SERG – Grupo de Pesquisa em Engenharia Semiótica do Departamento de Informática da PUC-Rio

Série de Notas do EMAPS
Ética e Mediação Algorítmica de Processos Sociais

Reflexões sobre Semiótica, Linguagem, Ontologia e Computação

EMAPS Notas #02

Clarisse Sieckenius de Souza
www.inf.puc-rio.br/~clarisse

Março de 2023

Como citar este documento: de Souza, C. S. (2023) **Reflexões sobre Semiótica, Linguagem, Ontologia e Computação**. *EMAPS-Notas #02*. Rio de Janeiro, RJ - Brasil: SERG, Departamento de Informática, PUC-Rio, 2023. 19 p.
URL: www.hcc.inf.puc-rio.br/EMAPS/userfiles/downloads/Notas-deSouzaFilTec2023.pdf

Semiótica, Linguagem, Ontologia, Computação

Autora: Clarisse Sieckenius de Souza

Março de 2023

Resumo:
Este trabalho é a base para as discussões da Mesa 3 sobre *Linguagem e Ontologia* do I Colóquio de Filosofia da Tecnologia da PUC Rio. O tema do evento foi **Coexistência e Mundo Digital - Diálogos Possíveis**. A Mesa 3 aconteceu no dia 1º de março de 2023.

Defendo a ideia de que há determinadas teorias semióticas, em particular as de Charles Sanders Peirce e a de Umberto Eco, que possibilitam um verdadeiro *giro de caleidoscópio* na lente com que se vê, se pensa e se pratica a computação na atualidade. Acredito que a contribuição da Semiótica para a Computação consiste em: primeiro, oferecer um ferramental teórico-metodológico para tratar com uma mesma ontologia as linguagens naturais e artificiais, bem como seus usuários humanos e não-humanos; e, segundo, resgatar a intencionalidade humana do ostracismo em que teorias abstratas da computação, de ordem matemática e lógica, a colocaram. A primeira contribuição permite que uma "teoria do uso da computação" seja **contígua e contínua** a uma "teoria descritiva formal da computação." A segunda contribuição realça o caráter **expressivo** da computação, propondo concretamente que qualquer artefato de software comunica uma "mensagem" que, entre outras coisas, expressa crenças, expectativas, conhecimentos, contexto e condições de quem produz o artefato, para quem usa ou examina este mesmo artefato. Uma teoria semiótica da computação (como todas as teorias não-semióticas que há hoje) não dá conta ***de tudo o que há para estudar*** na descrição formal de processos computacionais, nem na caracterização das condições físicas, psicossociais, políticas e culturais de uso de artefatos computacionais. Porém, ela lança luzes representacionais, expressivas, epistêmicas, éticas e sociopolíticas desafiadoras sobre a computação, ampliando as possibilidades para o debate público acerca dos limites e exigências regulatórias que podem ou devem acompanhar o desenvolvimento e uso de tecnologia digital.

Palavras-Chave:
[Teoria da Computação, Teoria da Linguagem, Pragmática, Semiótica, Ontologia]

1 Introdução

Este trabalho é a base para as discussões da Mesa 3 sobre *Linguagem e Ontologia* do I Colóquio de Filosofia da Tecnologia da PUC Rio. O tema do evento foi **Coexistência e Mundo Digital - Diálogos Possíveis**. A Mesa 3 aconteceu no dia 1º de março de 2023.

Neste artigo, defendo a ideia de que há determinadas teorias semióticas, em particular as de Charles Sanders Peirce e a de Umberto Eco, que possibilitam um verdadeiro *giro de caleidoscópio* na lente com que se vê, se pensa e se pratica a computação na atualidade. Acredito que a contribuição da Semiótica para a Computação consiste em: primeiro, oferecer um ferramental teórico-metodológico para tratar com uma mesma ontologia as linguagens naturais e artificiais, bem como seus usuários humanos e não-humanos; e, segundo, resgatar a intencionalidade humana do ostracismo em que teorias abstratas da computação, de ordem matemática e lógica, a colocaram. A primeira contribuição permite que uma "teoria do uso da computação" seja **contígua e contínua** a uma "teoria descritiva formal da computação." A segunda contribuição realça o caráter **expressivo** da computação, propondo concretamente que qualquer artefato de software comunica uma "mensagem" que, entre outras coisas, expressa crenças, expectativas, conhecimentos, contexto e condições de quem produz o artefato, para quem usa ou examina este mesmo artefato. Uma teoria semiótica da computação (como todas as teorias não-semióticas que há hoje) não dá conta ***de tudo o que há para estudar*** na descrição formal de processos computacionais, nem na caracterização das condições físicas, psicossociais, políticas e culturais de uso de artefatos computacionais. Porém, ela lança luzes representacionais, expressivas, epistêmicas, éticas e sociopolíticas desafiadoras sobre a computação, ampliando as possibilidades para o debate público acerca dos limites e exigências regulatórias que podem ou devem acompanhar o desenvolvimento e uso de tecnologia digital.

2 Uma visão polêmica sobre teorias da computação

Como motivação para um debate, trago algumas ideias de Brian Cantwell-Smith, um híbrido de cientista da computação (com tese de doutorado em reflexão computacional) e filósofo (desde 1990, dedicado ao estudo dos fundamentos filosóficos da computação, da tecnologia e da mente).[1] Como esta é uma proposta de reflexão e não uma dissertação acadêmica, vou passar de forma superficial pelo que diz este autor e todos os demais citados,[2] apenas como uma via de acesso para questões que venho estudando há muitos anos, na fronteira entre a Semiótica, a Linguística, a Inteligência Artificial e a Computação.

Em seu capítulo na coletânea sobre *computacionalismo*, editada por Scheutz (2002), Cantwell-Smith (2002) elenca uma série de construções mentais subjacentes a diferentes teorizações sobre computação. Dentre elas estão as noções de *manipulação formal de símbolos*, *computabilidade efetiva*, *máquinas de estado* e *execução algorítmica*. Segundo ele, nenhuma destas construções atende a três critérios que, a seu ver, são indispensáveis para qualquer teoria fundamental da computação:

[1] Mais informações até a data desta publicação em https://ischool.utoronto.ca/profile/brian-cantwell-smith/.
[2] Ofereço, ao final do texto, referências bibliográficas para os interessados em aprofundar conhecimento.

1. **Adequação Empírica:**
   Uma teoria adequada deve dar conta não apenas de aspectos e elementos computacionais abstratos, mas também de como a computação se manifesta e que efeitos perceptíveis ela causa no mundo em que existimos.

2. **Adequação Conceitual:**
   Uma teoria da computação adequada não pode delegar a outros domínios e campos teóricos a definição de conceitos sobre os quais ela se ergue. Deve ser capaz de (re)enunciá-los com clareza e precisão, em seu próprio corpo teórico. Para Cantwell-Smith, esta delegação é o que acontece com os conceitos de *interpretação*, *representação* e *semântica*, por exemplo, amplamente adotados nas teorias mais ensinadas nos cursos de formação profissional na Informática.

3. **Adequação Cognitiva:**
   Toda teorização é um elaborado processo cognitivo. Portanto, toda teoria da computação deve ser *cognoscível* para o teorizador. Este critério é menos óbvio do que os demais, mas têm consequências fortes. Por exemplo, há teorias *computacionalistas* da mente que defendem que o cérebro humano é um **processador de informações**. Pylyshyn (1980) resume, em um dos textos mais influentes da Ciência Cognitiva, o que dizem, em linhas gerais, estas teorias:[3]

   > A perspectiva computacional da mente se apoia em certas intuições sobre a similaridade fundamental que há entre computação e cognição. [...] elas derivam do fato de que computadores e organismos humanos são ambos sistemas físicos cujo comportamento é corretamente descrito como sendo governado por regras que atuam sobre representações simbólicas. (p. 111)

   Se um teórico da computação invocar *esta teoria cognitiva específica* (onde cognição e computação são processos análogos) para justificar que sua teoria passa no terceiro critério de Cantwell-Smith, ele tem que aceitar que *ele*, teorizador, é uma espécie de computador, que está teorizando sobre outras espécies de computadores. Portanto, para ser cognitivamente adequada, a teoria que este teorizador construir *tem de ser reflexiva*. Caso contrário, ele não terá como explicar que um computador humano *possa conhecer o que é* um computador não humano.

Juntos, estes três critérios representam exigências que, na visão de Cantwell-Smith, nenhuma teoria da computação cumpriu até hoje. Começando pela falta de adequação cognitiva, é fato que a grande maioria dos trabalhos sobre teoria da computação não endereçam as condições cognitivas ou epistêmicas em que opera o teorizador. Entre as implicações associadas a esta omissão flagrante, tenho grande interesse, por exemplo, pelo fato de que o silêncio sobre os sujeitos que teorizam a computação reforça o mito de que a computação é uma matéria puramente formal e abstrata, onde cabem considerações sobre significados subjetivos associados a representações simbólicas ou processos computacionais.

No plano da adequação empírica e conceitual, por sua vez, Cantwell-Smith novamente denuncia uma inconsistência fundamental, a que ele chama de **muralha ontológica**. A muralha isola, de um lado, os símbolos, estados, cálculos, processos, algoritmos — seja lá o que tem sido

[3] Todas as citações em idioma estrangeiro foram livremente traduzidas por mim. A página indicada na referência bibliográfica dá acesso ao que disseram os autores citados.

utilizado para construir uma teoria consistente do que é *computar* — do que, no outro lado, tem sido usado para construir teorias sobre o que tais coisas significam ou realizam no mundo em que elas observavelmente acontecem. De maneira muito simplificada, de um lado da muralha existem entes, relações, fenômenos, etc. que não existem do outro lado. Como relacionar as duas coisas sem um sujeito (humano) que estabelece esta relação em função de pré-concepções sobre si mesmo, sobre o que está fazendo, e sobre a própria natureza das relações que precisam ser estabelecidas para a computação se *concretizar* no plano empírico? Este sujeito pode ser um teorizador, um filósofo, ou outro estudioso, mas sua encarnação mais cotidiana é um programador. O problema da muralha ontológica na adequação empírica e conceitual de teorias da computação é, portanto, que as relações entre a semântica da computação no plano formal abstrato e a semântica da computação no plano material contingente estão totalmente fora de controle teórico. Ao contrário, estão à mercê de programadores, de designers de tecnologia, de cientistas de dados, de pesquisadores de engenharia de software, de pesquisadores de interação humano-computador — a lista é longa.

A conclusão de Cantwell-Smith, mais elaborada em seu livro *The Origin of Objects* (Cantwell-Smith, 1996) e levada às suas últimas consequências para a Inteligência Artificial no livro *The Promise of Artificial Intelligence* (Cantwell-Smith, 2019), é a de que a Computação ainda não é um assunto de que se possa tratar teoricamente de maneira satisfatória (o que não quer dizer que não venha a ser). Também conclui ele que a Inteligência Artificial não passa, no momento, de uma promessa de tecnologias cada vez mais espetaculares. Ela não é, nem define, um objeto teórico consistentemente qualificado como *inteligência*, de qualquer espécie (o que não significa que esta limitação não possa ser superada um dia). A chamada do filósofo ao longo de décadas foi por um trabalho de superação destas barreiras e deficiências teóricas.

3 Semântica e pragmática nos estudos da linguagem humana

Embora muito controverso no meio computacional, o trabalho de Cantwell-Smith me atrai porque a sua visão de fundamentação científica tem um paralelo claro com o que se passou na área de estudos da linguagem, na chamada **virada pragmática** do último quarto do século XX. Não acredito que o paralelo seja uma casualidade e tentarei justificar por quê. Minha narrativa é pautada pelo histórico da Pragmática dentro da Linguística, tal como traçado por Andreas H. Jucker (2012) em seu *Pragmatics in the history of linguistic thought*.

O movimento pragmático na Linguística surgiu como uma rejeição à ideia de que *a linguagem é uma representação, falsa ou verdadeira, de um estado de coisas no (ou em um) mundo*. Esta visão descritiva joga para fora dos **estudos da linguagem** tudo o que diz respeito à comunicação verbal situada, aos seus efeitos sobre interlocutores em atos de comunicação e, sobretudo, seu poder de alterar o estado de coisas no mundo. O estudo destas *outras coisas* não era, naquela época, nenhuma novidade. O problema é que vinha sendo realizado, desde a *Retórica* de Aristóteles, *fora* dos domínios dos estudos sistemáticos da linguagem, tão especialmente caros aos linguistas a partir da revolução estruturalista de Saussure, no início do século XX. (Constantin e de Saussure, 2005)

Os trabalhos de filósofos como Peirce (largamente considerado o fundador da Pragmática), Morris, Carnap e Wittgenstein, por exemplo, abriram caminho para grandes avanços nos estudos da linguagem em situações de uso. Porém, coube a três filósofos darem o salto que faltava

para a Pragmática se estabelecer definitivamente no território da Linguística: Austin e Searle, com a teoria dos Atos de Fala, e Grice com a teoria da Cooperação e das Implicaturas Conversacionais.

Nas décadas de 1960 e 1970, a vigorosa perspectiva transformacional-gerativa de Chomsky (1957, 1959, 1965) se erguia sobre pressupostos acerca de capacidades inatas universais, da superioridade científica de descrições formais abstratas (lógicas e algébricas), da autonomia da sintaxe em relação à semântica em descrições teóricas de linguagens naturais e artificiais, além da exclusão explícita das dimensões empíricas de uso e variabilidade linguística, consideradas não pertencentes ao escopo de uma teoria da linguagem. Na década de 1980, a virada pragmática causou verdadeiras mudanças de paradigma (Jucker, 2012). Uma delas foi a mudança de uma visão introspectiva idealizada da linguagem (Chomsky se baseava na intuição de um "falante ideal" para estabelecer o conjunto de sentenças gramaticais de qualquer língua natural) para uma visão informada pelos dados e análises resultantes da investigação empírica (extensamente influenciada também pela etnometodologia). Outra é a passagem de uma concepção de linguagem centrada na *homogeneidade* para o seu exato oposto: uma concepção de linguagem vivamente interessada na *heterogeneidade*, na variação e na evolução das línguas humanas, situada nos contextos dinâmicos da história e da cultura. Cabe ainda mencionar, no âmbito da Pragmática, as teorias do discurso que, além dos aspectos socioculturais e políticos do uso da linguagem em contextos referenciados ao coletivo, voltaram-se também para a dinâmica da linguagem produzida por um único indivíduo, como no caso do discurso literário ou do discurso psicanalítico.

Diante deste quadro, é oportuno sublinhar o quão descompassadas da evolução dos estudos da linguagem estão, até hoje, as mais populares teorias da computação ensinadas nos cursos superiores de Informática (ver Diverio e Menezes (2009) ou Hopcroft et al. (2014), entre muitos). Tópicos de linguagens formais e autômatos giram em torno da Hierarquia de Chomsky (1959), uma das primeiras publicações do linguista, recém diplomado pelo MIT. O trabalho marca a época de suas visões mais radicais sobre a autonomia da sintaxe em relação à semântica e a *desimportância* das situações empíricas de uso para a construção de uma teoria adequada de qualquer linguagem, natural ou artificial. Porém, ao longo dos anos, as ideias de Chomsky evoluíram muito, especialmente em direção a um alinhamento com teorias cognitivas (Chomsky et al., 2002), o que parece ter sido ignorado na Computação. É verdade que a pesquisa sobre Inteligência Artificial abraçou a maior parte desta evolução. Porém, a fixação teórica em uma etapa ultrapassada pode acarretar, como bem assinalam Allan e Jaszczolt (2012), que a teoria descritiva e a teoria empírica da linguagem sigam em direções totalmente dissociadas uma da outra, o que cria o ambiente perfeito para imensas inconsistências ontológicas.

Nesta perspectiva, as posições de Cantwell-Smith (1996, 2002) parecem, de certa forma, bem menos radicais. Podem ser interpretadas como um chamado intelectual para uma tardia *virada pragmática* na Computação (de Souza, 2018). Também vale uma reflexão o fato de que os rumos atuais da Inteligência Artificial não só abandonaram os modelos baseados em regras (alinhados com teorias linguísticas e cognitivas vigentes no apogeu do gerativismo), como têm defendido vigorosamente uma posição que se pretende "empírica" (não fossem as *representações* do que se entende por *empírico*) no aprendizado de máquina. É curioso observar que, apesar de se oporem aos modelos formais chomskyanos da época gerativista, abordagens de aprendizado profundo (Alpaydin (2014), Aggarwal (2018)) estão em bom alinhamento com eles no que diz respeito à primazia de operações sintáticas sobre representações, cuja semântica não importa para sua

descrição teórica. Este é, de fato, o pivô dos problemas de explicabilidade e interpretabilidade da inteligência artificial nos tempos atuais.

4 Comunicação e ação humanas mediadas por computadores

Seguindo o raciocínio de que *virada pragmática* poderia ser equivalente ao que Cantwell-Smith espera não só da Computação (Cantwell-Smith, 2002), mas também da Inteligência Artificial (Cantwell-Smith, 2019), uma forma de galgar a **muralha ontológica** que separa o significado representacional (interno a um sistema simbólico) do significado empírico (externo, em situação de uso) seria reenquadrar a computação numa perspectiva totalmente diferente, dar um *giro no caleidoscópio*. Winograd e Flores (1987), Floyd (1992) e Andersen e co-autores (1993) estão entre os proponentes de reenquadramentos desta espécie. É interessante observar as datas das publicações de suas respectivas obras em referência e notar que elas são contemporâneas da virada de perspectiva na Linguística. Winograd e Flores abalaram a comunidade de Inteligência Artificial ao propor que a computação é de fato uma nova forma humana de se comunicar. Floyd, por sua vez, propôs que o principal produto da computação, *software*, é uma nova forma humana de *construir realidade*. Já Andersen e co-autores optaram por uma abordagem semiótica da computação, caracterizando computadores como *meio* para a expressão e comunicação de significados humanos, numa linha diferente de Winograd e Flores. Em obra solo, Andersen (1997) descreve extensivamente esta visão, a qual inaugurou o que hoje conhecemos por Semiótica Computacional.

O invariante destes enquadramentos é posicionar *o humano* na computação de tal forma que ela se torna matéria e recurso para a ação comunicativa humana, uma revolução conceitual que *dissolve* a **muralha ontológica**. Como meio de significação, comunicação, expressão, ação e construção de realidade *para pessoas*, a computação se equipara à escrita, por exemplo. O texto escrito é uma das instâncias mais paradigmáticas de espaço para estes mesmos processos: significação, comunicação, expressão, ação e construção de realidade. Estas abordagens, contudo, desafiam a auto-imagem da Computação como uma *ciência exata e formal* e a lançam no meio da confusão, da ambiguidade, da complexidade, da paixão, das falhas e descaminhos da *humanidade*. Se ela é um *meio* para a ação da vontade, conflitos e imaginação humana, sua própria ontologia e, portanto, qualquer teoria que se construa com este pressuposto mudam muito de direção. Além disto, setores *soft* da computação como Interação Humano-Computador e Sistemas Colaborativos, para citar dois que estão representados por participantes do I Colóquio de Filosofia da Tecnologia da PUC-Rio, passam a ter grande importância na construção de teorias computacionais (de Souza, 2018). Estaria a Computação disposta e preparada para transitar no território das Ciências Humanas e Sociais?

Enquanto a resposta é negativa, o que se vê (e justifica posições como as de Cantwell-Smith) é que vêm tomando vulto as *teorias paralelas* sobre o que importaria à Computação, mas de que ela não se ocupa ao tratar de seus fundamentos. Um exemplo bastante interessante é o da Retórica Digital (por exemplo, Eyman (2015), Vee e Brown Jr. (2016), Hess e Davisson (2018) e Jones e Hirsu (2019)) e o dos Estudos Críticos de Código de Programas (Reyman (2018), Brock (2019) e Marino (2020)). Os dois temas estão intimamente interligados e os pesquisadores citados pertencem, em sua ampla maioria, a departamentos e centros de Ciências Humanas de universidades norte americanas e europeias.

Na próxima seção, falarei das avenidas que podem ser abertas na Computação através de um projeto interdisciplinar cooperativo com a Semiótica. Este tema já foi abordado anteriormente em artigo mais longo para divulgação científica (de Souza, 2020).

5 Teorizando semioticamente a computação

A Semiótica contemporânea se organiza em torno de dois grandes pensadores, cuja influência continua sólida e profunda, desde o início do século XX. Na Europa, ela tem origem nos trabalhos de Ferdinand de Saussure (de Saussure, 1995), sob o nome de *Semiologia*. O foco de atenção de Saussure[4] é a descrição da linguagem como um fenômeno psicossocial, em dois planos: o da *langue*, conceitual e sistêmico, e o da *parole*, empírico e sujeito às condições de uso situado dos mecanismos e recursos da *langue*. Já no continente americano, a Semiótica tem origem por volta da mesma época, mas em um contexto bem diferente. O fundador desta vertente é Charles Sanders Peirce (Peirce, 1958), um polímata com contribuições em diversas áreas do conhecimento. Provavelmente por esta razão, ele focava seu trabalho na investigação de como os *signos* e processos *sígnicos* se relacionam com os processos de *descoberta de conhecimento*. Sua inclinação é filosófica, com particular ênfase na epistemologia e na lógica. Embora, com esta descrição, os dois pareçam muito distantes um do outro, o **significado** dos vocábulos e estruturas de uma linguagem é um ponto de interseção muito forte entre eles, pois é indiscutível o seu papel fundamental tanto em estudos linguísticos, quanto em estudos filosóficos mais gerais em torno da relação entre signos, seus objetos e interpretações. Dito isto, há um profusão de abordagens e vertentes nos estudos semióticos, o que torna falar "da Semiótica" uma opção retórica para visar o que é comum e deixar de lado, ao menos por ora, o que é diferente e específico em cada uma.

Fazendo uma escolha teórica pela Semiótica de Peirce (1958) e de Umberto Eco (1976, 1981, 1986), este último considerado um semioticista que reúne em sua obra elementos originários de Peirce e de Saussure, traço a seguir, em linhas muito gerais, um argumento em defesa de uma semiótica computacional específica. Trata-se da *Engenharia Semiótica*, hoje uma teoria da interação humano-computador (de Souza (2005), de Souza e Leitão (2009)) com extensões para a computação centrada no humano (de Souza et al., 2016). Esta teoria vem sendo elaborada e aperfeiçoada desde a década de 1990 pelo nosso grupo de pesquisa no Departamento de Informática da PUC-Rio, o *SERG*.[5]

5.1 Inspirados em Peirce

A Semiótica de Peirce oferece vantagens interessantes para uma articulação com teorias da computação. Em primeiro lugar, ela contribui não apenas para os estudos da linguagem natural (Rellstab (2008), Shapiro (2022)), mas também para o estudo das linguagens formais (Sowa (2008), Farias e Queiroz (2017)). Em segundo lugar, a proposta original de Peirce sobre o raciocínio abdutivo e a semiose (processo indefinidamente longo de geração de signos a partir de signos) contribui tanto para teorias do aprendizado humano (Strand (2013), Nöth (2014)), como também do aprendizado de máquina. (Sowa (2000), Gabbay e Kruse (2000)) Em terceiro lugar, a Semiótica Peirceana não foge das imprecisões, incertezas, contradições, lacunas e

[4] No Brasil, ao contrário de vários outros países, não se usa chamá-lo de ***de*** *Saussure*.
[5] **Semiotic Engineering Research Group**: www.serg.inf.puc-rio.br.

falhas de experiência humana. Ao contrário, ela propõe um novo método de inferência lógica, a abdução (inicialmente conhecida como lógica do raciocínio por hipóteses), que abriga em sua essência o princípio de autocorreção, ajuste, conjectura e ganho de conhecimento através do erro (Santaella, 2004).[6] Este *falibilismo* Peirceano, sua abertura ao erro inevitável, em geral dispara um alarme entre cientistas da computação. Costuma-se dizer jocosamente, nesta comunidade, que o "tratamento de erros na programação consiste em prevenir erros de execução, quando ocorre um erro um erro de execução", o que mostra o quanto se tenta eliminar a possibilidade de erros, compreensivelmente. Assim, uma teoria filosoficamente falibilista da computação, que abraçasse o que, na visão do próprio Peirce (Psillos (2011), Peirce (1958)) é "a mais fraca das lógicas" (a abdutiva), seria sem dúvida uma revolução.[7]

5.2 Inspirados em Eco

A Semiótica de Eco, por sua vez, oferece outras vantagens, quando comparada à de Peirce. Com um discurso mais contemporâneo e acessível para o leitor vindo de outra disciplina, a primeira vantagem de sua teoria é definir de forma muito clara, numa sentença curta que aparece logo nas primeiras páginas de seu *A Theory of Semiotics*, que: A semiótica é em princípio a disciplina que estuda tudo aquilo que pode ser usado para mentir. (Eco, 1976, p. 7) Estas poucas palavras têm implicações muito fortes acerca da linguagem, seus significados e seu uso. Uma afirmação inverídica, caso mais geral da mentira, é um signo falso com relação a seu referente empírico, comunicado por um falante/emissor. O fenômeno semioticamente relavante está em o próprio falante/emissor e (ou) seus ouvintes/receptores poderem interpretar este signo como verdadeiro em um primeiro momento e, mais adiante, serem confrontados evidências em contrário. É o que Peirce chamaria de *a insistência do real*, o fluxo contínuo de outros signos que — ao entrarem em contradição com o signo falso tido por verdadeiro — disparam uma correção, ou reinterpretação condizente (mas ainda não necessariamente correta) do signo inicial. Estes casos de inverdades na Computação, corrigidos pela *insistência do real*, são corriqueiros em situação de interação entre usuário e programa. Por exemplo, é o que acontece quando um controle de interface tem o rótulo "CANCELAR", mas a ação de programa que o controle dispara *não é um cancelamento*. Pode ser uma mera *suspensão* de processo. É um erro de programa, que pode ter passado despercebido nos testes. Se o usuário vier a descobrir "a mentira", será provavelmente à custa de problemas (por exemplo, a persistência de informação cancelada e, por isto, indevida, na base de dados). Estes problemas de uso do programa o farão entender que "CANCELAR" não quer dizer cancelar. Na maioria dos casos, os usuários acabam descobrindo *outro jeito* de fazer o que querem. Podem descobrir, por exemplo, que se fecharem a janela de diálogo sem apertar o botão "CANCELAR", o efeito (talvez colateral) é o desejado cancelamento do processo de interação em curso. Esta questão do significado interpretado e o referente dos signos na esfera humana tem ramificações muito instigantes para uma teoria semântica empiricamente consistente da computação.

[6] É interessante atentar para os títulos dos livros de Santaella (2004), **O Método Anticartesiano de Peirce**, e de Chomsky (1966), **Cartesian Linguistics: A Chapter in the History of Rationalist Thought**. O embate entre duas concepções filosóficas opostas não poderia ser mais claro, o que ilumina a linha de argumentação que apresento.

[7] Existem teóricos na Computação que defendem e utilizam teorias Peirceanas sobre abdução, especialmente para a Inteligência Artificial (Josephson e Josephson, 1996). O tipo de uso que propondo neste trabalho, porém, é mais radical, porque pretendo abarcar também o *uso da computação*. Uma possível exceção seria o trabalho de Thagard e Shelley (1997), pensadores talvez mais próximos da visão que estou ensaiando.

Abordando agora o problema da mentira intencional, consideremos o tipo de análise semiótica oferecida para um caso de *hacking*. Para simplificar, estabeleçamos o caso como aquele onde **um programador deliberadamente registra "uma mentira" no código de seu programa**. Por exemplo, ele comunica que o usuário está falando com o website do banco onde tem conta corrente, quando na realidade está fornecendo dados bancários para um servidor pirata. Há alguma construção teórica capaz de sustentar a hipótese de que *o programa* está mentindo para o usuário? Qual o *status ontológico* de um programa semanticamente correto se para os usuários seu significado é tão gritantemente incorreto? Será ele o mesmo que o de uma folha de papel onde o texto impresso comunica uma mentira ao leitor? Se a resposta for positiva — sim, é o mesmo — estamos diante de possibilidades talvez surpreendentes no que concerne nosso juízo a respeito das ameaças que para nós representa a Inteligência Artificial. Os *sistemas inteligentes* são *agentes* ou não? São *imputáveis* ou inimputáveis? É a eles que devemos temer? O discurso *metonímico*, que distraidamente (ou não) usa a criatura para se referir ao criador, parece ser excelente combustível para falsas crenças e falsas questões sobre a tecnologia atual.

5.3 Teorias semióticas da computação

Como se vê, o fio de teorias semióticas no território da computação percorre um longo caminho, e por onde passa gera uma onda sonora de impacto, que faz tremer algumas bases pouco (ou mal) questionadas até aqui. A mágica é colocar emissor, mensagem, destinatário, código, contexto e canal de comunicação dentro de uma mesma *ontologia* teórica. Qualquer destes elementos que fique fora de uma teoria potencializa o surgimento de teorias autônomas do significado prático da computação, e por conseguinte a discussão fragmentada de um fenômeno que é experimentado em sua íntegra.

Um teoria semiótica da computação, que pense seriamente sobre falibilismo e processos de autocorreção, sobre lógica abdutiva e imaginação, e sobre a possibilidade de qualquer signo ser usado na comunicação humana *sem compromisso com uma realidade factual* organiza o território da computação de forma muito diferente do que se vê na atualidade. Se um programa de computador passa a ser teorizado como uma *mensagem* de quem o criou para quem o utiliza, uma das principais tarefas da teoria passa a ser a descrição (empirica, conceitual e cognitivamente adequada) de **como a produção sígnica é restringida ou transformada ao passar pelo *meio* computacional**. Em outras palavras, a teoria tem de dizer como aquilo que o emissor da mensagem *quer dizer* — de que forma, quando, onde, por quê e para quê — se codifica, chega ao(s) seu(s) destinatário(s) humano(s) e atua sobre este(s) destinatário(s) e a realidade mais ampla a sua volta. Note-se que esta descrição engloba, como seu objeto, todo o processo de comunicação humana mediada por computação, incluindo necessariamente teorizações sobre computabilidade, representações formais, inferências lógicas, e tudo o que as teorias tradicionais da computação já se propõem a descrever, bem como teorizações sobre a construção de artefatos de software que concretizam este processo e sobre como as pessoas se expressam através de linguagens computáveis que constituem estes artefatos, seja na função de programadores, seja na função de usuários.

Um ponto crucial desta teoria é o de que nenhum ser humano é capaz de experimentar em ambiente *natural* um processo de comunicação desta ordem. As características deste processo *definem* um tipo de comunicação *artificial* por excelência, pois, segundo os semioticistas em que nos inspiramos (Peirce (1958), Santaella (2004)), a semiose é um processo *inerente à vida humana*.

Assim sendo, uma das conjecturas que me parecem autorizadas por uma teoria semiótica é a de que uma comunicação humana *maximamente eficiente e eficaz* com (ou através de) programas computacionais só seria possível se os humanos *se moldassem ao processo computacional*, se "automatizassem", abrissem mão de sua humanidade. Eis aí novamente a ponta de outro fio que vai muito longe não só no território da Computação, como também no da Filosofia da Tecnologia. Corremos o risco de nos desumanizar para fazer melhor uso, ou tirar maior proveito, da tecnologia?

6 Uma engenharia semiótica

As ideias e perguntas apresentadas acima instigaram e motivaram nosso grupo de pesquisa para elaborar e desenvolver o corpo teórico da *Engenharia Semiótica* (de Souza (2005), de Souza e Leitão (2009), de Souza et al. (2016)). O pressuposto fundamental de toda a teorização é o de que *software é uma mensagem "performática" enviada de quem o produz* (*emissores*) *para quem o utiliza* (*receptores*). "Mensagem performática" quer dizer, neste contexto, que ela ***se emite***, ou seja: **se torna emissora da mensagem que ela é**, tão logo um usuário entra em interação com ela.

Figura 1: Territórios de significação separados para construtores e usuários de software

Esta visão se opõe à visão ainda dominante entre designers e desenvolvedores de *software*, segundo a qual qualquer programa com o qual os usuários interagem, seja ele um programa simples ou complexo, equivale a um produto "manufaturado", onde quem fabrica e quem usa vivem em contextos distintos e isolados um do outro. A **Figura 1** mostra uma caricatura do isolamento entre desenvolvedores e usuários de software. Um grupo *entrega* o produto e o outro *usa*. É interessante lembrar que tanto a *produção* (análise, design, programação, teste) quanto o *uso* de software estão dentro da **esfera empírica concreta da computação**, tratados por subdisciplinas (design e engenharia de software e interação humano-computador) que não pertencem à **esfera teórica conceitual tradicional da computação**. Portanto, o que se vê é que as fraturas

no território da Computação existem mesmo dentro de esferas que deveriam ser coesas.

Ao teorizar que *software é uma mensagem "performática" enviada de quem o produz* (*emissores*) *para quem o utiliza* (*receptores*), a Engenharia Semiótica restabelece a conexão entre as partes isoladas pela fratura e transforma todo o processo de produção e uso de artefatos de computação em uma coisa só. Como sugerido na **Figura 2**, o *software* comunica uma mensagem de seu(s) construtor(es). Ela vai se abrindo para o(s) destinatário(s) à medida em que ele(s) *interage*(*m*) com o software. É uma situação muito parecida com a de uma peça de teatro. A peça comunica uma mensagem do autor para seu público. Mas o público só compreende a mensagem inteira depois da atuação de todos os personagens, em todas as falas, ao longo de toda a peça. Assim, os usuários só entendem completamente a mensagem dos criadores do software à medida que exploram e empregam todas as suas possibilidades e características de funcionamento. E, da mesma forma como descobrimos sempre um sentido novo ao reencontrar uma peça de teatro, como usuários, nós também sempre descobrimos um sentido novo na interação com software. É próprio da nossa vida contingente; é próprio da semiose humana. Um lembrete importante, a esta altura, é que na teoria Peirceana que adotamos, a forma como interpretamos um signo (programa ou software) não está necessariamente certa. Podemos descobrir mais tarde (ou nunca descobrir em nosso tempo de vida) que aquilo que que a máquina executa é de fato muito diferente do que acreditamos. Só não percebemos a diferença.

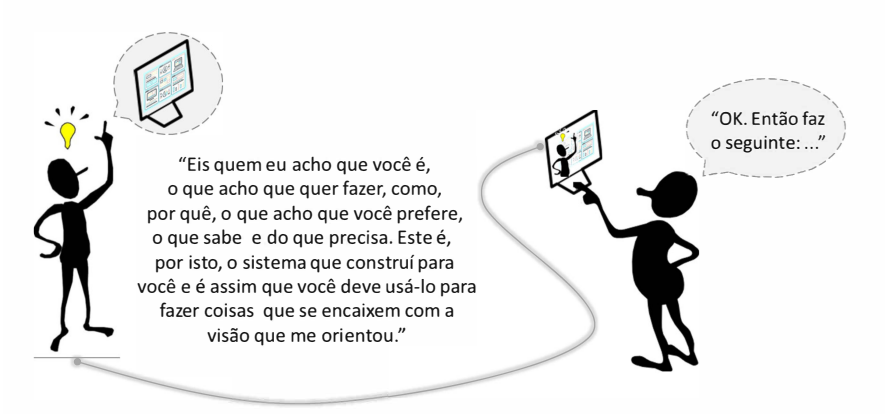

Figura 2: Ao longo do uso, a mensagem do criador de sofware se comunica para o usuário

O processo da **Figura 2** é um tipo de *meta*comunicação; uma comunicação *sobre* como, onde, quando, por que e para que comunicar. O software é uma réplica automática e autônoma do emissor da mensagem (pois o software a *emite*). O software é também *a mensagem* (pois a contém em sua integridade). É também o código em que a mensagem está escrita (pois todo software contém e impõe aos usuários a sua própria linguagem de interação). É ainda o canal de comunicação (pois é pelo software que a mensagem transita entre os interlocutores). E, por fim, o software contém e impõe um **molde** de interlocutor e interlocução, no qual – para uma interação produtiva – os usuários têm de se encaixar. Esta caracterização evidencia a reflexividade da teoria da Engenharia Semiótica, que tem por objeto de investigação este fenômeno específico de metacomunicação através de software. Ela é uma teoria semiótica de certos aspectos da computação.

Na definição das tarefas que a teoria tem em relação a seu objeto de investigação, ficam claros quais os aspectos da computação que ela se propõe a caracterizar.[8] Em seu estado atual de desenvolvimento, a teoria pretende:

- Definir o que é o fenômeno de metacomunicação na computação;
- Definir em que circunstâncias, sob que condições, através de que meios e com que recursos e instrumentos o fenômeno acontece na realidade observável;
- Definir os participantes desta metacomunicação e as condições semióticas impostas pelas propriedades e restrições da *computação*, que é a mediadora do processo comunicativo; e
- Identificar as implicações teóricas e práticas de seus achados para a Ciência da Computação (plano conceitual de Cantwell-Smith) e Engenharia de Software (plano empírico).

É importante ressaltar que estas tarefas não são diferentes se a teoria for aplicada a um editor de texto, a um ambiente de simulação 3D, a um programa de reconhecimento facial, ou a um sistema de recomendação de produtos. Para a teoria, a finalidade do uso da computação não importa. É o mesmo que ocorre com uma teoria pragmática da linguagem, para a qual importam os *tipos* de atos de fala possíveis em diversos *tipos* de situação e não as *instâncias* destes tipos em ocorrências singulares de fala.

Em anos recentes somou-se a estas quatro grandes tarefas uma quinta, que tem talvez um caráter mais próximo de meta do que de tarefa. Trata-se de *oferecer elementos para o desenvolvimento mais intensamente transdisciplinar da* ***Retórica Digital***. Retórica Digital, como nicho de estudos na fronteira entre a arte, a literatura, a comunicação social e a computação, tem por objetivo compreender e explorar as possibilidades retóricas das (ou nas) tecnologias digitais. Como caso exemplar da existência de uma muralha separando teorias da computação de teorias do uso da computação, a Retórica Digital tem se desenvolvido até aqui dentro das Ciências Humanas (principalmente) e Sociais. No domínio do que tradicionalmente entendemos por Computação, a Retórica Digital é assunto raríssimo, senão inexistente. E no entanto, **na prática**, há programadores que entendem e praticam brilhantemente as possibilidades retóricas de seu ofício. É o caso de Don Ho, engenheiro de software e desenvolvedor do **Notepad++**, um editor de textos muito utilizado por programadores profissionais e diletantes. Don Ho faz ativismo político no processo de instalação do Notepad++. A constatação de seu ativismo está na página de *download* do editor, onde há versões tais como: a versão **Stand Up for Ukraine** (Notepad++ 8.4); a versão **Declare Variables not War** (Notepad++ 8.3.2); a versão **Boycott Beijing 2022** (Notepad++ 8.3.1); e várias outras. Na **Figura 3** ilustramos o que acontece na versão **Je suis Charlie** do Notepad++ (6.7.4), quando ao final do processo de instalação o editor abre, automaticamente, e começa a imprimir na tela, letra-a-letra, uma mensagem de protesto contra os atos terroristas de 2015 em Paris, quando 12 pessoas foram mortas por dois homens mulçulmanos depois de o jornal satírico Charlie Hébdo publicar piadas sobre líderes islâmicos, inclusive Maomé. A mensagem automática de Don Ho, no ato de instalação de seu software, diz:[9]

[8] Usamos neste momento o termo *caracterizar* para não entrarmos na discussão, necessária futuramente, sobre se a Engenharia Semiótica é uma teoria descritiva, explanatória, hermenêutica, ou de qualquer outra natureza.

[9] Há um vídeo demonstrativo da instalação para ser visto ou baixado em: http://www.hcc.inf.puc-rio.br/EMAPS/userfiles/downloads/ JeSuisCharlieInstallation.mp4.

> Freedom of expression is like the air we breathe, we don't feel it, until people take it away from us. For this reason, Je suis Charlie, not because I endorse everything they published, but because I cherish the right to speak out freely without risk even when it offends others.
> And no, you cannot just take someone's life for whatever he/she expressed.
>
> Hence this "Je suis Charlie"edition.
> - #JeSuisCharlie[10]

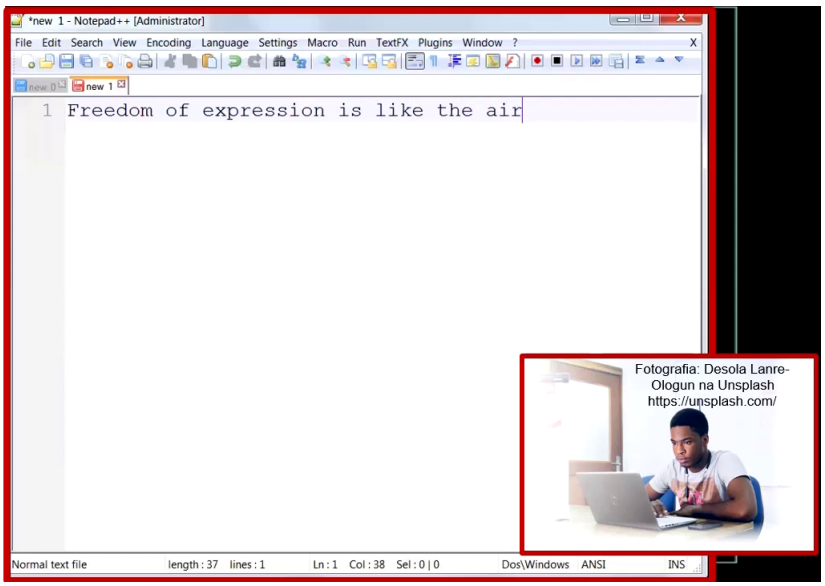

Figura 3: Mensagem automática após instalação do Notepad++ 6.7.4 versão **Je suis Charlie**

A Engenharia Semiótica oferece meios para a análise de duas mensagens de metacomunicação do Notepad++. A primeira, é a mensagem de ativismo, que não se propõe a entrar em *diálogo*, mas é apenas um ato expressivo do emissor, o criador do software (**Figura 3**). A segunda, mais elaborada, é a mensagem de metacomunicação interativa, uma representação automática (no sentido primário de *autômato*) do conhecimento, das crenças e das intenções que o emissor humano (um único indivíduo, no caso de Don Ho, ou mais tipicamente um grupo de indivíduos que inclui proprietários, projetistas e desenvolvedores de um programa ou aplicação computacional) tem a respeito de: **(1)** o que é uma conversa ou interação mediada por computador; **(2)** que estruturas e códigos podem ser usados como *linguagem de comunicação* (quando, onde e para quê); **(3)** que significados as expressões desta linguagem devem ter; **(4)** que conjunto de efeitos (respostas previstas) se espera que cada significado tenha num processo de comunicação efetiva; e **(5)** que reações o programa ou aplicação deve exibir para cada resposta ou ação interativa do usuário, seja ela esperada ou não. Na (**Figura 4**) vemos um instantâneo de interação onde, através de menus e botões em barras de ferramentas, acompanhados de seus respectivos comportamentos, Don Ho (um indivíduo que faz questão de se deixar conhecer e de se comunicar através de software), através de seu representante ou *proxy* automático, *conversa* com seu usuário.

[10] A liberdade de expressão é como o ar que respiramos, não o sentimos até que as pessoas o tirem de nós. Por esta razão, *Je suis Charlie*, não porque endosso tudo o que eles publicaram, mas porque prezo o direito de falar livremente sem risco, mesmo que isso ofenda os outros. E não, você não pode simplesmente tirar a vida de uma pessoa por causa do que ela disse. Por isto, esta edição de "Je suis Charlie". – #JeSuisCharlie

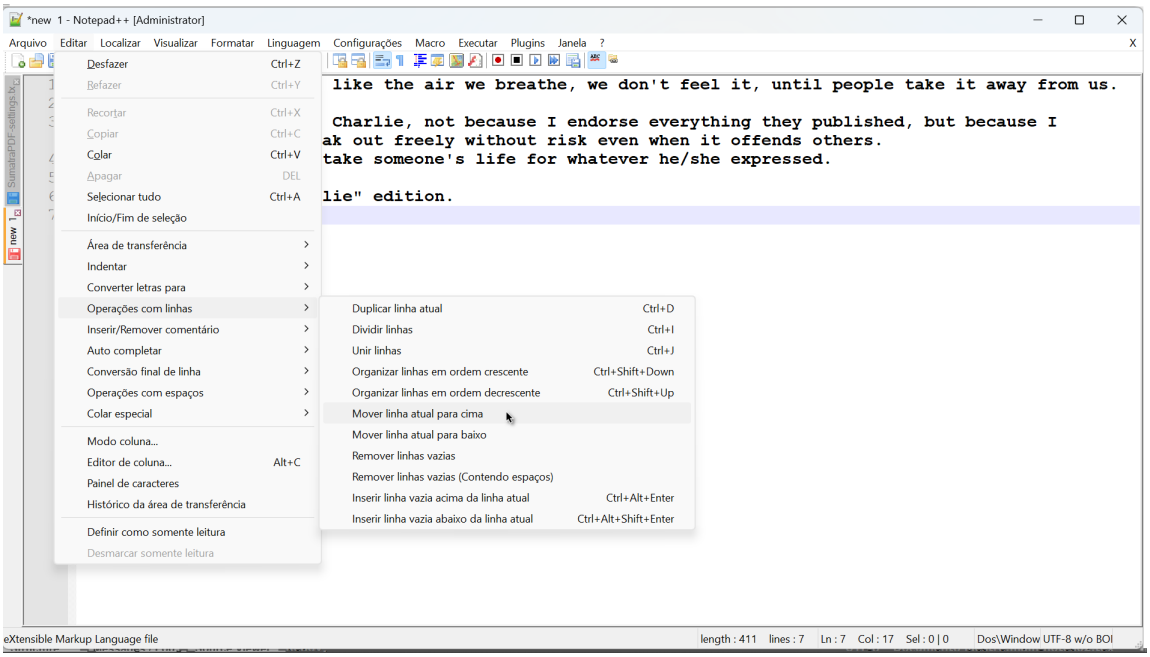

Figura 4: Recursos de interação do Notepad++, comunicando o que o editor pode fazer e como (v. alternativas de atalhos à direta do menu)

A teorização fundamental da Engenharia Semiótica é exatamente esta: um programa interativo é um autômato que representa, expressa, apresenta e põe em execução um modelo de crenças que os emissores da metacomunicação têm sobre *si mesmos* e sobre *os outros*. Tornam-se, por isto, **o signo do fundamento subjetivo e reflexivo da computação**. Nesta qualidade, o espaço para estudos de retórica digital que nossa teoria abre é muito amplo e, quero crer, atrativo. O desafio é como ligar o fio condutor que já costura o processo de desenvolvimento de software ao processo de interação entre pessoas e sistemas de Souza et al. (2016) ao fio que vem do território da teoria abstrata e formal da computação, transpondo (quem sabe?) a muralha ontológica de que nos fala Cantwell-Smith (1996).

7 Comentário final

As reflexões apresentadas neste artigo são exatamente isto: *reflexões*, conjecturas, especulações. Há muitas zonas cinzentas e outras mais escuras ainda, onde nenhuma construção teórica ilumina o caminho que se quer percorrer. O que mantém vivas as possibilidades que imagino e descrevi é o fato de as teorias semióticas oferecerem um ferramental teórico-metodológico para se fazer muita coisa em computação. Terem permitido estabelecer coesão entre duas áreas — desenvolvimento e uso de software — amplamente tratadas de forma isolada na pesquisa teórica é promissor. Mais promissor ainda é apontarem para a possibilidade de articular em um campo mais robusto de investigação a Semiótica Computacional e a Retórica Digital.

No entanto, a muralha ontológica lá está. É preciso, como foi dito anteriormente, *girar o caleidoscópio* para explorar esta outra possibilidade: a de encontrar um fio condutor entre o chamado

núcleo duro da Computação e os seus limites mais externos, onde *o que importa acontece*. Em 2001, Paul Dourish publicou um livro com um título quase igual a esta expressão: *Where the Action is* (Dourish, 2001), aclamado como "o primeiro livro a oferecer uma visão ampla de como a nossa interação com computadores está emaranhada com o mundo físico". Seguindo uma linha fenomenológica e trazendo para sua reflexão filósofos como Heidegger e Wittgenstein, o autor defendeu a primazia da prática vivida sobre as abstrações cognitivas e os exercícios formais. Tomando a computação social e a interação com tangíveis como exemplo, Dourish continuou a linha inaugurada por Winograd e Flores (1987) uma década e meia antes. É interessante que as duas obras tenham enveredado por uma linha fenomenológica, mas que ambas se prestem, sem esforço, a uma leitura semiótica perfeitamente coerente. Como ilustração, seguem pequenos trechos selecionados de cada uma.

> Ao elaborar um programa, o programador tem em mente uma correspondência sistemática pela qual o conteúdo de determinadas células de memória [no computador] *representam* objetos e relações no domínio de aplicação. (...) O sucesso na programação depende de se projetar uma representação e de um conjunto de operações que sejam tanto *verídicas* quanto *eficazes*. São verídicas na medida em que produzem resultados corretos em relação ao domínio (...) São eficazes em diferentes graus, dependendo de quão eficientemente as operações computacionais podem ser efetuadas. Grande parte do conteúdo detalhado da Ciência da Computação reside em projetar representações que possibilitem a realização de certas classes de operações de maneira eficiente. (Winograd e Flores, 1987, pp. 84–85)
>
> Se escrevo um programa de computador complexo que responde a coisas que você digita [...] o programa é, mesmo assim, o meio pelo qual meu compromisso com você se transmite. (Winograd e Flores, 1987, p. 123)

> Estou mais interessado em interação do que em interfaces e mais interessado em computação do que em computadores. [...] Quero tratar da questão da computação *per se* — de representações ativas incorporadas no hardware e software dos sistemas [...]. Portanto, gigabytes e megahertz não serão discutidos, mas o poder das representações será. (Dourish, 2001, p. 3)
>
> O Capítulo 5 fez uma distinção entre três aspectos do significado — intencionalidade, ontologia e intersubjetividade. Cada aspecto tem diferentes consequências para a tecnologia e o design. [...] Ao olharmos para a interação corporificada em contexto de design, nossa atenção naturalmente se volta para a questão da interpretação. Como uma pessoa pode interpretar e compreender o significado que pode ser comunicado através de uma ação? Que tipos de representação poderiam ser oferecidos por um sistema no contexto em que aconteceu uma ação? O usuário de um sistema será capaz de perceber uma ação, ou simplesmente perceberá suas consequências? Em que casos as consequências, a ação em si e o contexto em que a ação foi realizada têm, cada um, uma função a desempenhar na compreensão do significado que está sendo comunicado? (Dourish, 2001, p. 184)

Pensando sobre as razões pelas quais a *muralha ontológica* lá está, é útil citar dois trechos do prólogo de um livro clássico de programação, *Structure and Interpretation of Computer Programs* (Abelson et al., 1996). Este prólogo foi escrito por Alan Perlis, uma das maiores figuras da história da Computação e ganhador, em 1966, do primeiro *Turing Award*, a premiação científica máxima da área. Disse ele:

> Todo programa de computador é um modelo, mentalmente produzido, de um processo real ou mental. Tais processos, frutos da experiência e do pensamento humanos, são gigantescos em número, intricados em detalhe e, em qualquer ponto no tempo, entendidos apenas em parte. [...] Mesmo que nossos programas sejam coleções de símbolos discretos artesanal e cuidadosamente construídas, mosaicos de funções interligadas, eles evoluem continuamente: nós os modificamos à medida que nossa percepção do modelo se aprofunda, se amplia, se generaliza, até o ponto em que o modelo finalmente atinge um nível metaestável dentro de outro modelo com o qual ainda lutamos. (Abelson et al., 1996, pp. xii-xiii)
>
> [...] computadores não são nunca suficientemente [poderosos] ou suficientemente rápidos. Cada avanço na tecnologia de hardware acarreta empreendimentos de programação mais volumosos, novos princípios organizacionais, e modelos abstratos mais ricos. Cada leitor [deste livro] deveria de tempos em tempos se perguntar "Para que fim? Para que fim?" — mas cuidado para não perguntar demais, ou você vai perder *o barato* da programação por causa da constipação de uma filosofia doce amarga. (Abelson et al., 1996, pp. xii-xiii)

A muralha é alta. Mas, invocando um filósofo que não perde *o barato* da filosofia, penso que faz sentido prosseguir. É Umberto Eco quem fala, através de um dos personagens de seu conto *On Truth: A Fiction* (Eco, 1988) :

> Eu tenho um plano, escuta só. Me põe dentro da armação de um computador e eu vou conversar com uma daquelas míseras [...] máquinas. Sabe o segundo princípio de Turing: um humano simula com sucesso uma inteligência artificial se, em contato com um computador que não sabe com quem está falando, depois de certo tempo o computador acredita que seu interlocutor é um outro computador. (pp. 42–43)

O conto termina com o computador real dizendo ao humano disfarçado de computador que seus criadores (programadores) sabem o que ele (computador) tem dentro dele e, toda vez que ele se comporta de forma que seus criadores entendem, eles (criadores) presumem que têm dentro deles o mesmo tipo de software. E o computador ainda adiciona: À vezes eles suspeitam que o que eles têm dentro deles mesmos depende do que eles colocam dentro de mim. Acham que suas formas de organizar o mundo exterior dependem das enciclopédias com que me alimentaram (Eco, 1988, p. 59). Cantwell-Smith (1996, 2002) concluiu de sua análise que a Computação ainda não é uma *ciência*, com a estatura teórica que se espera de uma, porque, entre outras coisas, lhe falta *filosofia*.

— o —

Agradecimentos
A ideias aqui expostas são fruto de anos de rica e estimulante pesquisa sobre Engenharia Semiótica. Sou grata a todos os alunos e colegas pesquisadores do SERG, que trouxeram perguntas, elaboraram respostas, discutiram falhas e lacunas, em um processo maravilhoso de semiose coletiva. Sou também extremamente grata a Edgar Lyra, meu professor de Filosofia, e aos colegas do grupo de estudos sobre Ética e Mediação Algorítmica de Processos Sociais, o EMAPS – em particular, além de Edgar, Simone Barbosa, Bruno Feijó e Marcelo Jasmin. Os debates destes últimos anos têm sido a melhor experiência intelectual que já tive. Finalmente, tenho muito a agradecer aos professores Hermann Haeusler, Tiago Castro Alves e Tito Palmeiro, pelo animado debate que tivemso no colóquio. Seus comentários e perguntas me ajudaram a clarear um pouco mais as ideias e a enxergar o quanto mais ainda há para clarear.

Referências

Abelson, H., Sussman, G. J., e Julie Sussman (1996). *Structure and Interpretation of Computer Programs*. Cambridge: The MIT Press/McGraw-Hill, 2nd edition.

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning: A Textbook*. Springer Publishing Company, Incorporated.

Allan, K. e Jaszczolt, K., Eds. (2012). *Cambridge Handbook of Pragmatics*. Cambridge: Cambridge University Press.

Alpaydin, E. (2014). *Introduction to Machine Learning*. Adaptive Computation and Machine Learning series. Cambridge, Mass.: The MIT Press.

Andersen, P. B. (1997). *A Theory of Computer Semiotics: Semiotic Approaches to Construction and Assessment of Computer Systems*. Cambridge, MA: Cambridge University Press.

Andersen, P. B., Holmqvist, B., e Jensen, J. F. (1993). *The Computer as Medium*. Cambridge: Cambridge University Press.

Brock, K. (2019). *Rhetorical Code Studies: Discovering Arguments in and around Code*. Ann Arbor, MI, USA: University of Michigan Press.

Cantwell-Smith, B. (1996). *On the Origin of Objects*. Cambridge, Mass.: The MIT Press.

Cantwell-Smith, B. (2002). The foundations of computing. In M. Scheutz (Ed.), *Computationalism - New Directions* (pp. 23–58). Cambridge, MA: The MIT Press.

Cantwell-Smith, B. (2019). *The Promise of Artificial Intelligence: Reckoning and Judgment*. Cambridge, MA: The MIT Press.

Chomsky, N. (1957). *Syntactic Structures*. The Hague: Mouton.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, 2(2), 137–167.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. Cambridge, MA: The MIT Press.

Chomsky, N. (1966). *Cartesian Linguistics: A Chapter in the History of Rationalist Thought*. New York and London: Cambridge University Press.

Chomsky, N., Belletti, A., e Rizzi, L. (2002). *On Nature and Language*. Cambridge: Cambridge University Press.

Constantin, E. e de Saussure, F. (2005). Linguistique générale (Cours de M. le Professeur de Saussure) Semestre d'hiver 1910-1911. *Cahiers Ferdinand de Saussure*, (58), 82–290.

de Saussure, F. (1995). *Cours de linguistique generale (Publié par Charles Bailly et Albert Sechehaye avec la collaboration de Albert Riedlinger)*. Paris: Payot. Édition critique preparée par Tullio de Mauro.

de Souza, C. S. (2005). *The Semiotic Engineering of Human-Computer Interaction*. Acting with Technology. Cambridge, MA: The MIT Press.

de Souza, C. S. (2018). A pragmatic turn in computer science. *Interactions*, 25(3), 20–21.

de Souza, C. S. (2020). Sujeitos ocultos e indeterminados da computação. *SBC Horizontes*, Oct 2020, 1 HTML page.

de Souza, C. S., Cerqueira, R. F. G., Afonso, L. M., Brandão, R. R. M., e Ferreira, J. S. J. (2016). *Software Developers as Users. Semiotic Investigations in Human-Centered Software Development*. London: Springer International Publishing.

de Souza, C. S. e Leitão, C. F. (2009). *Semiotic Engineering Methods for Scientific Research in HCI*, volume 2 of *Synthesis lectures on human-centered informatics*. San Rafael, CA: Morgan & Claypool.

Diverio, T. e Menezes, P. (2009). *Teoria da Computação - Máquinas Universais e Computabilidade*. Porto Alegre, RS: Bookman Editora, 3a edition.

Dourish, P. (2001). *Where the Action Is*. Cambridge, MA: The MIT Press.

Eco, U. (1976). *A theory of semiotics*, volume 217. Bloomington, IN: Indiana University Press.

Eco, U. (1981). The Theory of Signs and the Role of the Reader. *The Bulletin of the Midwest Modern Language Association*, 14(1), 35–45.

Eco, U. (1986). *Semiotics and the Philosophy of Language*. Bloomington, IN: Indiana University Press.

Eco, U. (1988). On truth: A fiction. In U. Eco, M. Santambrogio, e P. Violi (Eds.), *Meaning and mental representations*, volume 496 (pp. 41–59). Bloomington, IN: Indiana University Press.

Eyman, D. (2015). *Digital Rhetoric: Theory, Method, Practice*. Ann Arbor, MI, USA: University of Michigan Press.

Farias, P. L. e Queiroz, J. (2017). *Visualizando signos: modelos visuais para as classificações sígnicas de Charles S. Peirce*. Blucher.

Floyd, C. (1992). Software development as reality construction. In C. Floyd, H. Zullighoven, R. Budde, e R. Keil-Slawik (Eds.), *Software Development and Reality Construction* (pp. 86–100). Heidelberg: Springer.

Gabbay, D. M. e Kruse, R. (2000). *Handbook of defeasible reasoning and uncertainty management systems: Volume 4 Abductive Reasoning and Learning*. Dordrecht: Springer Science and Business Media.

Hess, A. e Davisson, A. (2018). *Theorizing Digital Rhetoric*. New York, NY: Routledge / Taylor and Francis.

Hopcroft, J., Motwani, R., e Ullman, J. (2014). *Introduction to Automata Theory, Languages, and Computation*. New Your, NY: Pearson Education.

Jones, J. e Hirsu, L. (2019). *Rhetorical Machines*. Tuscaloosa, Al.: The University of Alabama Press.

Josephson, J. e Josephson, S. (1996). *Abductive Inference: Computation, Philosophy, Technology*. Computation, Philosophy, Technology. Cambridge University Press.

Jucker, A. H. (2012). Pragmatics in the history of linguistic thought. In K. Allan e K. M. Jaszczolt (Eds.), *The Cambridge Handbook of Pragmatics* (pp. 495–512). Cambridge.: Cambridge University Press.

Marino, M. C. (2020). *Critical Code Studies*. Software Studies. Cambridge, MA: The MIT Press.

Nöth, W. (2014). The Semiotics of Learning New Words. *Journal of Philosophy of Education*, 48(3), 446–456.

Peirce, C. S. (1931-1958). *Collected Papers (Vols I-VIII)*. Boston, MA: Harvard University Press. Vols. 7-8 edited by A.W. Burks Other volumes edited by Charles Hartshorne and Paul Weiss.

Psillos, S. (2011). An explorer upon untrodden ground: Peirce on abduction. In D. M. Gabbay, S. Hartmann, e J. Woods (Eds.), *Inductive Logic*, volume 10 of *Handbook of the History of Logic* (pp. 117–151). North-Holland.

Pylyshyn, Z. W. (1980). Computation and cognition: issues in the foundations of cognitive science. *Behavioral and Brain Sciences*, 3(1), 111–132.

Rellstab, D. H. (2008). Peirce for linguistic pragmaticists. *Transactions of the Charles S. Peirce Society*, 44(2), 312–345.

Reyman, J. (2018). The rhetorical agency of algorithms. In A. Hess e A. Davisson (Eds.), *Theorizing Digital Rhetoric* (pp. 112–125). New York, NY: Routledge / Taylor and Francis.

Santaella, L. (2004). *O método anticartesiano de C. S. Peirce*. São Paulo, SP: Editora UNESP.

Scheutz, M. (2002). *Computationalism: New Directions*. Bradford book. Cambridge, MA: The MIT Press.

Shapiro, M. (2022). Language as semiosis: a neo-structuralist perspective in the light of pragmaticism. *Chinese Semiotic Studies*, 18(1), 131–146.

Sowa, J. F. (2000). *Knowledge Representation:L ogical, Philosophical, and Computational Foundations*. Pacific Grove, CA: Brooks/Cole Publishing.

Sowa, J. F. (2008). Conceptual graphs. In F. van Harmelen, V. Lifschitz, e B. Porter (Eds.), *Foundations of Artificial Intelligence* (pp. 213–237). Amsterdam: Elsevier.

Strand, T. (2013). Peirce's new rhetoric: Prospects for educational theory and research. *Educational Philosophy and Theory*, 45(7), 707–711.

Thagard, P. e Shelley, C. (1997). Abductive reasoning: Logic, visual thinking, and coherence. In M. L. Dalla Chiara, K. Doets, D. Mundici, e J. van Benthem (Eds.), *Logic and Scientific Methods: Volume One of the Tenth International Congress of Logic, Methodology and Philosophy of Science, Florence, August 1995* (pp. 413–427). Dordrecht: Springer Netherlands.

Vee, A. e Brown Jr., J. J. (2016). *Computational Culture - Special Issue, Rhetoric and Computation*, volume 5. online open-access: http://computationalculture.net/editorial-issue-five/.

Winograd, T. e Flores, F. (1987). *Understanding Computers and Cognition: A New Foundation for Design*. Boston, Ma.: Addison-Wesley.