer resultado de $\mathcal{A}_{lea}$. Uma algoritmo que faz isso pode ser a que segue, o algoritmo $\mathcal{T}_{r1}$ (o predicado inicial, no caso, "`LET a + INP INP + PRT`", está armazenado em `mPrograma`):

```
1. mMenorComprimento ← pComprimentoDeLista(pSeparaPalavras(mPrograma))
2. mMenorPrograma ← mPrograma
3. mResultado ← pExecuta(mPrograma)
4. lTodasCombinacoes ← pCombinacoesLista(pSeparaPalavras(mPrograma))
5. pParaCadaItemDe(lTodasCombinacoes pEm lCombinacaoTestada) {
6.     mComprimentoTestado ← pComprimentoDeLista(lCombinacaoTestada)
7.     pSe(mMenorComprimento > mComprimentoTestado) {
8.         mProgramaTestado ← pConcatenaPalavras(lCombinacaoTestada)
9.         pSe(mResultado = pExecuta(mProgramaTestado)) {
10.            mMenorPrograma ← mProgramaTestado
11.            mMenorComprimento ← mComprimentoTestado
           }
       }
   }
12. pImprima(mMenorPrograma)
```

Este procedimento utiliza somente comandos presentes em $\mathbb{H}_I$, mais oito símbolos que representam locais de memória.

6.3.1 Investigação abdutiva

Como conceituamos no início da Seção 5.4 e na Seção 5.4.2, o raciocínio abdutivo em computadores é obtido através do sorteio de números aleatórios. Há várias maneiras de se implementar isso. Uma delas é um procedimento de sorteio de programas (procedimentos) que utiliza como substrato os procedimentos em $\mathbb{H}_I$ e obedeça às seguintes premissas, premissas que chamaremos $\mathrm{P_{Abd}}$:

1. Os procedimentos sorteados possuem de uma a seis instruções.
2. O sorteio é contextualizado: sorteia-se, para a posição de cada instrução, um dos 9 comandos que não retorna valor, e em seguida sorteiam-se os componentes do comando:
   a) Se o comando utiliza valores como parâmetros, para cada valor utilizado é sorteado um símbolo que pode retornar um valor.

i. Se o comando for `pSe`, o único comando que aceita um valor lógico como argumento, os valores são sorteados entre os 9 símbolos que retornam valores lógicos. Desses, 3 aceitam somente argumentos que são valores lógicos. Os valores desses são sorteados entre os 9, recursivamente, até que todos os valores estejam definidos.

ii. Se o comando for o comando de atribuição (←), então o lado esquerdo (que tem de ser um local de memória) é sorteado entre os 8 locais de memória possíveis. O mesmo vale para o componente `pEm` do comando `pParaCadaItemDe (...) {...}`.

iii. Se for qualquer outro comando, os valores sorteados serão sorteados entre os 13 + 8 = 21 símbolos remanescentes que retornam valores.[2]

b) Se o comando utiliza procedimentos, o procedimento é sorteado como um programa, ou seja, por este mesmo algoritmo.

c) Se o comando for o comando `pExecuta`, é sorteado um dos 8 locais de memória, ou, com a mesma probabilidade de cada um deles, um outro programa que é transformado em texto para ser usado como parâmetro. Portanto, a probabilidade de se sortear um dos locais de memória ou um programa é $\frac{1}{9}$.

d) Se o comando for o comando `pLabel` ou o comando `pGoto`, o rótulo também é sorteado entre 6 rótulos possíveis.

Esse procedimento possui importantes limitações, decorrentes não somente, mas também, do fato de ter sido construído especialmente para encontrar a solução procurada. O cálculo das probabilidades baseado nele, juntamente com considerações adicionais a respeito de suas limitações e das variáveis envolvidas aparecem no Apêndice D.

A probabilidade de um programa chegar a exatamente $\mathcal{T}_{r1}$ através de um procedimento de sorteio que siga as premissas P_{Abd} é de aproximadamente $1,1 \times 10^{-53}$. Isso corresponde a um tempo médio de processamento muitas vezes superior à idade conhecida do universo[3] no computador determinado no preâmbulo do Capítulo 4.

Mas o principal problema para encontrar exatamente o programa $\mathcal{T}_{r1}$ não é o tempo de processamento necessário pelo raciocínio puramente abdutivo: é o critério de parada. Estamos analisando um caso apenas (o resultado "`LET a + INP INP + PRT`"). Supondo que não conhecemos a solução do problema, o procedimento de sorteio deverá, sucessivamente, (1) sortear uma solução e (2) verificar se a solução é correta e eliminou passos redundantes de "`LET a + INP INP + PRT`"; se não, voltar ao passo (1). O procedimento $\mathcal{T}_{r1}$ certamente elimina passos redundantes do resultado em questão, mas na verdade faz mais que isso: ele elimina todos os

[2] São 31 comandos, dos quais 9 não retornam valor e 9 retornam valores lógicos, restando 31 − 9 − 9 = 13 comandos que retornam valores não lógicos, mais 8 locais de memória totalizando 21 símbolos.

[3] O cálculo não leva em conta a possibilidade, irrelevante no caso, de utilizar processamento paralelo.

passos redundantes de qualquer resultado. Isso não pode ser verificado pelo critério acima, um fenômeno que investigamos a seguir porque encerra uma interessante conjetura.

Podemos descrever formalmente que há um programa $\mathcal{T}$ elimina todos os passos redundantes de qualquer entrada e utilizando a seguinte fórmula:

$$\exists\mathcal{T}\forall e\nexists f\Big(R\big(\mathcal{T}(e)\big) = R(e) \wedge L\big(\mathcal{T}(e)\big) > L(f) \wedge R(f) = R(e) \wedge f = C(e)\Big) \tag{6.1}$$

Nessa equação, $L(x)$ é o número de instruções de x, $R(x)$ é o resultado do programa x e $y = C(x)$ indica que y é uma combinação de um conjunto qualquer de instruções de x. Já a descrição formal de que há um programa $\mathcal{T}$ que elimina passos redundantes de uma entrada e específica é dada por:

$$\exists\mathcal{T}\Big(R\big(\mathcal{T}(e)\big) = R(e) \wedge L\big(\mathcal{T}(e)\big) < L(e)\Big) \tag{6.2}$$

Observa-se que a diferença formal entre as Equações 6.1 e 6.2 é a mesma que colocaria dois conjuntos descritos por elas em diferentes hierarquias aritméticas (cf. Seção 5.2.2). É possível que razão da impossibilidade de se chegar ao programa $\mathcal{T}_{r1}$ tenha relação com o teorema de Post (cf. KLEENE, 1952): é preciso uma máquina-oráculo de grau superior para computar essa solução. Se o raciocínio abdutivo que propomos funciona, nesse contexto, como uma máquina-oráculo (uma afirmação a ser investigada), é preciso aplicações adicionais do "oráculo" para chegar à solução procurada. Conjeturamos, além disso, que tal aplicação poderia ser traduzida como a aplicação do "oráculo" ao próprio processo investigativo, de modo a encontrar um caminho de investigação que leve à solução desejada, como sugerido na Seção 5.3.1 para o caso em que o conhecimento do propósito, a ação a_p, é vago. Essa questão fica ainda mais evidente se levarmos em conta os resultados apresentados na Seção 6.4, na qual fica claro que a sobreposição de procedimentos investigativos pode levar a soluções mais gerais.

A partir da análise de um único caso é possível chegar, por sorteio, a diversas soluções que satisfaçam ao critério "eliminar passos redundantes no programa apresentado (gerado por $\mathcal{A}_{lea}$)". Há muitos outros programas possíveis, além de $\mathcal{T}_{r1}$, que retornam uma versão melhorada do predicado inicial. Por exemplo, o programa $\mathcal{T}_{r2}$ abaixo:

```
1. pImprima(pConcatenaPalavras(
          pExcluiDelista(pSeparaPalavras(mPrograma), 0)
   ))
```

A probabilidade de se chegar ao programa acima[4] é uma em 1.643.686.506, podendo ser obtido por sorteio na máquina prototípica (sem processamento paralelo) num processamento que dura

[4] Alterando-se adequadamente as premissas P_{Abd} de sorteio, ou seja: (1) admitindo a premissa de que números de 0 a 9 podem também ser sorteados como valores; (2) admitindo que os comandos `pExcluiDelista(..)` e, por extensão, `pIncluiEmLista(...)`, retornam as listas modificadas; e (3) se, além disso, alterássemos alguns parâmetros de sorteio, como por exemplo o número de instruções inicial. Com essas alterações, chegaríamos a performances muito melhores que essa.

em média cerca de três dias.[5] Uma das razões para essa alta probabilidade é o fato de o programa ter apenas uma linha; outra razão é que o programa não retorna uma versão ótima do programa-produto inicial. Retorna apenas uma versão melhorada. Além disso, o programa não é uma solução geral: vale somente para esse caso.

Portanto, tentar abdutivamente uma solução para o problema da eliminação de passos redundantes de programas-produto do programa $\mathcal{A}_{lea}$, tendo apenas uma ocorrência para observar e sem conhecer de antemão a solução, nos levará provavelmente ao procedimento $\mathcal{T}_{r2}$ e não a $\mathcal{T}_{r1}$. O procedimento $\mathcal{T}_{r2}$ satisfaz, como $\mathcal{T}_{r1}$, um critério claro de parada da investigação, mas tem muito mais chance de ser atingido antes de $\mathcal{T}_{r1}$. Mesmo assim, $\mathcal{T}_{r2}$ é um resultado que cumpre o papel de raciocínio abdutivo: traz uma ideia nova que pode ser aproveitada, de forma aderente à moldura filosófica que escolhemos. Vimos, na Seção 3.4, que uma forma rudimentar de aquisição de hábitos é a facilitação de um hábito exitoso graças à repetição. Aplicando esse conceito ao aprendizado em questão, teríamos que, num processo de investigação, os comandos do programa $\mathcal{T}_{r2}$ seriam considerados "promissores" como solução, numa busca abdutiva. Mas pode-se fazer uso dessa solução num outro tipo de investigação.

6.3.2 Investigação dedutiva

A principal característica da dedução é a relação de secundidade entre as premissas e as conclusões: aceitas as primeiras, não se pode negar as segundas. Essa determinação se dá por que existe conjunto de leis que estabelece as relações válidas entre as premissas e as conclusões, como visto na Seção 3.3. Assim como foi preciso acrescentar ao círculo funcional do programa hábitos que o habilitassem a investigar o ícone, e determinar um procedimento de sorteio — que também é um hábito — para gerar inferências abdutivas, será preciso dotar o círculo funcional do agente com hábitos que constituem leis de dedução. Pode-se escolher procedimentos de diferentes naturezas e com diferentes escopos. Uma característica dos sistemas baseados em dedução é que permitem a formação de procedimentos complexos a partir da combinação de procedimentos mais simples, uma característica que pode ser aproveitada na escolha de quais procedimentos dedutivos adotar.

Um procedimento dedutivo geral é conhecido informalmente como "força bruta": nele são testadas todas as alternativas possíveis e a melhor, de acordo com algum critério de parada, é selecionada. Isso pode ser feito por inúmeros métodos: permutações, combinações, arranjos, com ou sem repetição. Cada método pode corresponder a um procedimento do círculo funcional. Propomos um procedimento que siga as premissas dadas abaixo, chamemo-las $\mathrm{P}_{\mathrm{Ded}}$, que utiliza as informações a respeito dos hábitos disponíveis no programa presentes na lista $\mathbb{H}_I$ e os seguintes dados:

[5] Na prática, esse tempo pode ser trazido a valores da ordem de minutos com a utilização de paralelismo e equipamentos mais potentes.

mPrograma : um procedimento, composto de uma série de comandos presentes em $\mathbb{H}_I$.
mNumMin : um número inteiro, o menor de uma série a ser testada.
mNumMax : um número inteiro, o maior de uma série a ser testada.
mNumInc : um número inteiro, o incremento para ir de `mNumMin` a `mNumMax`.
lAcceptabilityTests : uma lista de procedimentos a serem executados para verificar a aceitabilidade ou não de um procedimento produzido.

Em linhas gerais, as premissas P_{Ded} de atuação do procedimento são os que seguem. Observar que o conhecimento de $\mathbb{H}_I$ é necessário para a execução dos passos 2 e 4:

1. Cria uma lista vazia `lSolucoesValidas`.
2. Cria uma lista `lTestes` contendo tantos itens quanto o número ocorrências de valores numéricos em `mPrograma`.
3. Inicializa todos os valores de `lTestes` com `mNumMin`.
4. Cria `mProgTeste` substituindo os valores numéricos de `mPrograma` pelos valores em `lTestes`.
5. Para cada item de `lAccetabilityTests`:
6. Se o programa `mProgTeste` não passa no teste:
7. Vá para o passo 9.
8. Inclua o programa `mProgramaATestar` nas lista `lSolucoesValidas`.
9. Se algum valor de `lTestes` for menor que `mNumMax`:
10. Localize o primeiro valor menor que `mNumMax` em `lTestes`.
11. Adicione `mNumInc` a esse valor.
12. Coloque o valor `mNumMin` em todos os valores anteriores.
13. Vá para o passo 4.
14. Imprima `lSolucoesValidas`

Se submetermos a ideia encontrada na etapa abdutiva a esse programa, os parâmetros seriam os seguintes:

mPrograma :

```
pImprima(pConcatenaPalavras(
        pExcluiDelista(pSeparaPalavras(mPrograma), 0)
))
```

mNumMin : 0
mNumMax : 9
mNumInc : 1
lAcceptabilityTests : `(pExecuta(mPrograma) = pExecuta(mProgramaTestado))`

Com isso obteremos a seguinte lista de respostas válidas:

```
1.  pImprima(pConcatenaPalavras(
        pExcluiDelista(pSeparaPalavras(mPrograma), 0)
    ))
2.  pImprima(pConcatenaPalavras(
        pExcluiDelista(pSeparaPalavras(mPrograma), 1)
    ))
3.  pImprima(pConcatenaPalavras(
        pExcluiDelista(pSeparaPalavras(mPrograma), 2)
    ))
4.  pImprima(pConcatenaPalavras(
        pExcluiDelista(pSeparaPalavras(mPrograma), 7)
    ))
5.  pImprima(pConcatenaPalavras(
        pExcluiDelista(pSeparaPalavras(mPrograma), 8)
    ))
6.  pImprima(pConcatenaPalavras(
        pExcluiDelista(pSeparaPalavras(mPrograma), 9)
    ))
```

Esse programa equivalem, respectivamente, às seguintes versões do programa inicial (que, lembremos, era "`LET a + INP INP + PRT`"):

```
1. a + INP INP + PRT
2. LET + INP INP + PRT
3. LET a INP INP + PRT
4. LET a + INP INP + PRT
5. LET a + INP INP + PRT
6. LET a + INP INP + PRT
```

Ao observarmos esse conjunto de informações, fica evidente que outro método dedutivo pode ser aplicado para obtenção de um resultado ainda melhor: é a extração dos elementos comuns de cada solução válida. Trata-se de um passo investigativo natural (em seres humanos) quando várias alternativas apresentam o mesmo resultado: perguntar o que há de comum a todas elas. O resultado de tal aplicação ao caso acima seria "`INP INP + PRT`", que corresponde ao resultado que seria dado pelo programa $\mathcal{T}_{r1}$. Mas essa é uma observação nossa, não do programa. O mesmo resultado — uma lista de soluções válidas à qual se pode aplicar um método dedutivo

para extrair uma solução melhorada — poderia ser obtida através de outra forma de raciocínio, que talvez seja mais apropriada para a obtenção de tal lista.

6.3.3 Investigação indutiva

A investigação indutiva se caracteriza pela formulação de uma hipótese e posterior confirmação ou refutação desta pela experimentação, o que pode ser feito de diversas maneiras.

Um exemplo é a investigação preliminar de um fenômeno qualquer: sem saber se há ou não lei subjacente ao fenômeno, o agente o experimenta, baseado na hipótese de que tal lei talvez exista. Desses resultados preliminares pode surgir abdutivamente uma lei, uma teoria, que parece explicar o fenômeno em questão. A experimentação a confirma ou não.

Disponibilizando para o programa um processo semelhante ao descrito acima, adaptado para o caso que estamos estudando, teríamos um procedimento indutivo que produziria diferentes saídas do programa $\mathcal{A}_{lea}$: isso corresponderia à investigação inicial do fenômeno. A Tabela 10 mostra cinco resultados (reais) de $\mathbb{A}_{lea}$ tendo como dados de entrada os números 7 e 2 como entrada do programa-produto e 9 como a saída desejada.

Tabela 10 – Resultados (reais) de $\mathbb{A}_{lea}$

n	Dados de entrada: $\{E_o = \{7,2\}, S_o = \{9\}\}$ Resultados
1	`9 INP INP + PRT + PRT 2 7`
2	`IF0GTO 9 8 ALET e 5 INP INP + PRT 2 7`
3	`LBL 7 2 INP INP + PRT 2 7`
4	`LBL 9 INP INP + PRT LET a 2 7`
5	`PRT INP INP LET b + PRT POP 2 7`

O resultado é uma lista à qual se pode aplicar o procedimento dedutivo que extrai os elementos comuns de todos os itens, como no caso anterior, só que produzida por uma experimentação indutiva. O resultado seria "`INP INP + PRT 2 7`",[6] o mesmo se fosse aplicado o programa $\mathcal{T}_{r1}$.

Outra característica da investigação indutiva é que os resultados obtidos tendem a reforçar ou enfraquecer a hipótese dada. Nesse sentido, pode-se conjeturar que as redes neurais são exemplos de mudança de hábito por raciocínio indutivo, uma hipótese a ser confirmada por investigação futura.

[6] Observemos que nesse caso aparecem também os dados de entrada, que são os número ao final do programa (cf. Apêndice B).

6.4 Possibilidades de diálogo

Os procedimentos de investigação abdutiva, dedutiva e indutiva vistos até agora teriam de estar à disposição do programa na sua lista $\mathbb{H}$ para que a investigação pudesse ser levada a cabo. Aliás, várias versões de procedimentos abdutivos, dedutivos e indutivos poderiam estar à disposição do usuário, cada um deles podendo ser executado com diferentes critérios de parada. Com os procedimentos vistos até o momento, podemos imaginar ao menos dois caminhos em que o diálogo levaria a investigação à solução do problema, ou seja, eliminar os passos redundantes do resultado de $\mathcal{A}_{lea}$:

I Via abdutiva – dedutiva:

1. Investigação abdutiva sobre o ícone "`LET a + INP INP + PRT`", levando ao procedimento $\mathcal{T}_{r2}$.
2. Investigação dedutiva "força bruta" sobre os valores numéricos em $\mathcal{T}_{r2}$, levando a uma lista de soluções viáveis.
3. Investigação dedutiva a respeito dos elementos comuns nas soluções viáveis encontradas, resultando num procedimento otimizado.

II Via indutiva – dedutiva:

1. Investigação indutiva exploratória sobre $\mathcal{A}_{lea}$, resultando numa lista de soluções viáveis.
2. Investigação dedutiva a respeito dos elementos comuns nas soluções viáveis encontradas, resultando num procedimento otimizado.

Ou seja, a partir do momento c_d em que é feita a escolha pelo diálogo, o usuário, ao invés de efetuar as escolhas c_{d_i} que levariam à otimização do procedimento com base na sua própria compreensão da solução, teria ao menos dois caminhos de escolhas c'_{d_i} nos quais as ações o levariam ao resultado final sem que precisasse compreender a solução por si próprio, apenas os passos da investigação que efetua.

É importante observar que qualquer um dos dois caminhos constituem, de forma limitada, uma série de escolhas c'_{d_i} que levariam à eliminação de passos redundantes de qualquer saída de $\mathcal{A}_{lea}$. O algoritmo procurado $\mathcal{O}_1$ poderia ser implementado pela automatização das escolhas c'_{d_i} feitas no diálogo. O programa assim obtido não seria uma solução geral, como é o algoritmo $\mathcal{T}_{r1}$, pois poderia apresentar soluções incorretas. Por exemplo, se adotada a via II, um problema que poderia encontrar é o caso em que uma das soluções geradas espontaneamente por $\mathcal{A}_{lea}$ estar incorreta. Uma solução que imprima, por acaso, o número correto, sem realizar a soma, se encaixa nessa descrição. Nesse caso, a solução final que seria encontrada não realizaria a soma desejada, mas, provavelmente, imprimiria um número qualquer. O mesmo erro poderia ocorrer, embora por outras razões, no caso da via I ser adotada. A falibilidade da solução a que

se poderia chegar, não obstante, é aderente à ideia de que não se pode esperar um resultado exato de um diálogo entre agentes autônomos, se esse diálogo admite um certo grau de vagueza e indeterminação, como é o caso.

O fato de que se poderia obter alguma solução, ainda que parcialmente correta, através do diálogo, levanta a questão se seria possível obter essa mesma solução de maneira mais autônoma. Por exemplo, ao invés de o usuário propor a cada passo qual forma de raciocínio aplicar, e com quais parâmetros de parada, poderia propor uma série pré-determinada de passos de raciocínio, por exemplo, abdução-dedução-indução. Outra solução para implementar uma investigação mais autônoma pode ser a adoção de um raciocínio abdutivo que decida qual ou quais os próximos passos de investigação. O critério de parada poderia ser encontrar um procedimento geral para a eliminação de passos redundantes nos resultados de $\mathcal{A}_{lea}$, expresso formalmente na Equação 6.1. Isso levaria a investigação autônoma um passo adiante. Tais soluções exigiriam uma critério de parada mais geral, e sugerem mais um caminho para futuras investigações.

Há ainda uma outra alternativa: o registro do procedimento investigativo efetuado pelo usuário humano permite criar um repertório de soluções, que poderia ser investigado. Isso permitiria criar um procedimento investigativo para generalizar o próprio procedimento investigativo, levando em conta as decisões humanas. Para isso pode ser interessante uma maneira mais detalhada de comunicar, entre programa e usuário, o andamento da investigação. Sugerimos o estudo de como implantar os conceitos de proposição e abstração.

6.5 Estudo de implementação de proposições

Na Seção 6.2 vimos como dar ao programa condições de mudar os próprios hábitos, e como isso cria condições para registrar os resultados da execução de procedimentos, em particular do procedimento que cria sequências de comandos. Na Tabela 9 temos, em particular, o procedimento que registra a lista de todas as execuções desse procedimento. Essa lista, nomeada “log-Alea”, contém todas as entradas e todas as saídas de todas as execuções já feitas do procedimento $\mathcal{A}_{lea}$.

A lista de execuções de $\mathcal{A}_{lea}$ teria um conteúdo como o que pode ser visto na Tabela 3. A linguagem de programação utilizada para a construção dos programas-produto, lembremos, está detalhada no Apêndice B. Observemos também que alguns dos programas-produto mostrados na Tabela 3 contém instruções redundantes. Os dados da tabela podem ser tomados como proposições. A primeira linha diz, por exemplo, que “O programa $\mathcal{A}_{lea}$ recebe como entrada ‘$\{E_o = \{3,5\}, S_o = \{8\}\}$’ e produz o programa-produto ‘`LET a + INP INP + PRT`’”. A frase, lembremos, expressa uma proposição, mas não é uma. Como vimos na Seção 5.4.4, o que caracteriza a proposição é sua participação na investigação, e não sua expressão. No caso específico do programa $\mathcal{O}_1$, queremos que ele elimine passos redundantes nas sequências geradas

pelo programa $\mathcal{A}_{lea}$. "Gerar proposições", nesse contexto, consiste em tomar os elementos que as compõem e tratá-los do mesmo modo que os elementos das proposições são tratados no processo de investigação.

6.5.1 Conceitos

Introduziremos, então, alguns conceitos que poderiam ser implementados no programa. São conceitos inspirados na filosofia peirciana[7] e podem ser mapeados no exposto nos Capítulos 2 e 3. O primeiro conceito é o conceito de "cognoscível". Definimos um cognoscível como algo que, como o nome diz, pode ser conhecido, embora seja possível não saber nada a respeito dele. Seu papel é o de funcionar, no programa, como uma secundidade pura. Serão os sujeitos das proposições. Há uma lista $\mathbb{K}$ de cognoscíveis no programa. Cada item dessa lista é, basicamente, um número, que é único na lista, permanece o mesmo sempre e identifica o cognoscível para o programa. Uma outra informação que pode aparecer em cada item da lista $\mathbb{K}$ é uma descrição do cognoscível. Essa descrição não é utilizada pelo programa: serve para auxiliar o usuário programador. Em princípio, cada cognoscível tem sua própria identidade, embora a evolução da investigação possa levar à conclusão de que dois cognoscíveis são, na realidade, o mesmo, caso em que ambos partilham entre si a relação de identidade.

Isso nos leva a um outro elemento: a relação. É o elemento que relaciona um ou mais cognoscíveis, formando um diagrama. As relações, de certa forma, atribuem características aos elementos relacionados. São compostas por "vagas", "descrições" e "procedimentos", e são registradas numa lista $\mathbb{R}$ do programa.

Vagas são espaços vazios que serão preenchidos por cognoscíveis. A cada vaga pode corresponder um e somente um cognoscível. Não há limite para o número de vagas possíveis em uma relação.[8]

Descrições são textos relacionando os espaços vazios da relação entre si. Esses textos servem apenas para auxiliar o usuário programador a compreenderem o que está acontecendo e em princípio não são levados em consideração pelo programa.

Procedimentos são os procedimentos, ou hábitos, que no programa estabelecem a relação em questão entre as vagas. Estão na lista $\mathbb{H}$, e as suas entradas e saídas estão associadas a vagas da relação.

[7] Uma inspiração importante desses conceitos, que não aparece no presente trabalho, provém do estudo dos grafos existenciais de Peirce, que têm um resumo publicado por Roberts (1992). O mesmo autor publicou muito antes um livro sobre o assunto (ROBERTS, 1973). Esses grafos implementam na forma de diagramas desenhados a lógica elementar como conhecemos hoje.

[8] Peirce dizia que o maior número de elementos necessários numa relação era três; pois combinando-se relações ternárias é possível criar relações de qualquer n-aridade (CP 1.354–368, especialmente 1.363, 1890). Furtamo-nos a adotar essa premissa por simplicidade.

Pode haver procedimentos em $\mathbb{H}$ que criam elementos que podem ser cognoscíveis, e relações entre cognoscíveis. Por exemplo, o procedimento $\mathcal{A}_{lea}$ cria uma relação que pode ser expressa pela frase "O programa-produto '`LET a + INP INP + PRT`' é tal que, recebendo como entrada $\{3,5\}$, produz como saída $\{8\}$". O programa $\mathcal{A}_{lea}$ determina qual programa-produto é capaz de gerar uma saída dada uma entrada. Esses três elementos: a entrada, a saída e o programa-produto se relacionam entre si de acordo com a relação acima. Essa relação é uma relação diferente da proposição "O programa $\mathcal{A}_{lea}$ recebe como entrada '$\{E_o = \{3,5\}, S_o = \{8\}\}$' e produz o programa-produto '`LET a + INP INP + PRT`'". Essa segunda relação é expressa por uma relação especial de determinação que veremos mais à frente.

Há cinco relações particularmente importantes. Três delas podem ser determinadas a priori: a relação de identidade, como vimos, a relação de predicação e a relação de determinação. As outras duas são relações especiais de determinação: determinação da execução de um procedimento pelo programa em si, e incorporação, ou corporificação, de um ícone.

A relação de identidade tem duas vagas e é uma relação especial, pois não "existe": se uma relação de identidade é estabelecida entre dois cognoscíveis, o resultado disso é que um deles pode deixar de existir, podendo ser excluído da lista $\mathbb{K}$, e todas as relações em que participa podem ser "transferidas" para o cognoscível remanescente (podendo gerar duplicidade de relações). Ou seja, a relação de identidade não relaciona dois elementos; ela apenas declara que há um mesmo cognoscível que aparece com dois registros diferentes na lista $\mathbb{K}$. Isso pode ocorrer porque o conhecimento a respeito de um conjunto de cognoscíveis tende a aumentar à medida em que as investigações avançam. É possível, por exemplo, que em um estágio de conhecimento a teoria supunha haver dois dois cognoscíveis que, com o aumento do conhecimento, se percebe que são, na verdade, um só.

A relação de predicação associa um ícone ao cognoscível; ela funciona como a cópula proposicional entre o cognoscível e o ícone, que é seu predicado. O que nos leva a introduzir o conceito de ícone *no contexto do programa*: qualquer coisa que não seja um cognoscível pode ser um ícone. Portanto, uma relação também pode ser um ícone.

A relação de determinação é uma relação especial porque possui uma vaga especial: a vaga do legissigno. Trata-se da relação que existe entre os legissignos e suas instâncias, ou *tokens*, no termo peirciano. São possíveis inúmeros tipos de relação de determinação. Tomemos, por exemplo, cognoscíveis que queremos tomar como um conjunto, que é também um cognoscível. Há uma ou mais relações de pertencimento entre o conjunto e seus membros. Por exemplo, um conjunto "casal *ab*" (cognoscível c_{ab}) contém dois "cônjuges": cognoscíveis a e b. Há aqui ao menos duas relações de pertencimento: a relação r_1, "a pertence ao conjunto c_{ab}" e a relação r_2, "b pertence ao conjunto c_{ab}". Essas relações, que se dão entre cognoscíveis, estão subordinadas a uma lei, um hábito, que determina um casal. No caso, digamos que seja a "lei do casamento",

designada aqui por L_C. Podemos dizer, então, que L_C ocupa o lugar do legissigno nas relações "L_C determina r_1" e "L_C determina r_2".

Em particular, cada procedimento h de $\mathbb{H}$ pode ser um cognoscível, além de ser um legissigno. Então, se o procedimento h cria uma relação, ou mesmo um outro cognoscível, esse procedimento h, como cognoscível, pode ocupar a posição especial de legissigno na relação de determinação. Dessa forma, o procedimento h, quando executado, pode criar cognoscíveis e/ou estabelecer relações, que podem ser entre cognoscíveis e/ou entre cognoscíveis e ícones. Esses novos elementos criados, sejam cognoscíveis ou relações, podem ser considerados resultados de uma instância do procedimento h e, portanto, se relacionam a ele por determinação. Um exemplo dessa relação de determinação corresponde à proposição "O programa $\mathcal{A}_{lea}$ recebe como entrada '$\{E_o = \{3,5\}, S_o = \{8\}\}$' e produz o programa-produto '`LET a + INP INP + PRT`'".

Uma relação especial de determinação exige uma explicação separada. No contexto do programa, os únicos legissignos são os procedimentos à sua disposição na lista $\mathbb{H}$. Um procedimento pode gerar um resultado a partir de uma entrada. Se isso ocorre, então há uma relação de determinação decorrente do fato de o procedimento ter sido executado pelo programa. Essa relação é independente da natureza ou do resultado do procedimento. Nessa relação o legissigno é o próprio programa, que determina as consequências da execução do procedimento. Essas consequências formam outra relação com formato geral: "a execução do procedimento x, com dados de entrada y, na situação z, determinou a saída w". Essa relação de determinação não pode ser confundida com a explicada no parágrafo anterior. Aqui, toda vez que se executa um procedimento que está em $\mathbb{H}$ é possível criar a relação de determinação, pelo programa em si, da execução do procedimento.

Um último caso importante de relação de determinação: pode haver um cognoscível que corresponde ao usuário programador. Também é um legissigno que atua na criação de cognoscíveis e da relação "determinado pelo usuário". Isso permite estabelecer cognoscíveis arbitrariamente, e relações arbitrárias entre os cognoscíveis, comandadas pelo usuário. Esses cognoscíveis, e essas relações, poderão ser tratados,mais tarde, como proposições pelo programa.

Finalmente, há a relação que instancia o processo de abstração — a incorporação, ou corporificação. O ato de abstrair é um legissigno que atua quando algo que ainda não é cognoscível, por exemplo um predicado proposicional, começa a ser investigado. Esse algo, que será investigado, se tornará sujeito de outras proposições. É preciso um cognoscível para ele. Esse cognoscível se relaciona ao que é investigado: o cognoscível, que é a abstração, é a corporificação do que está sendo investigado, e a partir daí é possível investigá-lo. Pois, corporificado em um cognoscível, é possível, agora, atribuir predicados a ele. Por exemplo, "açúcar é doce" é uma relação de predicação entre o cognoscível "açúcar" e o ícone "doce". A abstração "doçura" corporifica o ícone "doce".

6.5.2 Proposições

Com esses elementos temos condições de representar, na Tabela 12, um conjunto de proposições de um modo passível de investigação. Para fazer isso, vamos, antes, determinar uma lista das relações possíveis $\mathbb{R}$, na Tabela 11. Algumas informações não aparecerão na tabela, sendo compreensíveis a partir do texto. Nesta tabela de relações, as vagas, no texto, aparecem com o formato "v_i": v_1, v_2 etc. A letra X representa uma incógnita qualquer. As colunas de cor cinza mostram dados utilizados somente pelo usuário programador, sendo ignorados pelo programa.

Tabela 11 – Relações disponíveis para descrição de resultados de $\mathcal{A}_{lea}$

	Número de vagas	Texto descritivo	Nome	Procedimento
1	1	v_1 é "X"	Predicação	Explicação no texto
2	2	v_1 é v_2	Identidade	Explicação no texto
3	2	v_1 determina v_2	Determinação	Explicação no texto
4	2	a execução de v_1, com entrada v_2, nas circunstâncias v_3, determinou a saída v_4	Execução	Explicação no texto
5	3	o programa v_1 recebe v_2 como entrada e retorna v_3 como saída	Relação $\mathcal{A}_{lea}$	v_1 é saída de $\mathcal{A}_{lea}(v_2, v_3)$
6	2	v_1 corporifica v_2	Corporificação	Explicação no texto

Com as relações estabelecidas na Tabela 11, vejamos, por exemplo, algumas proposições associadas ao primeiro item da Tabela 3 (nesse item os dados são: entrada = $\{3, 5\}$, saída = 8 e procedimento = `LET a + INP INP + PRT`). Lembremos que essa tabela corresponde às execuções do procedimento $\mathcal{A}_{lea}$ como seriam registradas pelo programa $\mathcal{O}_1$. Para facilitar a leitura, na Tabela 12 cada cognoscível será representado no formato k_i, onde i é o número associado ao cognoscível; nela, também, as colunas cinza são utilizadas apenas pelo usuário programador; a coluna "Linha da relação" indica em que linha da Tabela 11 está a relação explicitada. A coluna "Vaga" mostra a qual vaga no texto da relação corresponde o cognoscível da coluna "Cognoscível". Algumas considerações a respeito da Tabela 12:

- É uma tabela parcial. Muitas relações possíveis foram omitidas.
- As linhas 19, 20 e 21 mostram como aparece algo determinado pelo usuário; k_8 (linha 19) é a relação de predicação mostrada na linha 3 — que apresenta o cognoscível k_2 de uma maneira escolhida para que o usuário programador possa entendê-lo, já que não entramos em detalhes quanto à relação de predicação — tomada como cognoscível e, como tal, definida pelo usuário. No nosso atual estágio de conhecimento, todas as relações de predicação só podem ser definidas assim.

Tabela 12 – Possíveis proposições decorrentes de um resultado de $\mathcal{A}_{lea}$

	Cog-noscível	Relação	Linha da relação	Vaga	Texto
1	k_0	Predicação	1	v_1	k_0 é "o usuário programador"
2	k_1	Predicação	1	v_1	k_1 é "o programa em si"
3	k_2	Predicação	1	v_1	k_2 é "`LET a + INP INP + PRT`"
4	k_3	Predicação	1	v_1	k_3 é "$\{3,5\}$"
5	k_4	Predicação	1	v_1	k_4 é "8"
6	k_2	Relação $\mathcal{A}_{lea}$	5	v_1	o programa k_2 recebe k_3 como entrada e retorna k_4 como saída
7	k_3	Relação $\mathcal{A}_{lea}$	5	v_2	o programa k_2 recebe k_3 como entrada e retorna k_4 como saída
8	k_4	Relação $\mathcal{A}_{lea}$	5	v_3	o programa k_2 recebe k_3 como entrada e retorna k_4 como saída
9	k_5	Predicação	1	v_1	k_5 é "$\mathcal{A}_{lea}$"
10	k_6	Predicação	1	v_1	k_6 é "o programa k_2 recebe k_3 como entrada e retorna k_4 como saída"
11	k_5	Determinação	3	v_1	k_5 determina k_6
12	k_6	Determinação	3	v_2	k_5 determina k_6
13	k_2	Execução	4	v_4	a execução de k_5, com entrada k_3, k_4, determinou a saída k_2
14	k_3	Execução	4	v_2	a execução de k_5, com entrada k_3, k_4, determinou a saída k_2
15	k_4	Execução	4	v_2	a execução de k_5, com entrada k_3, k_4, determinou a saída k_2
16	k_5	Execução	4	v_1	a execução de k_5, com entrada k_3, k_4, determinou a saída k_2
17	k_7	Predicação	1	v_1	k_7 é "a execução de k_5, com entrada k_3, k_4, determinou a saída k_2"
18	k_1	Determinação	3	v_1	k_1 determina k_7
19	k_8	Predicação	1	v_1	k_8 é "k_2 é '`LET a + INP INP + PRT`'"
20	k_0	Determinação	3	v_1	k_0 determina k_8
21	k_8	Determinação	3	v_2	k_0 determina k_8
⋮	⋮	⋮	⋮	⋮	⋮

- As relações das linhas 1 a 5, e em particular as das linhas 3 a 5 têm como predicado elementos que não são propriamente ícones, embora possam em princípio serem tomados como ícones de algo. Voltaremos a essa questão mais adiante.
- Das linhas 6 a 18 vemos as consequências de uma execução do procedimento $\mathcal{A}_{lea}$:
  - As linhas 6 a 8 descrevem a relação criada por $\mathcal{A}_{lea}$.
  - A linha 9 introduz o cognoscível k_5 de uma maneira que o usuário programador entenda que estamos nos referindo a $\mathcal{A}_{lea}$.
  - A linha 10 introduz o cognoscível k_6 de modo que entendamos que se trata da relação criada por $\mathcal{A}_{lea}$ (descrita nas linhas 6 a 8).
  - As linhas 11 e 12 estabelecem que k_6 é uma instância da execução de $\mathcal{A}_{lea}$, que é um legissigno.
  - Uma vez que $\mathcal{A}_{lea}$ (k_5) é um legissigno, e k_6 produto de sua execução, podemos afirmar que o programa em si (k_1) é o "responsável" pela execução de $\mathcal{A}_{lea}$ (k_5),

usando como dados k_3 e k_4 (linhas 4 e 5) e tendo como saída k_2 — eliminamos neste exemplo as circunstâncias de execução (data, etc) por simplicidade. Essa relação é descrita para o programa das linhas 13 a 16, é tornada cognoscível na linha 17 (que introduz k_7, a própria relação), e é explicitada como sendo determinada pelo programa em si na linha 18.

Finalmente, uma vez que k_2 foi também gerado por $\mathcal{A}_{lea}$ (k_5), a relação "k_5 determina k_2" também poderia aparecer na lista como consequência da execução de $\mathcal{A}_{lea}$.

Adicionalmente, há uma relação entre k_2, k_3 e k_4 que está oculta ao programa: é o fato de o programa k_2, quando executado tendo como entrada k_3, resulta em k_4. Essa é uma proposição diferente de k_6, na linha 10, (k_6 é "o programa k_2 recebe k_3 como entrada e retorna k_4 como saída"), pois k_6 é uma instância do procedimento $\mathcal{A}_{lea}$ (cognoscível k_5), como vemos nas linhas 11 e 12. A relação oculta não é gerada por nenhum procedimento à disposição do programa. Tal procedimento poderia, não obstante, ser implementado no círculo funcional, e incluído em $\mathbb{H}$, e a relação que cria incluída em $\mathbb{R}$, e teríamos então condições de gerar proposições a partir dele.

6.5.3 Abstrações

Uma vez que a abstração pode ser entendida como a corporificação do ícone de uma proposição, para representar uma abstração hipostática a partir de uma proposição qualquer basta criar um cognoscível que corporifique o ícone desta proposição, numa relação de determinação. Por exemplo, tomando a linha 3 da Tabela 12, podemos, como usuários programadores, corporificar seu predicado (que é "`LET a + INP INP + PRT`"), resultando na Tabela 13. Essa tabela contém na linha 4 uma relação de corporificação. Nela está a proposição que associa "`LET a + INP INP + PRT`" a um cognoscível, k_9. As linhas 5, 6 e 7 "explicam" que isso foi determinado pelo usuário. A diferença entre o cognoscível k_2 e k_9 é que k_2 é o programa: ele é executado, ele

Tabela 13 – Um exemplo de abstração

	Cognoscível	Relação	Linha da relação	Vaga	Texto
1	k_0	Predicação	1	v_1	k_0 é "o usuário / programador"
2	k_1	Predicação	1	v_1	k_1 é "o programa em si"
3	k_2	Predicação	1	v_1	k_2 é "`LET a + INP INP + PRT`"
4	k_9	Corporificação	6	v_1	k_9 incorpora "`LET a + INP INP + PRT`"
5	k_{10}	Predicação	1	v_1	k_{10} é "k_9 incorpora '`LET a + INP INP + PRT`'"
6	k_0	Determinação	3	v_1	k_0 determina k_{10}
7	k_{10}	Determinação	3	v_2	k_0 determina k_{10}

cria relações entre entradas e saídas. Já k_9 é o texto "`LET a + INP INP + PRT`". Assim como as proposições, não basta que a abstração seja representada. Os recursos para investigá-la são os procedimentos que implementam raciocínios abdutivos, dedutivos e indutivos na estrutura.

6.5.4 Propósito

Podemos agora descrever como um propósito pode ser traduzido nessa estrutura. Como os demais elementos, não basta uma frase que o descreva: é preciso que o agente atue sobre ele como tal.

Para descrever um propósito aproveitaremos do fato de que a abstração, como descrita acima, permite o registro de algo ainda desconhecido. Um propósito de investigação, que é uma pergunta, pode ser apresentado estruturando um conjunto de cognoscíveis e descrevendo o que se busca, e como o programa saberá quando o que é buscado foi encontrado, ou seja, um critério de parada. Tomemos como exemplo um propósito que poderia levar à investigação abdutiva estudada na Seção 6.3.1. Ela procura um procedimento que tome um programa-produto de $\mathcal{A}_{lea}$ e retorne um programa-produto válido, mas com menos instruções. Para fazer essa descrição, utilizaremos novos procedimentos, que implementam novas relações na lista $\mathbb{R}$. Essas novas relações estão na Tabela 14, complementar à Tabela 11. Observa-se que incluímos na linha 9 o

Tabela 14 – Relações adicionais para descrição do propósito do raciocínio abdutivo

	Número de vagas	Texto descritivo	Nome	Procedimento
7	2	v_1 possui v_2 palavras	Conta-palavras	v_2 é saída de `pContaPalavras(` v_1 `)`
8	2	v_1 pode ser reduzido a v_2	Relação procurada	v_2 é saída do procedimento procurado tendo v_1 como dado de entrada
9	3	v_1 recebe v_2 como entrada e retorna v_3	Processamento tipo $\mathcal{A}_{lea}$	v_3 é saída do processamento de v_1 na linguagem de $\mathcal{A}_{lea}$ tendo como entrada v_2
10	2	v_1 é menor que v_2	Comparação menor que	A relação é criada pelo procedimento `pMenorQue(` v_1, v_2 `)` se v_1 for menor que v_2

processamento na linguagem de $\mathcal{A}_{lea}$, que estava oculta ao programa e relaciona k_2, k_3 e k_4 como apresentados na Tabela 12.

Tendo adicionado essas relações, podemos descrever o propósito através da estrutura de cognoscíveis da Tabela 15. Essa tabela repete as linhas 1 a 8 da Tabela 12 para caracterizar o cognoscível k_2, que é o programa que queremos "encurtar". Vemos na linha 12 o cognoscível que incorpora um procedimento. Sabe-se que se trata de um procedimento porque vemos, nas linhas 15 e 16, que cria a relação descrita nas linhas 13 e 14. Essa linhas também informam que o dado de entrada do procedimento é o cognoscível k_2, e a saída é o cognoscível k_{13}. A respeito de k_{13} sabemos: (1) da linha 17, que recebe k_3 como entrada e retorna k_4, desde que seja processado como um programa na linguagem usada por $\mathcal{A}_{lea}$ — esse relacionamento possui mais duas linhas, não mostradas na tabela. Sabemos também (2) que se contarmos as palavras

Tabela 15 – Proposições de um possível propósito de raciocínio abdutivo: encontrar o predicado da linha 12

	Cognoscível	Relação	Linha da relação	Vaga	Texto
1	k_0	Predicação	1	v_1	k_0 é "o usuário programador"
2	k_1	Predicação	1	v_1	k_1 é "o programa em si"
3	k_2	Predicação	1	v_1	k_2 é "`LET a + INP INP + PRT`"
4	k_3	Predicação	1	v_1	k_3 é "$\{3, 5\}$"
5	k_4	Predicação	1	v_1	k_4 é "8"
6	k_2	Relação $\mathcal{A}_{lea}$	5	v_1	o programa k_2 recebe k_3 como entrada e retorna k_4 como saída
7	k_3	Relação $\mathcal{A}_{lea}$	5	v_2	o programa k_2 recebe k_3 como entrada e retorna k_4 como saída
8	k_4	Relação $\mathcal{A}_{lea}$	5	v_3	o programa k_2 recebe k_3 como entrada e retorna k_4 como saída
9	k_{11}	Predicação	1	v_1	k_{11} é "7"
10	k_2	Conta-palavras	7	v_1	k_2 possui k_{11} palavras
11	k_{11}	Conta-palavras	7	v_2	k_2 possui k_{11} palavras
12	k_{12}	Incorporação	6	v_1	k_{12} incorpora o procedimento procurado
13	k_{13}	Relação procurada	8	v_2	k_2 pode ser reduzido a k_{13}
14	k_2	Relação procurada	8	v_1	k_2 pode ser reduzido a k_{13}
15	k_{14}	Predicação	1	v_1	k_{14} é "k_2 pode ser reduzido a k_{13}"
16	k_{12}	Determinação	3	v_1	k_{12} determina k_{14}
17	k_{13}	Processamento tipo $\mathcal{A}_{lea}$	9	v_1	k_{13} recebe k_3 como entrada e retorna k_4
18	k_{15}	Conta-palavras	7	v_2	k_{13} possui k_{15} palavras
19	k_{13}	Conta-palavras	7	v_1	k_{13} possui k_{15} palavras
20	k_{15}	Comparação menor que	10	v_1	k_{15} é menor que k_{11}

de k_{13} obteremos o cognoscível k_{15}, que deve ser menor que k_{11}, o número de palavras dos dados de entrada (de acordo com as linhas 10 e 11). Assim, a estrutura permite definir o que não é conhecido através de uma abstração, e dá os critérios para que se saiba como teremos encontrado o que procuramos.

Com a descrição do que se busca, sabemos que a investigação deverá definir o predicado do cognoscível k_{12} na proposição que está na linha 12 da tabela. Para isso deverá encontrar, de acordo com a linha 16, algo que determine k_{14}, descrito na linha 15: algo que reduza o programa k_2 a algo que tenha um menor número de palavras que k_2 (linha 20) mas que apresente os mesmos resultados (linha 17).

A estrutura proposta nesta Seção 6.5, ainda que aproximadamente implementada, permite não só o tratamento de proposições, abstrações e propósitos pelo programa, mas também seu registro de maneira inteligível para o usuário programador. Isso permite inferir algumas consequências da disponibilização dessa estrutura, e levantar questões pertinentes para futura investigação.

6.6 Algumas soluções a detalhar

A estrutura sugerida até aqui podem ser um primeiro passo para uma estrutura dialógica de programação. Mas, para isso, há desafios que podem demandar uma solução mais adequada. Elencamos a seguir algumas das questões que podem ser alvo de estudos suplementares.

O escopo do programa e seu círculo funcional

Uma das interrogações que surgem é: qual o repertório ideal de procedimentos no círculo funcional do programa? Qual a lista $\mathbb{H}$ ideal? A resposta depende, claro, da finalidade que queremos dar a ela. No nosso exemplo, trata-se de investigar programas de computador; é a estrutura mais simples de propor se levarmos em conta o Umwelt do agente que é o próprio computador. E mesmo nesse caso, não são imediatamente claras as capacidades que gostaríamos de ter à disposição. Essa questão poderá ser melhor respondida numa implementação real de uma programação dialógica, dentro de um escopo melhor definido de atuação.

Uma lista de hábitos como forma de aprendizado

A maneira escolhida para determinar os hábitos e como se alteram foi colocá-los em uma lista. Essa alternativa tem a vantagem de ser acessível a quem não conhece detalhadamente a disciplina de criação de linguagens de programação e seus compiladores, como é o caso do autor. Tal escolha, entretanto, apresenta inúmeras desvantagens, entre elas:

- É preciso um algoritmo à parte, $\mathcal{E}_{th}$, para garantir a execução dos procedimentos. Essa algoritmo apresenta inúmeros problemas, como a dificuldade em lidar com a mudança nas circunstâncias, e a dificuldade de representar a ordem em que os hábitos são avaliados — essa ordem, no nosso caso, foi relegada às circunstâncias de execução, ou seja, não foram estudadas.
- Para funcionar efetivamente, os comandos na lista devem ser tais que a lista seja Turing-completa (cf. ROGERS, 1967), ou seja, o conjunto de comandos efetivamente disponível deve ter tanto poder de processamento quanto uma máquina de Turing automática.
- A introdução de um novo algoritmo na lista, a partir dos elementos antigos, nem sequer foi abordada. O fato de cada item necessitar um comando para execução evidencia o caráter indireto da atuação da lista.

É mais provável que uma interface mais eficiente para lidar com os hábitos do programa seja implementada através de uma linguagem dialógica de programação, com funcionamento semelhante, mas não igual, às linguagens de programação. Como vimos, deverá ser possível a execução de comandos cujo resultado não é, em princípio, conhecido. Há uma campo interessante de pesquisa na fundamentação teórica necessária para tal linguagem.

As listas de proposições

Como estão propostas na Seção 6.5, as proposições cumprem a função de descrever abstrações, propósitos e possíveis investigações e seus resultados. Sua leitura e compreensão, entretanto, não é trivial. Um desafio para a implementação real da programação dialógica é encontrar uma solução que viabilize uma forma de visualização mais amigável do conhecimento interno que o programa tem na forma de proposições.

Uma visão geral da implementação

Embora o estudo apresentado ilustre como os conceitos filosóficos de Peirce podem ser utilizados na efetiva implantação de uma solução dialógica de programação, não está claro como tal solução poderá ser implementada, nem qual será o resultado dessa implementação. Um caminho de pesquisa futura consiste em determinar um escopo de atuação e construir um sistema que incorpore as capacidades determinadas neste trabalho.

Apesar dessas limitações, podemos vislumbrar possibilidades de atuação que justificam pesquisa ulterior.

6.7 Possibilidades emergentes

As possibilidades que enumeraremos a seguir são uma enumeração não exaustiva de possíveis rotas de pesquisa às quais pode interessar o detalhamento da especificação e a implementação real da estrutura proposta.

Programação para não programadores

Como fica claro, a estrutura proposta é capaz de produzir programas a partir de descrições vagas. Uma ideia que pode ser levada adianta é a possibilidade de disponibilizar um processo de programação que pode ser levado adiante sem que o programador tenha conhecimento de linguagens formais de programação, aproveitando a característica dialógica da interação.

Há inúmeros escopos em que isso pode ser desejável hoje em dia: nas disciplinas de Big Data e análise massiva de dados, na construção de sistemas de informação baseados em aprendizado de máquina, na busca por soluções *ad hoc* para problemas de negócio.

Tratam-se de escopos que, diferente do escopo de programação de computadores, incorporam muitos mais elementos indeterminados e vagos que precisam, de alguma forma, ser encaminhados pelos usuários. Aqui pode haver um interesse genuíno na programação dialógica de computadores.

Exploração em ciência da computação

Como vimos na Seção 4.3.1, as máquinas-oráculo são utilizadas no estudo de problemas de decidibilidade e computabilidade. Não existem máquinas-oráculo de acordo com a definição de Turing (1939).

A forma de raciocínio abdutivo que propomos não corresponde à máquina-oráculo. Tampouco corresponde à máquina de Turing automática: possui a característica de ser não computável, sendo capaz de encontrar procedimentos computáveis gerais, ainda que com probabilidades ínfimas de acerto.

O teorema de Post (KLEENE, 1952) determina o grau de incomputabilidade de certos conjuntos com base na complexidade de sua descrição. Uma possibilidade de pesquisa é buscar relações entre as descrições de conjuntos obteníveis pela sucessiva aplicação dos raciocínios propostos e as descrições de conjuntos de acordo com níveis mais altos da hierarquia aritmética de Kleene, que determinam a complexidade do que pode ou não ser computável por uma máquina-oráculo. Esse talvez seja um caminho para encontrar uma relação entre as máquinas-oráculo e os raciocínios possíveis pela programação dialógica, uma conjetura que é reforçada pelo trabalho de Davis (2006, p. 128), exibido na Seção 4.3.1.

Tratamento de vagueza e linguagens naturais

O mecanismo de geração de abstrações — criação de cognoscível que incorpora um ícone a ser investigado — possibilita o tratamento da vagueza: tudo o que não pode ser imediatamente processado pelo programa pode ser associado a um cognoscível que o incorpora. Isso cria a possibilidade de dar tratamento ao que é vago, uma característica intrínseca da linguagem natural (cf. NÖTH; SANTAELLA, 2011) que, se incorporada aos programas de computador, permitiria mudar o paradigma de utilização dessas máquinas (cf. GAZONI, 2018).

7 Conclusões

A questão fundamental da pesquisa indaga: compreendendo os elementos básicos da agência semiótica à luz da filosofia de Peirce, é possível tomar o computador como um agente semiótico e dessa forma propor novas formas para sua programação?

A resposta afirmativa a essa questão passa por uma conceituação peirciana de agência semiótica, que de fato nunca foi exposta por Peirce como um capítulo de sua obra. A visão unificada e consistente desse e de outros elementos da filosofia peirciana continua um trabalho em andamento, recrutando inúmeros intérpretes de sua obra. Essa primeira fase da pesquisa teve como resultados as visões dos processos de inferência e aprendizado, e da participação do entorno na formação do raciocínio. Tais visões mostraram-se suficientes para propor uma articulação entre esses elementos no contexto dos computadores eletrônicos.

O estudo dos computadores e funções computáveis, isolado no Capítulo 4, apontou questões a respeito dos limites teóricos e suas diferenças dos limite práticos da computação. Essas questões se revelaram importantes à luz da moldura conceitual pela qual as analisamos, e permitiram sugerir uma estrutura de atuação, que, ainda que tenha sido iniciada com a publicação do programa de Gazoni (2016) (aqui batizado $\mathcal{A}_{lea}$), precisa passar pelo teste da implementação para revelar sua real usabilidade: um tema para pesquisa futura. Mesmo assim, independentemente da sua implementação ou não, a estrutura proposta expôs características que convidam ao desenvolvimento de pontos relevantes nas ciência da computação hoje, como por exemplo o teorema de Post e o processamento da linguagem natural.

Assim, consideram-se alcançados tanto os objetivos específicos quanto o objetivo geral, o que coloca a filosofia peirciana como moldura conceitual necessária para o futuro estudo da utilização autônoma dos computadores, numa atuação que talvez devesse ser considerada uma inteligência propriamente artificial.

Desdobramentos possíveis

Toda pesquisa constitui, de certa forma, um trabalho em andamento, e esta não é diferente de outras. Para além dos desdobramentos já apontados, as questões nesta seção expostas também despertam uma predisposição inquisitória.

Um outro Umwelt

A pesquisa limitou a aplicação da agência semiótica ao Umwelt e ao círculo funcional factível no computador. Poderia ser aplicada a agentes dotados de capacidades físicas diferentes,

tais como robôs. Nessa moldura, a agência semiótica ganha outros contornos. Por exemplo, na medida em que a fase de experimentação da investigação passa a incluir a possibilidade de atuar no mundo físico que compartilhamos com a máquina.

O trabalho de Camargo (2018) propõe um método de mapeamento de sistemas biológicos em sistemas computacionais baseado na semiótica de Peirce. Não é impossível que essa abordagem, estendida pela conceituação de autonomia mental, seja capaz de implementar soluções para problemas atuais como o dos automóveis autônomos.

Continuidade das qualidades

A filosofia de Peirce preconiza a essencialidade da noção de contínuo. Isso se aplica às qualidades: segundo Peirce, elas são contínuas. Essa premissa inspira uma solução para um problema técnico. De fato, da maneira pela qual a estrutura de aplicação da agência semiótica é proposta neste trabalho, as capacidades de um círculo funcional não podem ser utilizadas em outro. Isso implica que a agência semiótica está associada ao Umwelt do agente: um carro autônomo tem uma mente diferente de uma computador, por exemplo. Mas isso não precisa ser assim.

O conceito matemático de contínuo em Peirce é diferente dos conceitos mais usuais de contínuo, provenientes do trabalho de Cantor. Ainda assim julgamos possível utilizar essa ideia para implementar capacidades nos círculos funcionais de diferentes agentes de um modo mais geral. A ideia é ultrapassar os limites do Umwelt de cada agente individual, permitindo o reaproveitamento da agência semiótica desenvolvida numa certa plataforma em outras plataformas diferentes.

Interface com pesquisa corrente em inteligência artificial

As formas de raciocínio peircianas parecem encontrar algum eco em certas técnicas utilizadas em inteligência artificial hoje. É possível que haja um componente abdutivo na programação genética; o aprendizado das redes neurais tem algo do raciocínio indutivo.

Pesquisa ulterior pode apontar sinergias entre as diferentes abordagens, possibilitando aproveitar mutuamente os avanços alcançados. Em particular, uma abordagem filosoficamente mais estruturada da agência semiótica pode ter a contribuir no desenvolvimento da inteligência artificial.

Além disso, técnicas desenvolvidas para a inteligência artificial podem ser incorporadas ao círculo funcional de uma solução dialógica, aumentando seu poder de inferência autônoma.

Filosofia

A visão do computador como agente semiótico autônomo levanta questões filosóficas que se adicionam ao já intenso questionamento filosófico em torna da inteligência artificial. Por ter uma origem filosófica em sua base, a agência semiótica autônoma em computadores e seu alcance podem iluminar essas discussões, e trazer questões novas ao teatro de pesquisa.

Referências

AIGUIER, M. *et al.* Explanatory relations in arbitrary logics based on satisfaction systems, cutting and retraction. *International Journal of Approximate Reasoning*, 2018. Disponível em: <https://arxiv.org/abs/1803.01571>. Acesso em: 7.9.2018.

ARTIFICIAL NEURAL NETWORK. *In*: WIKIPEDIA. Wikimedia, 2019. Disponível em: <https://en.wikipedia.org/wiki/Artificial_neural_network>. Acesso em: 21.5.2019.

BALLABIO, A. The genesis of the creative experience in C. S. Peirce. *Cognitio*, v. 19, n. 2, p. 220–226, 2018.

BLOCH, I. *et al.* Morphologic for knowledge dynamics: revision, fusion, abduction. *CoRR – Computing Research Repository*, 2018. Disponível em: <https://arxiv.org/abs/1802.05142>. Acesso em: 7.9.2018.

BROOKS JR., F. P. No silver bullet – essence and accidents of software engineering. *In*: KUGLER, H. J. (ed.). *Information processing 86*. Amsterdam: Elsevier, 1986. v. 20, n. 1. Disponível em: <http://www.cs.nott.ac.uk/~cah/G51ISS/Documents/NoSilverBullet.html>.

____. *O mítico homem-mês, edição de 20o aniversário*. Tradução de Cesar Brod. Rio de Janeiro: Campus-Elsevier, 2009.

BUXTON, J. N.; RANDELL, B. (eds.). *Software Engineering Techniques: Report of a Conference Sponsored by the NATO Science Committee*. Rome, Italy, 27-31 Oct. 1969, Brussels, Scientific Affairs Division, NATO: [s.n.], 1970. Disponível em: <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1969.PDF>. Acesso em: 10.3.2014.

BYLANDER, T. *et al.* The computational complexity of abduction. *Artificial Intelligence*, v. 49, n. 1, p. 25–60, 1991. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/0004370291900055>. Acesso em: 7.9.2018.

CAMARGO, C. E. P. de. *Semiótica da vida artificial*. Tese (Doutorado), Tecnologias da Inteligência e Design Digital, Pontifícia Universidade Católica de São Paulo – PUC-SP, São Paulo, 2018.

CARNIELLI, W. A.; EPSTEIN, R. L. *Computabilidade, funções computáveis, lógica e os fundamentos da matemática*. São Paulo: Editora UNESP, 2006.

CHURCH, A. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, v. 58, n. 2, p. 345–363, Abril 1936. Disponível em: <http://www.jstor.org/stable/2371045>. Acesso em: 21.6.2014.

COSTA, N. C. A. da. *Introdução aos fundamentos da matemática*. 3. ed. São Paulo: Hucitec, 1992.

DAI, W.-Z. *et al.* Tunneling neural perception and logic reasoning through abductive learning. *CoRR – Computing Research Repository*, 2018. Disponível em: <https://arxiv.org/abs/1802.01173>. Acesso em: 7.9.2018.

DAVIS, M. The Church-Turing thesis: consensus and opposition. *In*: BECKMANN, A. *et al.* (eds.). *Logical Approaches to Computational Barriers*: Cie 2006. lecture notes in computer science, vol 3988. Heidelberg: Springer, 2006. p. 125–132.

DENECKER, M.; KAKAS, A. C. Abduction in logic programming. *In*: *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*. Londres: Springer-Verlag, 2002. p. 402–436. Disponível em: <http://dl.acm.org/citation.cfm?id=646001.675770>. Acesso em: 7.9.2018.

DESCARTES, R. *Discurso do método*. Tradução de Paulo Neves. Porto Alegre: L&PM, 2004. Introdução de Denis Lerrer Rosenfield.

DREYFUS, H. L. *What Computers Still Can't Do: A Critique of Artificial Reason*. Cambridge, MA: MIT Press, 1992.

FELDBACHER-ESCAMILLA, C. J.; GEBHARTER, A. Modeling creative abduction Bayesian style. *European Journal for Philosophy of Science*, Springer Nature, v. 9, n. 1, nov 2018.

FLACH, P. A.; KAKAS, A. C. On the relation between abduction and inductive learning. *In*: ____. *Abductive Reasoning and Learning*. Dordrecht: Springer, 2000. p. 1–33. Disponível em: <https://doi.org/10.1007/978-94-017-1733-5_1>. Acesso em: 7.9.2018.

GAZONI, R. M. Peircean semiotics: perspectives in computer science. *In*: *Proceedings of the 2015 International Conference on Foundations of Computer Science*. Las Vegas, NV: CSREA, 2015. p. 79–85.

GAZONI, R. M. *Semiótica da programação: levantamento crítico e perspectivas peirceanas*. Dissertação (Mestrado) — Tecnologias da Inteligência e Design Digital, Pontifícia Universidade Católica de São Paulo – PUC-SP, São Paulo, 2015.

____. Creative thinking in artificial intelligence: a Peircean account. *In*: *Proceedings of the 2016 International Conference on Computational Science and Computational Intelligence*. Las Vegas, NV: IEEE-CPS, 2016. p. 537–540.

____. A semiotic analysis of programming languages. *Journal of Computer and Communications*, v. 6, n. 3, p. 91–101, March 2018. Disponível em: <http://www.scirp.org/journal/PaperInformation.aspx?PaperID=83246>.

GÖDEL, K. *On Formally Undecidable Propositions of Principia Mathematica and Related Systems*. Translated by B. Meltzer, Introduction by R. B. Braithwaite. New York, NY: Dover, 1992.

GOLD, E. M. Limiting recursion. *The Journal of Symbolic Logic*, v. 30, n. 1, p. 28–48, 1965.

GUDWIN, R.; QUEIROZ, J. (eds.). *Semiotics and Intelligent Systems Development*. Hershey, PA: Idea Group, 2007.

JIANG, Y.; KORPAS, L. M.; RANEY, J. R. Bifurcation-based embodied logic and autonomous actuation. *Nature Communications*, v. 10, n. 128, 2019.

JUBA, B. Learning abductive reasoning using random examples. *In*: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*. New Orleans, LA: AAAI, 2016. p. 999–1007. Disponível em: <https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12186>. Acesso em: 7.9.2018.

JUBA, B.; LI, Z.; MILLER, E. Learning abduction under partial observability. *In*: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. New Orleans, LA: AAAI, 2018. p. 8097–8098. Disponível em: <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17323>. Acesso em: 7.9.2018.

KAKAS, A. C.; TONI, F.; KOWALSKI, R. A. Abductive logic programming. *Journal of Logic and Computation*, v. 2, n. 6, p. 719–770, December 1993.

KAUFFMAN, L. H. The mathematics of Charles Sanders Peirce. *Cybernetics and Human Knowing*, v. 8, n. 1-2, p. 79–110, 2001.

KLEENE, S. C. *Introduction to Metamathematics*. Amsterdam: North Holland, 1952.

LALOR, B. J. The classification of Peirce's interpretants. *Semiotica*, v. 114, n. 1/2, p. 31–40, 1997.

LISZKA, J. J. Peirce's interpretant. *Transactions of the Charles S. Peirce Society*, v. 26, n. 1, p. 17–62, 1990.

MACHINE LEARNING. *In*: WIKIPEDIA. Wikimedia, 2019. Disponível em: <https://en.wikipedia.org/wiki/Machine_learning>. Acesso em: 21.5.2019.

MARTY, R. *76 definitions of the sign by C. S. Peirce*. 1997. Disponível em: <http://www.iupui.edu/~arisbe/rsources/76DEFS/76defs.HTM>. Acesso em: 31.5.2019.

MERRELL, F. *Signs Grow: Semiosis and Life Processes*. Toronto: University of Toronto Press, 1996.

NAGEL, E.; NEWMAN, J. R. *Gödel's Proof*. New York, NY: New York University Press, 1986.

NAUR, P.; RANDELL, B. (eds.). *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee*. Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO: [s.n.], 1969. Disponível em: <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>. Acesso em: 8.3.2014.

NÖTH, W. On the instrumentality and semiotic agency of signs, tools, and intelligent machines. *Cybernetics and Human Knowing*, v. 16, n. 3-4, p. 11–36, 2009.

____. Human communication from the semiotic perspective. *In*: IBEKWE-SANJUAN, F.; DOUSA, T. M. (eds.). *Theories of Information, Communication, and Knowledge: A Multidisciplinary Approach*. Heidelberg: Springer, 2013. p. 97–119.

____. The growth of signs. *Σημειωτική: Sign Systems Studies*, v. 42, n. 2/3, p. 172–192, 2014.

____. The semiotics of models. *Σημειωτική: Sign Systems Studies*, v. 46, n. 1, p. 7–43, 2018.

____. *Manual de semiótica*. São Paulo: EDUSP, 2019. No prelo.

NÖTH, W.; SANTAELLA, L. Meanings and the vagueness of their embodiments. *In*: THELLEFSEN, T.; SØRENSEN, B.; COBLEY, P. (eds.). *From First to Third via Cybersemiotics - A Festschrift Honoring Professor Søren Brier on the Occasion of his 60th Birthday*. Copenhagen: SL Forlagene, 2011. p. 247–282.

____. *Introdução à semiótica*. 1. ed. São Paulo: Paulus, 2017.

PEIRCE, C. S. *The Collected Papers of Charles Sanders Peirce*: HARTSHORNE, C.; WEISS, P. (Eds. v. 1-6), BURKS, A. W. (Ed. v. 7-8). Cambridge, MA: Harvard University Press, 1931–1958.

____. *Papers*. Cambridge, MA: The Houghton Library of University Microproduction: The Houghton Library of University Microproduction, 1963–66. Microfilm edition. 30 reels.

____. *The New Elements of Mathematics*: Mathematical Philosophy, volume iv. The Hague: Mouton, 1976.

____. *The Essential Peirce*: Selected Philosophical Writings Volume 2 (1893–1913). Bloomington, IN: Indiana University Press, 1998.

PUTNAM, H. Trial and error predicates and the solution to a problem of Mostowski. *The Journal of Symbolic Logic*, v. 30, n. 1, p. 49–57, 1965.

QUARESMA, A. Inteligências artificiais e os limites da computação. *Paakat: Revista de Tecnología y Sociedad*, v. 8, n. 15, 2018.

ROBERTS, D. D. *The Existential Graphs of Charles S. Peirce*. The Hage: Mouton, 1973.

____. The existential graphs. *Computers & Mathematics with Applications*, v. 23, n. 6, p. 639–663, 1992. ISSN 0898-1221.

ROGERS, H. *Theory of Recursive Functions and Effective Computability*. Cambridge, MA: MIT Press, 1967.

SANTAELLA, L. *Estética de Platão a Peirce*. São Paulo: Experimento, 1994.

____. *A teoria geral dos signos*. São Paulo: Pioneira, 2000.

____. *O método anticartesiano de C. S. Peirce*. São Paulo: Editora UNESP, 2004.

____. O amplo conceito peirciano da mente: sua relevância para a biologia, inteligência artificial e cognição. *In*: FERREIRA, A.; GONZALEZ, M. E. Q.; COELHO, J. G. (eds.). *Encontro com as ciências cognitivas*. São Paulo: Cultura Acadêmica Editora, 2005. v. 4, p. 167–180.

____. *Percepção: fenomenologia, ecologia, semiótica*. São Paulo: Cengage Learning, 2012.

SCHURZ, G. Patterns of abduction. *Synthese*, Springer Nature, v. 164, n. 2, p. 201–234, aug 2008.

SEARLE, J. *Minds, Brains and Science*. Cambridge, MA: Harvard University Press, 1984.

SHORT, T. L. Interpreting Peirce's interpretant: a response to Lalor, Liszka, and Meyers. *Transactions of the Charles S. Peirce Society*, v. 32, n. 4, p. 488–541, 1996.

SOARE, R. I. *Recursively Enumerable Sets and Degrees: A Study of Computable Functions and Computably Generated Sets*. New York, NY: Springer-Verlag, 1987. (Perspectives in Mathematical Logic).

SOUZA, C. S. de. *The Semiotic Engineering of Human-Computer Interaction*. Cambridge, MA: MIT Press, 2005.

____. Semiotics. *In*: SOEGAARD, M.; DAM, R. F. (eds.). *The Encyclopedia of Human-Computer Interaction*. 2. ed. Aarhus: The Interaction Design Foundation, 2013. Disponível em: <http://www.interaction-design.org/encyclopedia/semiotics_and_human-computer_interaction.html>.

SPREADSHEET. *In*: WIKIPEDIA. Wikimedia, 2019. Disponível em: <https://en.wikipedia.org/wiki/Spreadsheet>. Acesso em: 21.5.2019.

STANDISH GROUP. *CHAOS Manifesto 2013*. 2013. Disponível em: <https://www.scribd.com/document/198550543/Chaos-Manifesto-2013>. Acesso em: 7.3.2014.

SUCHAN, J. *et al.* Visual explanation by high-level abduction: on answer-set programming driven reasoning about moving objects. *In*: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. New Orleans, LA: AAAI, 2018. p. 1965–1972. Disponível em: <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17303>. Acesso em: 7.9.2018.

TANAKA-ISHII, K. *Semiotics of Programming*. Cambridge, MA: Cambridge University Press, 2010.

TURING, A. M. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, v. 2, n. 42, p. 230–265, 1936.

____. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, s2-45, n. 1, p. 161–228, 1939.

____. Computing machinery and intelligence. *Mind*, v. 59, n. 236, p. 433–460, 1950.

UEXKÜLL, J. von. The theory of meaning. *Semiotica*, v. 42, n. 1, p. 25–82, 1982.

VON NEUMANN, J. First draft of a report on the EDVAC. *Annals of the History of Computing, IEEE*, v. 15, p. 27–75, 1993.

WHITEHEAD, A. N.; RUSSELL, B. *Principia Mathematica*: 3 vols. Cambridge: The University Press, 1912–1913.

WOLFRAM RESEARCH. *WolframAlpha*. 2019. Disponível em: <https://www.wolframalpha.com/>.

YAMAMOTO, K.; ONISHI, T.; TSURUOKA, Y. Hierarchical reinforcement learning with abductive planning. *CoRR – Computing Research Repository*, 2018. Disponível em: <https://arxiv.org/abs/1806.10792>. Acesso em: 7.9.2018.

APÊNDICE A – Lista de símbolos matemáticos

Elencamos aqui os símbolos matemáticos menos usuais utilizados nas equações e proposições lógicas formais.

Símbolos lógicos:

$\neg$: negação lógica

$\vee$: disjunção lógica (“ou”)

$\wedge$: conjunção lógica (“e”)

$\Rightarrow$: implicação lógica

$\forall$: quantificador universal (“todos”)

$\exists$: quantificador existencial (“existe”)

$\nexists$: negação do quantificador existencial (“não existe”)

$|$: tal que

$\stackrel{?}{=}$: a igualdade é uma conjetura

$\vDash$: implicação semântica

Relações entre grandezas:

$=$: igual a

$>$: maior que

$\geq$: maior ou igual a

$<$: menor que

$\leq$: menor ou igual a

Conjuntos:

$\varnothing$: conjunto vazio

∞ : infinito

$\in$: pertencimento a um conjunto

$\bigcup$: união de conjuntos

$\subset$: está contido

$\subseteq$: está contido, podendo ser igual a

Símbolos aritméticos:

$1, 2, 3, \ldots$: os números um, dois, três, ...

$+$: soma

$-$: subtração

$\times$: multiplicação (pode ser omitido)

$.$: multiplicação (pode ser omitido)

$\div$: divisão

$\frac{a}{b}$: divisão de a por b

a^b : exponenciação, a elevado à b-ésima potência

APÊNDICE B – A linguagem de programação utilizada por $\mathcal{A}_{lea}$

O artigo de Gazoni (2016) apresenta um programa que aqui foi batizado $\mathcal{A}_{lea}$. É um programa que gera programas-produto escritos em uma linguagem construída especialmente para ele, para dar ao autor total controle sobre ela. A linguagem se baseia numa pilha FIFO (*First In, First Out*), ou "o primeiro que entra é o primeiro que sai". Seus comandos são:[1]

- <QUALQUER NÚMERO>: Coloca o número no topo da pilha.
- POP: Extrai (e apaga) o número que está no topo da pilha.
- PRT: Mostra (imprime) o que está no topo da pilha.
- LBL <NÚMERO>: Cria um rótulo, de número <NÚMERO>, que aponta para a instrução seguinte.
- GTO <NÚMERO>: Continua a execução a partir da instrução definida pelo rótulo <NÚMERO>, se existir.
- IF0GTO <NÚMERO>: Se o que está no topo da pilha for igual a 0 (zero), salta para a instrução de rótulo <NÚMERO>; senão, continua a execução.
- IFN0GTO <NÚMERO>: Se o que está no topo da pilha for diferente de 0 (zero), salta para a instrução de rótulo <NÚMERO>; senão, continua a execução.
- LET <NÃO NÚMERO>: Armazena o que está no topo da pilha numa variável de nome <NÃO NÚMERO>.
- ALET <NÃO NÚMERO> <NÚMERO>: Armazena o que está no topo da pilha na <NÚMERO>-ésima posição do array de nome <NÃO NÚMERO>.
- INP: Lê dados de entrada e coloca no topo da pilha; os dados de entrada são localizados no final do programa.
- + - . / %: Realiza a operação aritmética (<OP>) entre o que está no topo da pilha (<TOP>) e o número precedente, que está "abaixo" dele na pilha (<PREC>), na ordem <PREC> <OP> <TOP>, e coloca o resultado no topo da pilha (tendo antes extraído <TOP> e <PREC>). '.' é o sinal de multiplicação.
- <NÃO NÚMERO>: Coloca o conteúdo da variável <NÃO NÚMERO> no topo da pilha, se a variável existe; se não existe e <NÃO NÚMERO> é o nome de um array, substitui o topo da pilha com o valor do item do array cujo número está no topo da pilha.

[1] Texto entre "<>" deve ser entendido pelo conteúdo; assim, "<QUALQUER NÚMERO>" deve ser entendido como um número qualquer, por exemplo, 32. "<NÃO NÚMERO>" pode ser qualquer conjunto de símbolos que não pode ser entendido como um número. Esses não números são escritos, preferencialmente, em letras minúsculas.

O funcionamento é bastante simples, e deve ser familiar aos habituados à notação polonesa reversa utilizada em algumas calculadoras financeiras. Alguns exemplos de utilização estão na Tabela 16.

Tabela 16 – Exemplos de programas na linguagem utilizada por $\mathcal{A}_{lea}$

Programa	Resultado	Conteúdo da pilha no final da execução (topo à esquerda)
`13`		$\{13\}$
`13 PRT`	13	$\{13\}$
`13 12 PRT`	12	$\{12, 13\}$
`13 12 + PRT`	25	$\{25\}$
`13 INP PRT INP PRT 12 5`	5 12	$\{12, 5, 13\}$
`INP INP + PRT 13 12`	25	$\{25\}$
`INP INP + PRT 7 2`	9	$\{9\}$
`LET a INP INP + PRT 7 2`	9	$\{9\}$
`INP INP - PRT 2 7`	5	$\{5\}$

É uma linguagem interpretada, implementada em C++. O código fonte do programa utilizado no artigo está disponível em http://semiotic.com.br/Peirce/program.tar.gz.

APÊNDICE C – A lista de hábitos $\mathbb{H}_I$

Os procedimentos a seguir fazem parte do programa que infere o procedimento de eliminação de redundâncias de um programa gerado por $\mathcal{A}_{lea}$ a partir da observação do próprio programa. Os procedimentos foram colocados num formato semelhante aos de comandos de programação para facilitar a compreensão dos algoritmos mostrados.

Nessa lista os textos entre "<>" são substituídos por expressões adequadas. Textos no formato "<NOME ...>" são substituídos por textos com a seguinte convenção:

- Nomes de locais de memória iniciam com a letra "m".
- Nomes de listas iniciam com a letra "l".
- Nomes de procedimentos iniciam com a letra "p"; procedimentos são escritos entre chaves ({}).
- Rótulos iniciam com a letra "r".
- O texto tenta explicar a que se refere usando o padrão *camel case*. Assim, a lista de telefones chamar-se-ia, por exemplo, "lTelefones". Um local de memória contendo a quantidade de itens na lista de telefones teria o nome, por exemplo, "mQuantidadeTelefones".

Os textos no formato "<VALOR ...>" podem ser substituídos nomes — caso em que será levado em conta o valor associado ao nome (se for um local de memória, o conteúdo do local; se for um procedimento, o resultado da execução do procedimento; uma lista, o conteúdo da lista) —, por valores numéricos ou por expressões que retornem valores.

Algumas outras convenções: os comandos podem ser separados por ponto-e-vírgula para clareza. O conteúdo das listas é escrito entre chaves (por exemplo, "{1, 2, 3}"). Os textos no formato "<PROCEDIMENTO ...>" podem ser substituídos por um ou mais comandos. Textos literais são escritos entre aspas (duplas ou simples, conforme a conveniência); portanto, o texto "'TEXTO'" corresponde à palavra "TEXTO" sem as aspas.

Os comandos numerados de 1 a 12 são uma lista mínima que dão ao programa poder de computação igual ao de qualquer máquina de Turing. Os comandos 13 a 17 são os comandos sugeridos como os que capacitam o agente a acessar parte da iconicidade dos programas gerados por $\mathcal{A}_{lea}$. Os comandos 18 e 19 poderiam ser gerados por algoritmos que utilizam os comandos de 1 a 12; estão na lista por comodidade. Do mesmo modo, o comando 15 poderia ser gerado por algoritmos que utilizam os comandos de 1 a 13, e o comando 16, pelos comandos 1 a 12, mais o comando 14.

1. **<NOME> ← <VALOR>**

O que: Guarda no local de memória de nome <NOME> o valor <VALOR>.
Relação: O valor v_1 foi guardado em memória.
Retorna: Nada.
Ex.: `mTeste ← 5; mTexto ← "UM TEXTO"`

—

Guarda o valor 5 no local de memória com o estranho nome `mTeste`, e o texto "`UM TEXTO`", sem as aspas, no local de memório `mTexto`.

2. **<NOME DE LISTA>[<VALOR POSICAO>)]**

O que: Retorna o valor do item que está na posição <VALOR POSICAO> da lista <NOME DE LISTA>. A primeira posição é a posição 0.
Relação: A posição v_1 de v_2 contém v_3.
Retorna: v_3.
Ex.: `lLista ← {1, 2, 3}; mPrimeiro ← lLista[0]`

—

Guarda o valor do primeiro item da lista `lLista` (no caso, 1) no local de memória `mPrimeiro`.

3. **pImprima(<VALOR1>, <VALOR2>, ...)**

O que: Imprime <VALOR1>, <VALOR2>, ... sem separação entre eles. Texto entre aspas é impresso literalmente.
Relação: Nenhuma.
Retorna: Nada.
Ex.: `mTeste ← 5; pImprima("mTeste contém ", mTeste)`

—

Imprime o texto "`mTeste contém 5`".

4. **pParaCadaItemDe(<NOME DE LISTA> pEm <NOME MEMORIA>) {<PROCEDIMENTO>}**

O que: Começando do primeiro, pega cada item da lista <NOME DE LISTA>, coloca o valor dele em <NOME MEMORIA> e executa o <PROCEDIMENTO>.
Relação: v_1 foi executado para cada item de v_2.
Retorna: Nada.
Ex.:
```
lLista ← {1, 4, 2}
pParaCadaItemDe(lLista pEm mItem)
{ pImprima(mItem, ", ") }
```

—

Imprime "`1,4,2`".

5. **<VALOR1> <OPERADOR ARITMETICO> <VALOR2>**

O que: Realiza a operação aritmética definida por <OPERADOR ARITMETICO>, que pode ser "+", "-", "." (multiplicação), "/" ou "^" (exponenciação) entre os valores <VALOR1> e <VALOR2> e retorna o resultado. Operadores aritméticos têm precedência sobre os demais operadores, mas não sobre parênteses.

Relação: v_1 é resultado da operação v_2 de v_3 por v_4.

Retorna: v_1.

Ex.: `mXis ← 5; mY ← mXis . 4`

—

Armazena o valor `20` em `mY`.

6. **<VALOR1> <OPERADOR RELACIONAL> <VALOR2>**

O que: Realiza a comparação definida por <OPERADOR RELACIONAL>, que pode ser "=", ">", "<", "≥", "≤" ou "≠" entre os valores <VALOR1> e <VALOR2> e retorna o resultado, que pode ser TRUE, se a comparação for verdadeira, ou FALSE, caso contrário. Operadores relacionais têm precedência sobre operadores lógicos, mas não sobre operadores aritméticos.

Relação: v_1 é resultado da operação v_2 entre v_3 e v_4.

Retorna: v_1.

Ex.: `mXis ← 5; mY ← mXis . 4; pImprima(mXis ≠ mY)`

—

Imprime "TRUE".

7. **<VALOR LOGICO1> <OPERADOR LOGICO> <VALOR LOGICO2>**

O que: Realiza a operação definida por <OPERADOR LOGICO>, que pode ser "AND", "OR" ou "NOT" entre os valores lógicos (que podem ser TRUE ou FALSE) e retornam o resultado, que também pode ser TRUE ou FALSE.

Relação: v_1 é resultado da operação v_2 entre v_3 e v_4.

Retorna: v_1.

Ex.: `mXis ← 5; mY ← mXis . 4; pImprima(mXis ≠ mY AND 1 = 0)`

—

Imprime "FALSE".

8. **pSe(<VALOR LOGICO>) {<PROCEDIMENTO 1>} pSenao {<PROCEDIMENTO 2>}**

O que: Se <VALOR LOGICO> for TRUE executa <PROCEDIMENTO 1>. Senão, executa <PROCEDIMENTO 2>. O trecho "pSenao <PROCEDIMENTO 2>" é opcional.

Relação: v_1 levou à execução de v_2 e não de v_3.

Retorna: Nada.

Ex.: `pSe(1 = 0) {pImprima("JAMAIS")} pSenao {pImprima("SEMPRE")}`

—

Imprime "SEMPRE".

9. **pLabel <NOME ROTULO>**
   ⋮
   pGoto <NOME ROTULO>

O que: pLabel <NOME ROTULO> cria um rótulo no fluxo do programa; pGoto <NOME ROTULO> transfere a execução do programa para a instrução seguinte à pLabel <NOME ROTULO>.
Relação: Nenhuma.
Retorna: Nada.
Ex.:
```
mContador ← 5;
pLabel rInicio;
pImprima(mContador, “, ”); mContador ← mContador - 1;
pSe(mContador > 0) {pGoto rInicio} pSenao {pImprima(“Acabou!”)}
```
—
Imprime “5, 4, 3, 2, 1, Acabou!”.

10. **pIncluiEmLista(<NOME LISTA> pValor <VALOR> pNa <VALOR POSICAO>)**

O que: Inclui o valor <VALOR> na posição <VALOR POSICAO> da lista <NOME LISTA>.
Relação: v_1 é o resultado da inclusão de v_2 na posição v_3 de v_4.
Retorna: Nada.
Ex.:
```
lLista ← {1, 2, 3};
pIncluiEmLista(lLista pValor 15 pNa 1);
pImprima(lLista)
```
—
Imprime “{1, 15, 2, 3}”.

11. **pExcluiDeLista(<NOME LISTA>, <VALOR POSICAO>)**

O que: Exclui o item da posição <VALOR POSICAO> da lista <NOME LISTA>.
Relação: v_1 é o resultado da exclusão do item na posição v_2 de v_3.
Retorna: Nada.
Ex.:
```
lLista ← {1, 2, 3};
pExcluiDeLista(lLista, 1);
pImprima(lLista)
```
—
Imprime “{1, 3}”.

12. **pComprimentoDeLista(<NOME LISTA>)**

O que: Retorna o número de itens da lista <NOME LISTA>.
Relação: v_1 é o número de itens de v_2.
Retorna: v_1.
Ex.: `lLista ← {1, 2, 3};`
`pImprima(pComprimentoDeLista(lLista))`

—

Imprime "3". Observemos que
`pIncluiEmLista(lLista pValor <VALOR> pNa pComprimentoDeLista(lLista))`
vai sempre incluir <VALOR> no final da lista `lLista`.

13. **pSeparaLetras(<VALOR TEXTO>)**

O que: Retorna uma lista contendo em cada elemento uma letra do texto <VALOR TEXTO>.
Relação: A lista v_1 contém todas as letras de v_2.
Retorna: v_1.
Ex.: `pImprima(pSeparaLetras("Este texto é um teste"))`

—

Imprime "`{E, s, t, e, ␣, t, e, x, t, o, ␣, é, ␣, u, m, ␣, t, e, s, t, e}`". Observe que as posições marcadas com ␣ representam os espaços em branco que seriam invisíveis de outra forma.

14. **pConcatenaLetras(<NOME LISTA LETRAS>)**

O que: O oposto de `pSeparaLetras(<VALOR TEXTO>)`: retorna um texto formado com as letras da lista <NOME LISTA LETRAS>, na ordem que aparecem.
Relação: A lista v_1 concatenada forma o texto v_2.
Retorna: v_2.
Ex.: `lListaLetras ← {"E", "s", "t", "e", " ", "t", "e", "x", "t", "o", " ", "é", " ", "u", "m", " ", "t", "e", "s", "t", "e"};`
`pImprima(pConcatenaLetras(lListaLetras))`

—

Imprime "`Este texto é um teste`".

15. **pSeparaPalavras(<VALOR TEXTO>)**

O que: Retorna uma lista contendo em cada elemento uma palavra do texto <VALOR TEXTO>.
Relação: A lista v_1 contém todas as palavras de v_2.
Retorna: v_1.
Ex.: `pImprima(pSeparaPalavras("Este texto é um teste"))`

—

Imprime "`{Este, texto, é, um, teste}`".

16. **pConcatenaPalavras(<NOME LISTA PALAVRAS>)**

O que: O oposto de pSeparaPalavras(<VALOR TEXTO>): retorna um texto formado com as palavras da lista <NOME LISTA PALAVRAS>, na ordem que aparecem, separadas por um espaço em branco.

Relação: A lista v_1 concatenada, com itens separados por espaços em branco, forma o texto v_2.

Retorna: v_2.

Ex.: `lListaPalavras ← {“Este”, “texto”, “é”, “um”, “teste”};`
`pImprima(pConcatenaPalavras(lListaPalavras))`

—

Imprime “`Este texto é um teste`”.

17. **pExecuta(<VALOR TEXTO>)**

O que: Executa o texto contido em <VALOR TEXTO> como um programa, e retorna uma lista contendo os valores impressos pelo programa, um item para cada vez que o comando de impressão é executado no programa. Por conveniência, admitiremos que o texto em questão pode conter tanto programas escritos com os comandos da presente lista, como programas escritos com comandos da lista de comandos reconhecida pelo programa $\mathcal{A}_{lea}$, descrita no Apêndice B.

Relação: A execução de v_1 resulta em v_2.

Retorna: v_2.

Ex.: `lResultado ← pExecuta(“mWords ← pSeparaPalavras(‘Este texto é um teste’); pImprima(mWords[3]);”);`

—

Armazena `{um}` em `lResultado`. Um programa na linguagem de $\mathcal{A}_{lea}$:
`pImprima(pExecuta(“INP INP + PRT 2 7”))`

—

Imprime `{9}`.

18. **pTrocaItens(<NOME LISTA> pTroca <VALOR POSICAO1> pPor <VALOR POSICAO2>)**

O que: Troca de lugar os itens das posições <VALOR POSICAO1> e <VALOR POSICAO2> na lista <NOME LISTA>.

Relação: A troca dos itens de posição v_1 e v_2 em v_3 resulta em v_4.

Retorna: Nada.

Ex.:
```
lLista ← {1, 2, 3};
pTrocaItens(lLista pTroca 3 pPor 1)
pImprima(lLista)
```

—

Imprime "{3, 2, 1}". Observemos que este procedimento poderia ser implementado como um algoritmo que utiliza os procedimentos anteriores, recebendo como entrada lNomeLista, mPosicaoA e mPosicaoB:

```
mValorA ← lNomeLista[mPosicaoA];
pIncluiEmLista(lNomeLista pValor lNomelista[mPosicaoB] pNa mPosicaoA);
pExcluiDeLista(lNomeLista, mPosicaoA + 1)
pIncluiEmLista(lNomeLista pValor mValorA pNa mPosicaoB);
pExcluiDeLista(lNomeLista, mPosicaoB + 1)
```

19. **pCombinacoesLista(<NOME LISTA>)**

O que: Retorna uma lista de listas, contendo todas as combinações possíveis de itens da lista <NOME LISTA> nos itens; contempla, a lista vazia e a própria lista original.

Relação: As listas v_1 constituem todas as combinações possíveis dos itens de v_2, sem repetição de itens.

Retorna: v_1.

Ex.:
```
lLista ← {1, 2, 3};
pImprima(pCombinacoesLista(lLista))
```

—

Imprime "{ {}, {1}, {2}, {3}, {1, 2}, {2, 1}, {1, 3}, {3, 1}, {2, 3}, {3, 2}, {1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1} }".

APÊNDICE D – Cálculo da probabilidade do procedimento $\mathcal{T}_{r1}$ ser sorteado

Apresentamos aqui o cálculo da probabilidade de um sorteio de um procedimento qualquer. Isso, claro, depende do procedimento de sorteio utilizado, além da complexidade e tamanho do procedimento sorteado. Vamos tomar como exemplo de procedimento sorteado o algoritmo $\mathcal{T}_{r1}$ mostrado na Seção 6.3, apresentado a seguir. Ele utiliza procedimentos presentes na lista $\mathbb{H}_I$ (detalhados no Apêndice C) e serve para eliminar os passos redundantes de um procedimento, que inicialmente está armazenado em `mPrograma`:

```
1. mMenorComprimento ← pComprimentoDeLista(pSeparaPalavras(mPrograma))
2. mMenorPrograma ← mPrograma
3. mResultado ← pExecuta(mPrograma)
4. lTodasCombinacoes ← pCombinacoesLista(pSeparaPalavras(mPrograma))
5. pParaCadaItemDe(lTodasCombinacoes pEm lCombinacaoTestada) {
6.     mComprimentoTestado ← pComprimentoDeLista(lCombinacaoTestada)
7.     pSe(mMenorComprimento > mComprimentoTestado) {
8.         mProgramaTestado ← pConcatenaPalavras(lCombinacaoTestada)
9.         pSe(mResultado = pExecuta(mProgramaTestado)) {
10.            mMenorPrograma ← mProgramaTestado
11.            mMenorComprimento ← mComprimentoTestado
           }
       }
   }
12. pImprima(mMenorPrograma)
```

Vamos calcular a probabilidade de um programa construir o algoritmo $\mathcal{T}_{r1}$ a partir de tentativas aleatórios, supondo os símbolos equiprováveis e que o programa que sorteia trata adequadamente os integrantes dos comandos sorteados.[1] Além dos procedimentos, $\mathcal{T}_{r1}$ lida com outros oito símbolos mostrado na Tabela 17. Esses oito símbolos, mais dezessete[2] comandos, três operadores lógicos, seis operadores relacionais e cinco operadores aritméticos resultam em $8+17+3+6+5 = 39$ símbolos (31 dos quais são comandos). O procedimento todo tem seis instruções; a última instrução, `pParaCadaItemDe(lTodasCombinacoes pEm lCombinacaoTestada) {}`, contém

[1] Ou seja, se o comando sorteado é a operação aritmética +, o programa sorteador sabe que tem de sortear dois operadores para complementar o comando.

[2] Consideramos `pLabel` e `pGoto` como dois comandos.

Tabela 17 – Símbolos em $\mathcal{T}_{r1}$ que não pertencem a $\mathbb{H}_I$

`mPrograma`	`mMenorComprimento`
`mMenorPrograma`	`mResultado`
`lTodasCombinacoes`	`lCombinacaoTestada`
`mComprimentoTestado`	`mProgramaTestado`

duas instruções, das quais uma, `pSe(mMenorComprimento > mComprimentoTestado) {}` contém duas instruções das quais uma, `pSe(mResultado = pExecuta(mProgramaTestado) {}`, contém mais instruções.

Vamos calcular a probabilidade de se obter o programa acima supondo a utilização de um procedimento de sorteio que siga as seguintes premissas, que chamaremos P_{Abd}:

1. Os procedimentos sorteados possuem de uma a seis instruções.
2. O sorteio é contextualizado: sorteia-se, para a posição de cada instrução, um dos 9 comandos que não retorna valor, e em seguida sorteiam-se os componentes do comando:
   a) Se o comando utiliza valores como parâmetros, para cada valor utilizado é sorteado um símbolo que pode retornar um valor.
      i. Se o comando for `pSe`, o único comando que aceita um valor lógico como argumento, os valores são sorteados entre os 9 símbolos que retornam valores lógicos. Desses, 3 aceitam somente argumentos que são valores lógicos. Os valores desses são sorteados entre os 9, recursivamente, até que todos os valores estejam definidos.
      ii. Se o comando for o comando de atribuição (←), então o lado esquerdo (que tem de ser um local de memória) é sorteado entre os 8 locais de memória possíveis. O mesmo vale para o componente `pEm` do comando `pParaCadaItemDe (...) {...}`.
      iii. Se for qualquer outro comando, os valores sorteados serão sorteados entre os 13 + 8 = 21 símbolos remanescentes que retornam valores.[3]
   b) Se o comando utiliza procedimentos, o procedimento é sorteado como um programa, ou seja, por este mesmo algoritmo.
   c) Se o comando for o comando `pExecuta`, é sorteado um dos 8 locais de memória, ou, com a mesma probabilidade de cada um deles, um outro programa que é transformado em texto para ser usado como parâmetro. Portanto, a probabilidade de se sortear um dos locais de memória ou um programa é $\frac{1}{9}$.
   d) Se o comando for o comando `pLabel` ou o comando `pGoto`, o rótulo também é sorteado entre 6 rótulos possíveis.

[3] São 31 comandos, dos quais 9 não retornam valor e 9 retornam valores lógicos, restando 31 – 9 – 9 = 13 comandos que retornam valores não lógicos, mais 8 locais de memória totalizando 21 símbolos.

Calculemos, então, a probabilidade de sortear o conteúdo da linha 11:

$$l_{11} = \texttt{mMenorComprimento} \leftarrow \texttt{mComprimentoTestado}$$

Representando a probabilidade do evento x como $P(x)$, temos:

$$\begin{aligned}
P(\leftarrow) &= \frac{1}{9} \\
P(\texttt{mMenorComprimento}) &= \frac{1}{8} \\
P(\texttt{mComprimentoTestado}) &= \frac{1}{21} \\
P(l_{11}) &= P(\leftarrow) \times P(\texttt{mMenorComprimento}) \times P(\texttt{mComprimentoTestado}) \\
&= \frac{1}{9} \times \frac{1}{8} \times \frac{1}{21} \\
&= \frac{1}{1512}
\end{aligned}$$

O mesmo valor pode ser aplicado às linhas 2 e 10:

$$\begin{aligned}
l_2 &= \texttt{mMenorPrograma} \leftarrow \texttt{mPrograma} \\
l_{10} &= \texttt{mMenorPrograma} \leftarrow \texttt{mProgramaTestado} \\
P(l_2) = P(l_{10}) &= P(l_{11}) = \frac{1}{1512}
\end{aligned}$$

As linhas 6 e 8 têm a mesma probabilidade:

$$\begin{aligned}
l_6 &= \texttt{mComprimentoTestado} \leftarrow \texttt{pComprimentoDeLista(lCombinacaoTestada)} \\
l_8 &= \texttt{mProgramaTestado} \leftarrow \texttt{pConcatenaPalavras(lCombinacaoTestada)}
\end{aligned}$$

Tomando l_8:

$$\begin{aligned}
P(\leftarrow) &= \frac{1}{9} \\
P(\texttt{mProgramaTestado}) &= \frac{1}{8} \\
P(\texttt{pConcatenaPalavras(...)}) &= \frac{1}{21} \\
P(\texttt{lCombinacaoTestada}) &= \frac{1}{21} \\
P(l_8) &= P(\leftarrow) \times P(\texttt{mProgramaTestado}) \times \\
&\quad \times P(\texttt{pConcatenaPalavras(...)}) \times P(\texttt{lCombinacaoTestada}) \\
&= \frac{1}{9} \times \frac{1}{8} \times \frac{1}{21} \times \frac{1}{21} \\
&= \frac{1}{31752} = P(l_6)
\end{aligned}$$

Do mesmo modo, as linhas 1 e 4 têm a mesma probabilidade:

l_1 = `mMenorComprimento ← pComprimentoDeLista(pSeparaPalavras(mPrograma))`

l_4 = `lTodasCombinacoes ← pCombinacoesLista(pSeparaPalavras(mPrograma))`

Tomando l_1:

$$P(\leftarrow) = \frac{1}{9}$$
$$P(\texttt{mMenorComprimento}) = \frac{1}{8}$$
$$P(\texttt{pComprimentoDeLista(...)}) = \frac{1}{21}$$
$$P(\texttt{pSeparaPalavras(...)}) = \frac{1}{21}$$
$$P(\texttt{mPrograma}) = \frac{1}{21}$$
$$P(l_1) = \frac{1}{9} \times \frac{1}{8} \times \frac{1}{21} \times \frac{1}{21} \times \frac{1}{21}$$
$$= \frac{1}{666792} = P(l_4)$$

A probabilidade de sortear a linha 3:

l_3 = `mResultado ← pExecuta(mPrograma)`

$$P(\leftarrow) = \frac{1}{9}$$
$$P(\texttt{mResultado}) = \frac{1}{8}$$
$$P(\texttt{pExecuta(...)}) = \frac{1}{21}$$
$$P(\texttt{mPrograma}) = \frac{1}{9}$$

$$P(l_3) = \frac{1}{9} \times \frac{1}{8} \times \frac{1}{21} \times \frac{1}{9}$$
$$= \frac{1}{13608}$$

A probabilidade de sortear a linha 12 é dada por:

$$
\begin{aligned}
l_{12} &= \texttt{pImprima(mMenorPrograma)} \\
P(\texttt{pImprima(...)}) &= \frac{1}{9} \\
P(\texttt{mMenorPrograma}) &= \frac{1}{21}
\end{aligned}
$$

$$
\begin{aligned}
P(l_{12}) &= \frac{1}{9} \times \frac{1}{21} \\
&= \frac{1}{189}
\end{aligned}
$$

Para prosseguir, vamos agora calcular a probabilidade dos comandos que têm procedimentos como argumentos. Essa probabilidade leva em conta a probabilidade de sortear o próprio comando, a probabilidade de o procedimento possuir o número correto de instruções, e a probabilidade de cada instrução em si. Como há comandos com procedimentos dentro de comandos com procedimentos, vamos iniciar do mais interno para o mais externo.

A probabilidade de sortear a linha 9 é dada por:

$$
\begin{aligned}
l_9 &= \texttt{pSe(mResultado = pExecuta(mProgramaTestado)) \{...\}} \\
P(\texttt{pSe(...)}) &= \frac{1}{9} \\
P(=) &= \frac{1}{9} \\
P(\texttt{mResultado}) &= \frac{1}{21} \\
P(\texttt{pExecuta(...)}) &= \frac{1}{21} \\
P(\texttt{mProgramaTestado}) &= \frac{1}{9}
\end{aligned}
$$

Procedimento:

$$
\begin{aligned}
P(\text{2 instruções}) &= \frac{1}{6} \\
P(l_{10} \text{ e } l_{11}) &= P(l_{10}) \times P(l_{11}) \\
&= \frac{1}{1512^2}
\end{aligned}
$$

$$
\begin{aligned}
P(l_9) &= \frac{1}{9} \times \frac{1}{9} \times \frac{1}{21} \times \frac{1}{21} \times \frac{1}{9} \times \frac{1}{6} \times \frac{1}{1512^2} \\
P(l_9) &= 2,2676658 \times 10^{-13}
\end{aligned}
$$

A probabilidade da linha 7:

$$l_7 = \texttt{pSe(mMenorComprimento > mComprimentoTestado) \{...\}}$$
$$P(\texttt{pSe(...)}) = \frac{1}{9}$$
$$P(>) = \frac{1}{9}$$
$$P(\texttt{mMenorComprimento}) = \frac{1}{21}$$
$$P(\texttt{mComprimentoTestado}) = \frac{1}{21}$$

Procedimento:

$$P(2 \text{ instruções}) = \frac{1}{6}$$
$$P(l_8 \text{ e } l_9) = P(l_8) \times P(l_9)$$

$$P(l_7) = \frac{1}{9} \times \frac{1}{9} \times \frac{1}{21} \times \frac{1}{21} \times \frac{1}{6} \times \frac{1}{31752} \times 2,2676658 \times 10^{-13}$$
$$= 3,332215695 \times 10^{-23}$$

A probabilidade da linha 5 é semelhante:

$$l_5 = \texttt{pParaCadaItemDe(lTodasCombinacoes pEm lCombinacaoTestada) \{...\}}$$
$$P(\texttt{pParaCadaItemDe(...)}) = \frac{1}{9}$$
$$P(\texttt{lTodasCombinacoes}) = \frac{1}{21}$$
$$P(\texttt{lCombinacaoTestada}) = \frac{1}{8}$$

Procedimento:

$$P(2 \text{ instruções}) = \frac{1}{6}$$
$$P(l_6 \text{ e } l_7) = P(l_6) \times P(l_7)$$

$$P(l_5) = \frac{1}{9} \times \frac{1}{21} \times \frac{1}{8} \times \frac{1}{6} \times \frac{1}{31752} \times 3,332215695 \times 10^{-23}$$
$$= 1,15680186 \times 10^{-31}$$

Para calcular a probabilidade de todo o programa basta utilizar os valores das probabilidades de cada linha, multiplicada pela probabilidade de serem seis:

$$P(programa) = P(6 \text{ instruções}) \times P(l_1) \times P(l_2) \times P(l_3) \times P(l_4) \times P(l_5) \times P(l_{12})$$
$$= \frac{1}{6} \times \frac{1}{666792} \times \frac{1}{1512} \times \frac{1}{13608} \times \frac{1}{666792} \times 1,15680186 \times 10^{-31} \times \frac{1}{189}$$
$$= 1,115113492 \times 10^{-53}$$

Os cálculos acima poderiam ser mais precisos por inúmeras razões; por exemplo, não levam em conta que a ordem de alguns comandos poderia ter sido alterada, bem com a ordem de alguns operadores, especialmente nas comparações lógicas, sem alterar o resultado final. Não obstante, as premissas para o sorteio, que é o que efetivamente determina as probabilidades, foram intencionalmente construídas para resultarem uma probabilidade alta. Mesmo assim, podem também ser melhoradas em outros aspectos:

- Alguns valores utilizados como argumentos nos comandos são de tipos bem definidos: números, textos, listas. Assim, ao sortear um símbolo para ocupar o lugar de um valor num comando, este deveria ser sorteado entre os que retornam o tipo adequado.
- Sendo os comandos sorteados na ordem em que serão executados, pode-se não permitir o uso como valor de nenhum local de memória cujo valor não tenha sido previamente determinado.

Essas circunstâncias, por capazes que sejam de melhorar o desempenho do sorteio, não o tornariam viável. Tomando a máquina prototípica (determinada no preâmbulo do Capítulo 4, sem utilização do processamento paralelo) como aproximação, que é capaz de sortear e testar aproximadamente 3.500 programas (escritos na linguagem de $\mathcal{A}_{lea}$ com, em média, 6 instruções por programa) por segundo, o tempo médio de processamento para chegar à solução do problema[4] é dado por:

$$\begin{aligned} P(programa) &= 1,115113492 \times 10^{-53} \\ T &= \frac{1}{P(programa) \times 2 \times 3500} \\ &= 1,28 \times 10^{49} s = 4,06 \times 10^{41} anos \end{aligned}$$

Ou seja, uma quantidade de tempo imensamente superior à idade suposta do universo.

Adicionalmente, é possível elencar um importante conjunto de variáveis que devem ser determinadas para qualquer sorteio desse tipo:

- Probabilidade relativa: todos os símbolos são equiprováveis. Pode-se alterar o peso de alguns símbolos, o que pode trazer algumas vantagens. No caso de haver necessidade de sortear um procedimento como argumento para o comando `pExecuta(...)`, pode-se também determinar probabilidades relativas diferentes das do programa original — por exemplo, nesse caso será muito importante que o programa sorteado apresente ao menos um comando `pImprima(...)`.

[4] Lembremos que o tempo médio de processamento é metade do tempo de processamento necessário para calcular todas as alternativas.

- Sorteio de números e textos: As premissas não contemplam o sorteio de números ou textos nas posições reservadas aos valores dos comandos. Pode haver casos em que isso é importante, o que em compensação vai diminuir consideravelmente a probabilidade de qualquer combinação de símbolos.
- Número de comandos por procedimento: as premissas preconizam 6 porque esse é o número de comandos da solução conhecida. Trata-se de um parâmetro importante que determina a complexidade das soluções possíveis, quanto menor, mais simples o programa sorteado, e maior a probabilidade de se obtê-lo.
- Número de diferentes rótulos possíveis: o número de linhas também determina o número de diferentes rótulos possíveis; é possível que a diminuição do número de rótulos aumente a probabilidade de escolha de um algoritmo correto.
- Número de diferentes locais de memória: o limite de oito também veio da intenção de um programa ajustado à solução. Quanto maior esse número, menor a probabilidade de encontro do algoritmo correto.

Numa situação em que o procedimento investigativo é conduzido pelo usuário, são variáveis que provavelmente serão determinadas pelo usuário. Na moldura filosófica peirciana, podem ser vistas como hábitos éticos sem causa para o agente — já que não há como o programa compreender as escolhas do usuário —, ou seja, são de origem puramente estética.