A MÁQUINA-CRIANÇA
O Ensino Fundamental em uma cultura tecnológica

ANDRÉ SOUZA LEMOS

RESUMO

Contemplamos o estabelecimento de uma cultura tecnológica que chega à escola anunciando novos materiais expressivos; o novo cria o velho. É possível, entretanto, ver na ciência da computação, tão nova, velhos (belos) entendimentos que estavam esquecidos. O velho, de novo.

Palavras-chave: *cultura tecnológica, ensino fundamental, ciência da computação, tecnologia da informação, informática e educação.*

LA MÁQUINA-NIÑA
– La enseñanza fundamental en una cultura tecnológica

Resumen: *Contemplamos el establecimiento de una cultura tecnológica que llega a la escuela anunciando nuevos materiales expresivos; el nuevo crea el viejo. Es posible, entretanto, ver en la ciencia de la computación, tan nueva, viejos (bellos) entendimientos que estaban olvidados. El viejo, de nuevo.*

Palabras-clave: *cultura tecnológica, enseñanza fundamental, ciencia de la computación, tecnología de la información, informática e educación.*

O COLETIVO INVISÍVEL

Vivemos os estudos científicos mais como efeitos de campo do que coleções de saberes monumentais, especialmente na medida em que privilegiamos as novas ciências, as fronteiras do conhecimento nascente. Todo campo de estudo científico nascente, entretanto, espera tornar-se legítimo. Este desejo de construir-se, que é um desejo de movimento (e a ciência não deixa de ser produção desejante) contém, paradoxalmente, também um desejo de repouso, de estase dos conhecimentos. Trata-se de um complexo jogo de forças que inclui fatores heterogêneos e no qual a entrada de um novo participante, individual ou coletivo, somente ocorre por meio da desestabilização parcial do campo de estudos a ponto de permitir que novas tendências sejam toleradas. Sendo bem sucedido na sua inauguração, o novo participante deve, num segundo momento, aceitar e colaborar para que se imponha um novo acordo que torne estáveis as suas próprias contribuições, permitindo que o seu nome e a sua terminologia se fixem, de algum modo (Latour, 2000). É uma operação delicada, cujo exercício constante sustenta a serenidade das instituições científicas, e que se desdobra num outro dilema: o corpo da ciência deseja congregar uma comunidade de pesquisadores ativos e angariar reconhecimento público, mas também deseja encontrar consistência própria, que se auto-sustente diante de injunções estrangeiras, ou seja, independente do público. Em tempos de questionamento das formas clássicas do pensamento ocidental, pode-se pensar que esta segunda vertente é mais fraca, e até inferior, sendo uma comunidade científica assemelhada tão-somente a uma comunidade de opinião sofisticada, especializada, ou a uma variedade de empresa. É possível, entretanto, que no próprio movimento de fixação das opiniões resista sempre um pouco de caos. Bem na hora em que tudo parece estar acabado, às vezes justamente por isso mesmo (e enquanto houver movimento) retorna a desordem, que chama a atenção do cientista mais uma vez.

Falaremos aqui da computação como uma ciência nascente. Que o seu objeto não se tenha dado ainda com clareza não é, portanto, uma dificuldade. A rigor, sequer sabemos se se trata de uma só, ou de uma multiplicidade. O "prazo de carência" dado a uma ciência para se estabelecer não é fixo, depende, entre outras coisas, da expectativa que é capaz de despertar. E as expectativas em torno da

computação ainda são muito elevadas, embora já tenha ela – mesmo na sua forma mais particular, mais recente – uma trajetória relativamente longa. Se este é o caso, considerar a sua presença no Ensino Fundamental é um verdadeiro problema, justamente porque é uma urgência. A computação se apresenta na vida prática, violentamente, antes que se tenha a chance de produzir uma doutrina que lhe sirva de suporte. Se fizermos uma comparação com a medicina, estaríamos, no que diz respeito à computação, vivendo tempos pré-modernos. Felizmente podemos operar com uma hipótese que nos dá força suplementar, a de que os sistemas computacionais coincidem com uma forma muito peculiar dos desenvolvimentos técnicos, que se dá na medida em que a complexidade vem ao mesmo tempo de baixo para cima e de cima para baixo, entre a coletividade dos pequenos sistemas e o emaranhado (*meshwork*) dos grandes sistemas, entre a criança e o adulto, entre o leigo e o especialista.[1] Em outras palavras, a computação pode ser, em todas as suas manifestações, sintoma do advento de uma *scienza nuova*. É por isso que nos voltamos para o problema do Ensino Fundamental, como sítio exemplar deste novo desenvolvimento. Tomamos por hipótese que a educação infantil é a caixa de ressonância dos novos modos de produção, e que a presença da informática nos espaços escolares é um drama ainda não resolvido.

Conhecemos as inovações tecnológicas, as encontramos em nossa vida, porque há um movimento destas inovações no sentido de encontrar-nos onde já estamos, e fazer de nós seus usuários. Nós eventualmente as acolhemos – ou não; elas, antes mesmo de que começasse nosso momento de decisão, já preparavam o encontro.[2] É como se as manifestações naturais, tradicionalmente associadas a forças anímicas, fossem gradualmente sendo substituídas pelos artefatos, à medida que estes se desfazem da ligação com o artesão e ganham uma origem remota, sem autoria. Assumem um movimento misterioso, mesmo sendo produção humana rigorosamente planejada, aliás, numa época em que o exercício da racionalidade científica é organizado em escala jamais vista antes. Um desenvolvimento surpreendente, mas explicável: o princípio da natureza sai de cena, é eclipsado, mas não a vida e suas incertezas.

O mistério de que se cerca o movimento das inovações tecnológicas é genuíno, não redunda em misticismo: não se trata de um movimento individual (de cada objeto que se apresenta), ou parti-

cular (de cada caso, de cada encontro), justamente por que no espaço dos indivíduos situa-se a ação dos personagens, e ali os objetos apenas compõem o cenário. Neste lugar aparece a capacidade que têm os usuários, de escolha, ou de determinação dos objetos. Na cena teatral, o movimento próprio dos objetos é quase invisível. Os produtos apelam à sensibilidade do consumidor, mas só o que se pode efetivamente ver é a ação do consumidor, que acumula todas as conseqüências das relações que mantém com cada produto, das quais insiste em ser fonte de sentido, referência, destino último. As determinações do mercado (e outras determinações quase naturais) aparecem como força fraca, ao passo que as iniciativas destes personagens comunicam-se, fazem a sua linguagem, organizam grupos e agentes institucionais, políticos e econômicos.

Desde que a produção se fez produção de consumo – o capital é agente de fluxos, não de produtos – o movimento dos artefatos só se torna visível em sua forma coletiva mais ampla, seja a da tensão criadora entre elemento e conjunto (Simondon, 1958) ou entre modelo e série (Baudrillard, 1989); seja na forma da rede, ou na do campo (ou cena), como no caso de um estilo de vida tomado como objeto de consumo, que é uma presença limítrofe, quase inefável, na fronteira entre o próprio consumidor e o consumo. No caso da computação, numa primeira análise, toda essa dupla perseguição entre os sistemas e seus clientes parece fazer parte de um movimento bastante recente, ainda que a indústria tenha já 50 anos de idade, pelo menos. Na verdade, somente com a *World-Wide Web* – cuja presença entre nós tem pouco mais de dez anos – puderam os sistemas aceitar um usuário verdadeiramente leigo (não só pela sua condição de não-especialista, mas também pelos seus interesses não específicos), ainda assim com enorme resistência, só amenizada pelo reconhecimento tácito de que, afinal, essas máquinas não funcionam muito bem, e admiti-lo faz parte da experiência.[3] Lembremos que há não mais de duas décadas não havia maneira de integrar quem quer que seja ao uso de um computador sem antes fazê-lo deslocar-se, na verdade converter-se a um determinado desejo de movimento. Este é o orgulho dos programadores antigos, a possibilidade de pôr-se em movimento livre, de projetar-se a um espaço abstrato, sem obstáculos. Não há verdadeiramente usuário destes sistemas antigos, não há quem desfrute deles; prestam-se ao confinamento, não à domesticação. Somos compelidos a revisitar,

mais uma vez, a diferença temática entre *1984* e *Admirável mundo novo*. Fazemos a releitura destes romances como reportagens jornalísticas, referentes a ocorrências passadas, não como alegorias de um futuro imaginado, lições de moral. Com Orwell, a privação de contato, o confinamento, a anestesia; com Huxley (que, por sinal, escreveu antes), a suavização dos contatos, uma imensa variação estética cujo propósito é a invariância na totalidade, que nunca cresce ou diminui, destinada que está à reprodução idêntica de si.

A primeira grande bifurcação na história da computação eletrônica teve origem no êxito que os computadores tiveram como sistemas de controle e gestão, para além das tarefas de cálculo numérico a que estiveram entregues desde o início. Nascem juntas duas grandes capacidades, a de *combinação* e a de *registro*. A primeira, prevista e projetada, dá origem à corporificação da computação propriamente dita, enquanto a segunda, que vem a ser inicialmente um apêndice necessário, passa a um primeiro plano dos funcionamentos. Armazenar e buscar, imprimir relatórios e preencher formulários: registrar é lembrar, lembrar é poder antecipar. As empresas modernas, públicas e privadas, sempre fizeram uso de tecnologias de informação e controle. É isso que as define como empresas sofisticadas, ou seja, a capacidade de converter a supervisão direta do processo produtivo em controles sutis e invisíveis, possíveis somente a partir da adoção de sistemas de informação cada vez mais complexos. A utilização deste tipo de sistema é tão vantajosa do ponto de vista da gestão, pelos ganhos de produtividade e de escala que enseja, que mesmo antes da vulgarização dos transistores, quando o preço dos computadores ainda era exorbitante, eles já estavam presentes no mundo dos negócios. A generalização de um artefato técnico é produtora de cultura, e com os computadores não foi diferente. Programar computadores torna-se uma escritura abstrata, distante da interação com a máquina computacional. Uma atividade comum, que cria seus próprios padrões, rapidamente disseminados a ponto de se tornarem universais. Poucos anos se passam, e o imenso volume de programas criados dá origem a uma verdadeira crise ecológica, pressentida desde os primeiros anos.[4] Surge a necessidade de um disciplinamento da programação de computadores, uma vez que os textos que vinham a ser programas, por carecerem de contextualização, perdiam facilmente a legibilidade, mesmo para o seu próprio autor.

A história da computação deu duas respostas a este dilema, baseadas em duas metáforas orientadoras distintas. A primeira, chamada "programação estruturada", deseja encontrar o contexto que permita ler os textos dos programas no ambiente humano em que estão funcionando. Cria-se a imagem de que cada instalação de um computador é uma grande comunidade de informações, regida por um imenso conjunto de programas. Estes textos formam um todo coeso, são integrados numa hierarquia bem definida que organiza a vida do lugar, ao mesmo tempo ordenando e reagindo a sua cultura institucional. Os fabricantes de grandes computadores passam a ser vendedores de soluções sob medida, e cada instalação é vista na sua singularidade. Sendo assim, o problema da profusão de programas passa a ser atacado pela maneira como são relacionados uns com os outros, partes do grande texto da organização. Cada programa, por sua vez, passa a ser um conjunto de subprogramas, que devem ser pequenos e bem escritos, facilitando a leitura. Reutilizar um programa é possível, mas em geral obedece à lógica do lugar, as suas peculiaridades. Propõe-se que a escrita dos programas seja precedida por uma etapa de análise, mais ou menos como se fosse a construção de um grande edifício, que demanda um projeto urbanístico detalhado, análise do seu "impacto ambiental", ou seja, cuidados relacionados à criação de um objeto único, de grande porte, cuja realização é difícil de reverter, ou praticamente irreversível.

O momento "Admirável Mundo Novo" da computação começa quando tem início a estetização do computador, que se torna objeto *pessoal*, fazendo com que sua presença viesse a ser comum em um lugar inteiramente novo. Está nesse evento a origem da experiência que temos hoje dos sistemas computacionais. Mais uma vez fez-se de um agenciamento maquínico uma cultura, e desta vez uma cultura de massa. De início, tornar um computador acessível ao público requeria fazer com que fosse portátil e relativamente barato. A primeira geração de computadores pessoais, entretanto, esteve muito distante de um campo de aplicação realmente amplo, eram máquinas adotadas quase exclusivamente por aficionados. Desprovidas de uma variedade de programas suficientemente grande até mesmo para atrair os profissionais experientes, não despertaram a atenção dos grandes fabricantes. Constituindo apenas o anúncio de algo que talvez estivesse por vir, transformaram-se em um elemento do ambiente contracultural americano nos anos 70. Mesmo quando,

anos mais tarde, os microcomputadores já haviam conquistado um mercado suficientemente grande, quando os primeiros programas de grande sucesso comercial puderam aparecer (*Visicalc*, *dBase*, etc.), inaugurando também um novo mercado para a produção de software e chamando finalmente a atenção das grandes corporações, ainda assim o computador pessoal ainda era um objeto difícil de encontrar fora dos ambientes de negócios.

O que explica a primeira grande entrada do usuário doméstico no mercado consumidor de computadores pessoais é a adoção, para a utilização dos computadores, de um modo de interação totalmente diferente do ato de programar. Programar implica – pelo menos tem sido assim – quebrar a harmonia sensório-motora que permite que qualquer máquina seja operada sem necessidade de raciocínio lógico-formal. Equivale a tratar a máquina exclusivamente de acordo com uma nova – e estranha – norma culta da linguagem escrita, que sequer é o registro com o qual uma elite cultural já estaria acostumada.[5] Um registro muito sofisticado, que no caso dos microcomputadores encontrava-se restrito aos neófitos de uma cultura informática nascente. Paradoxalmente, um meio cultural protagonizado por jovens empresários irreverentes, em geral apaixonados pela informalidade e pelo desleixo quanto aos costumes e os modos de tratamento. Essa nova comunidade culta, porém, produz uma forma de apropriação indireta do seu objeto pelo "vulgo iletrado", e isso constitui uma grande mudança de perspectiva. A introdução da interação baseada em "eventos" (*mouse* + grafismos) substitui a forma sintagmática da linguagem computacional pelo encadeamento de ciclos de ação e resposta. Essa mutação, pela primeira vez, faz com que a indústria da computação diga algo realmente interessante aos que não são aficionados, e um surto de consumo de computadores coloca a *Microsoft* no lugar que ocupa até hoje. Por que uma cultura letrada rejeita formas novas de comunicação que aparentemente pressupõem tão somente o mesmo tipo de letramento? As gírias e os costumes populares têm uma dinâmica muito intensa, feita de espontaneidade, mas a adoção de uma nova língua culta interfere em relações de forças e territorialidades fortes. Afeta a todos, em qualquer nível: seja na aristocracia, em meio aos "escribas", ou entre os iletrados, que aceitam a norma culta na medida em que se submetem as suas determinações, mesmo sem penetrá-las. Leva tempo para acontecer, e freqüentemente implica alteração nos mapas geopolíticos.

O que ocorre com a produção de software, por força desta nova realidade, é que cópias idênticas de cada programa começam a habitar uma quantidade gigantesca de máquinas diferentes, utilizadas em situações inteiramente distintas. Com o tempo, os sistemas computacionais passam a ser *commodities*, itens reprodutíveis à exaustão, muito semelhantes entre si.[6] Uma nova crise da produção de softwares se avizinha: não mais é possível recontextualizar os textos dos programas com base no seu ambiente humano local; precisam poder circular, ter um sentido que se define internamente ao seu próprio jogo.

As interfaces gráficas de janelas vêm da composição de diversas tecnologias relativamente antigas (o novo, na tecnologia da informação, é freqüentemente apenas a soma de antiguidades, penosamente vendidas como criações). Em primeiro lugar, uma interface gráfica é necessariamente um sistema de processos concorrentes, mesmo que resida em um sistema operacional que não favoreça este tipo de funcionamento. Sendo assim, implementar uma interface impõe a superação da idéia de programa estruturado, monolítico, é um sistema que deve estar *à espera de eventos*, ou seja, não impor um roteiro fixo. Em segundo lugar, é um sistema que tem uma relação direta com o usuário, que vem a ser bastante complexa, muito mais sofisticada do que aquela que ocorre nas camadas mais profundas de um sistema operacional, que também se organiza como sistema de processos concorrentes. Em suma, as interfaces gráficas forçaram a adoção de princípios de implementação de sistemas operacionais, somados à idéia de ambiente virtual baseado em tipos abstratos de dados que já vinham sendo experimentados nos anos 70. Consagram-se novas linguagens de programação ditas "orientadas a objetos": *Smalltalk* é pioneira, C++ a mais difundida. Estas linguagens dão origem à idéia de programar como mobiliar dinamicamente um espaço vazio, e não é à toa que os sistemas de desenvolvimento passam a se chamar *ambientes*.

A difusão da Internet cria a possibilidade de uma retomada deste mesmo salto em um outro grau de complexidade. De início uma interface hipertextual passiva (a primeira fase da *World-Wide Web*) rapidamente dá lugar a novos modelos de computação distribuída, ligados à possibilidade de descolar definitivamente os aspectos de infra-estrutura dos sistemas computacionais, que são inevitavelmente territorializados, do contexto das aplicações. Estas passam

a habitar um universo à parte, fazendo o sucesso da plataforma *Java*, que se movimenta harmoniosamente entre estes dois mundos sem jamais confundi-los. Nunca antes foi assim imprescindível pensar a programação como uma escritura que se perde das territorialidades concretas, mas que encontra outras restrições ainda mais sufocantes – mais abstratas e compreensivas – nas novas plataformas e ambientes de desenvolvimento.

Desterritorializações muito rápidas às vezes produzem efeitos paradoxais. Esta proposta de uma programação baseada em classes, apesar de ser vista geralmente como uma libertação de condicionamentos do passado, desde os seus primórdios pretende retomar o momento orwelliano da tecnologia de processamento de dados, momento esse que irá flertar cada vez mais intensamente com o ideal de perfeição axiomática da engenharia de *software*. Um esforço que mal se separa de um outro movimento, o da busca de uma dogmatização suficiente da ciência prática da computação, que a converta em carreira profissional reconhecida, ou seja, um desejo de fixação identitária, de endurecimento, de segurança. Assim como os sistemas de cuja concepção participa, o profissional de informática vive também uma angústia da invisibilidade.

O esforço da nova indústria que se produz a partir desse momento é propositalmente anestésico: uma vez construída uma moldura lógica suficientemente resistente para os funcionamentos, deveria ser possível isolá-los inteiramente dos fatores "extralógicos", experienciais, ou seja, dos acontecimentos. Ocorre que esta empreitada – um "catálogo do possível" para a engenharia de *software* – sempre acabará sendo complexa demais para se converter num facilitador, e a programação orientada a objetos tem feito da programação uma atividade ainda mais esotérica do que antes. Não há como saber o que teria produzido este movimento, caso tivesse sido amplamente bem-sucedido também como método matemático, o que lhe daria uma autoridade incontornável. Entenda-se, esta possibilidade permanece viva, ainda que em estado latente, capaz de um dia dar sua resposta final.

O que se deu no lugar de uma arquitetura universal para a produção de *software* foi uma política de restrição à produção de *software* como um todo, de extermínio das variações oriundas da própria existência de uma comunidade profissional de programado-

res, agora livre de restrições de contexto. Duas iniciativas se associam nesse sentido: a primeira, apenas uma reprise de esforços mais antigos, é o avanço na criação de padrões de mercado cada vez mais extensos e compreensivos, que com o tempo criaram verdadeiros *frameworks* (pré-estruturações) de ampla utilização no projeto de *software*. Padrões de *hardware* e de sistema operacional, padrões de interconexão de redes e de sistemas, padrões de interface de todas as formas imagináveis, e, o mais importante, padrões de concepção, projeto e documentação de sistemas. Criações coletivas, de início propostas ou adotadas por inovadores das mais diversas origens e tamanhos[7], logo são encampadas por grandes corporações ou consórcios de empresas e organizações (governamentais ou não), e passam a habitar novas linhas de produtos e serviços. Estes padrões afirmam-se com tal força no mercado, sua presença é universalizada de tal forma, que aparecem junto ao consumidor final e ao pequeno projetista quase como fenômenos da natureza, assim como a energia elétrica ou as estações do ano. Nesse ponto, a nova economia dos sistemas de informação utiliza-se de estratégias muito semelhantes às das revoluções industriais do passado.

A segunda iniciativa é o fechamento seletivo do *software* proprietário à interferência da comunidade das pequenas empresas e dos projetistas autônomos. O trabalho destes programadores continua sendo importante, mas precisa ser restrito a repertórios de ações bastante limitados (Ullman, 2000). Para que não interfiram na natureza do produto, é preciso bloquear o desenvolvimento dos sistemas abertos e livres em sua multiplicidade. Mantém-se a identidade destes sistemas no mercado sob o controle rígido da empresa que retém direitos de propriedade sobre eles, o que dá ao singelo ato da sua utilização a forma de uma mercadoria ou serviço essencial, ao qual o acesso deve ser autorizado, e cujo fornecedor é um só.[8] Como resultado disso, o panorama da ciência da computação de hoje é semelhante ao da autoridade eclesiástica medieval: baseada num código escrito em uma língua separada do vernáculo, restrita ao mandado de uma casta fechada de hermeneutas, protegidos por um aparato jurídico que aprenderam a controlar. Nem por isso desejamos automaticamente uma reforma protestante, pelo menos não antes de sabermos a que serve.

O ENSINO DA COMPUTAÇÃO É FUNDAMENTAL

Não se inclui o conhecimento da ciência da computação no Ensino Fundamental (que seria muito diferente do mero uso de tecnologias da computação em situação de ensino-aprendizagem, assim como assistir filmes é diferente de entender como são feitos) por se tratar de um conhecimento considerado puramente matemático em sua origem, e que só vai constituir especificidade em um grau muito sofisticado (ou especializado) da sua expressão técnica. Uma dupla decepção: um conhecimento visto como sofisticado demais para o público jovem, específico demais para o público geral, e, ainda que assim não fosse, que não constituiria conteúdo independente, sendo tão-somente acréscimo ao currículo da educação matemática, já tão carregado. Curiosamente, não se estranha o fato de que muitos *hackers* sejam bastante jovens, e que a sua formação matemática, na maioria dos casos, seja muito deficiente. Também pouca gente estranha que o cotidiano do projeto de *software* prescinda quase que totalmente do conhecimento da teoria matemática da computação, apesar dos repetidos esforços acadêmicos no sentido de provar o contrário. Finalmente, muito pouca gente sabe que não existe propriamente uma didática para o ensino da programação de computadores, uma disciplina com larga tradição de autodidatismo, enorme dificuldade para a produção de qualquer conhecimento duradouro (que possa ser transmitido a gerações futuras), histórica dificuldade na formulação de uma linguagem comum[9], e na gestão do trabalho coletivo (Brooks, 1975). Em suma, a negação de todas as características elementares do trabalho científico metódico.

Por onde anda a ciência da computação que não dialoga nem com a prática da computação e nem com o interesse geral pela ciência? Mesmo a Física teórica ocupa lugar nos jornais e nas revistas de circulação mais ampla, então por que não o faz também a computação, que está cada vez mais presente, como prática, em cada aspecto da vida humana? Por que não aparece junto às editorias de ciência, mas apenas em guias de consumo (e entre eles os chamados "cadernos de informática" dos jornais)? É possível tentar responder a esta pergunta investigando a construção histórica da ciência da computação como instituição acadêmica, nos corredores e gabinetes universitários. Sabemos, entretanto, que o conhecimento

da computação não é um desenvolvimento tipicamente universitário[10], ainda que os pesquisadores freqüentemente mantenham vínculos com instituições acadêmicas, por conveniência mútua; que os avanços decisivos ocorreram quase sempre mediante a intervenção direta de órgãos governamentais ou empresas privadas, quando não por ação exclusiva destes.

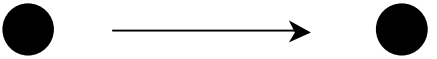
A formação dos currículos universitários de computação é quase sempre reativa às determinações que vêm de fora, o que coloca o fazer universitário em crise, haja vista que nem sequer há reinterpretação do saber que recebe pronto, como pré-condição da sua iniciativa. Neste ponto há uma diferença marcante em relação às demais ciências aplicadas, que, grosso modo, se dividem em dois grupos: algumas (Engenharias em geral) defendem-se erigindo como muralha a disciplina da Modelagem Matemática, uma vez que mantêm um diálogo com a realidade econômica compatível com o velho contrato galileano (a tentativa de estender este contrato à computação aplicada – até o momento – revelou-se pouco frutífera). As demais (Medicina, Economia, entre outras) defendem-se pela necessidade da evocação a uma ética como problema – que a solução axiomática desejaria "resolver" –, e pelo recurso a um certo exercício doutrinário não totalizante, típico da política.

Seria preciso, talvez, encontrar o lugar da ciência da computação em meio a uma história não escrita[11], como se fosse uma forma microscópica de tradição, que ocorre nos interstícios da oficialidade. Uma tradição da qual participamos todos, leigos e especialistas, organizações, coletivos de opinião, e até mesmo agenciamentos coletivos efêmeros, sem vínculos identitários de qualquer natureza. Como se tivéssemos descoberto, por experiência e não por especulação (mas a verdadeira especulação é uma experiência), que o conhecimento transborda continuamente naquilo que se propõe e daquilo que se espera. A partir disso, pensar que só não existe sistema computacional construído verdadeiramente de baixo para cima (*bottom-up*), ou seja, tendo o usuário leigo como protagonista – numa espécie diferente de democracia –, por que é algo que não se tentou seriamente fazer. É uma idéia muito poderosa. Basta que uma primeira experiência bem-sucedida seja realizada e provavelmente não haverá mais como voltar atrás. Algo como o descobrimento da América pela Europa católica: para além da volúpia dos aventureiros (que no caso da computação seriam os *hackers*) e dos interesses

dos seus patrocinadores (hoje: os investidores e suas corporações), ninguém poderia imaginar que no novo mundo surgiria um dia uma outra cultura, um outro sentido de nacionalidade, e um centro de influência capaz de se tornar hegemônico, até mesmo com relação ao velho mundo. Para além de uma comparação meramente alusiva entre os dois eventos, parece mesmo que há algo que se repete, genuinamente.

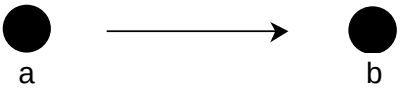
PARTÍCULAS EM MOVIMENTO

Um princípio constituinte da ciência moderna é a generalidade. Quando nos referimos ao movimento de um corpo sólido, por exemplo, é difícil dizer que houve um fluxo, uma vez que o movimento de um sólido não se define pelo seu recipiente, ou mesmo pelas singularidades do seu entorno, sem perda de generalidade. Todas estas condições devem ser traduzidas em jogos de forças. Sequer importa a noção de lugar: em Física há apenas umas poucas constantes, e uma infinidade de variáveis. Mas na experiência computacional é importante poder descrever o movimento de um *token*[12], determinando concretamente os seus lugares de origem e destino; sendo assim, é de fluxos que estamos falando, não apenas de movimentos. E não apenas de fluxos que já se pode antecipar como possibilidade, mas também de fluxos inusitados, aberrantes, ou seja, da virtualidade dos fluxos.

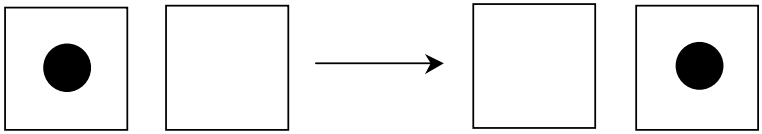


Então, para dizer que um *token* movimentou-se, é preciso dizer que foi de um lugar para outro, ambos conhecidos. Não basta atribuir-lhes literais, como por exemplo *s*, ou s_0, se for o início do movimento. Fazendo isso podemos estar justamente dizendo que não nos importam os lugares, e então perdendo a possibilidade de desencadear funcionamentos singulares, que é justamente a essência do ato de programar. Não que não se possa utilizar nomes, e mesmo nomes que sejam somente uma letra, apenas a qualidade do signo deve ser tal que *indique* um lugar conhecido por experiência, ainda que imaginária ou eventual, e não apenas um lugar geométrico, ou um não-lugar.[13] É preciso poder apontar para um *token* e dizer

"está ali, não está mais aqui". Deve, portanto, ter uma certa identidade, que se dá pela mera presença, ainda que possa não ter nome. O jogo que descrevemos ao programar é real, e não uma experiência de pensamento. Se possui regras, não são regras inventadas, nem regras "especiais". São regras quaisquer, poderiam ser outras, poderão mudar. Mas não são regras simbólicas, esta é uma condição essencial.



lugares diferentes

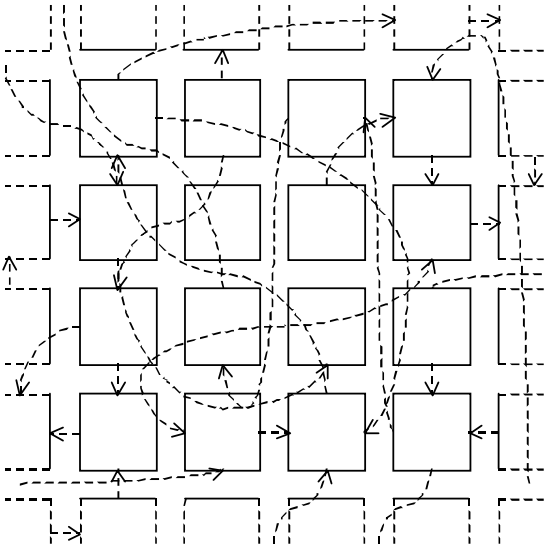


tempos e lugares diferentes

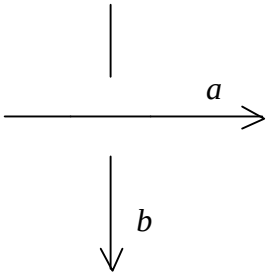
Sendo assim, um *token* está em um lugar, mas não *no lugar de* alguma outra coisa. Não é algo que se possa observar a distância, ou manipular com luvas, mas algo que simplesmente afeta, e de um modo unívoco. O *token* cria tempos e lugares diferentes ao movimentar-se, sendo ele o mesmo. Sofre restrições ao seu movimento e interfere no movimento de outros tokens, multiplica-se, transforma-se e transforma a outros *tokens*. Se tomarmos os sistemas biológicos como sistemas computacionais – e a Biologia é cada vez mais uma ciência computacional –, veremos este aspecto como preponderante, ou seja, são tomados em primeiro lugar como sistemas de afectos[14], e somente em um segundo nível como montagens de representações (afecções). Se tomarmos como ponto de partida que todos os sistemas complexos são sistemas físicos de símbolos[15], descobriremos que as diferentes ciências privilegiam estes dois aspectos diferentemente. A dinâmica dos ecossistemas, tanto em nível microscópico (molecular) como macroscópico (ecossistema), é tomada pela Biologia primeiramente como fenômeno material, ao passo que a tradição logocêntrica da ciência da computação trata dos

sistemas computacionais primeiramente como sistemas de símbolos. Sob este prisma, poderíamos arriscar a suposição de que uma nova ciência da computação será produzida pela Biologia quando completar o seu devir-computacional; uma ciência da qual a computação atual, a "computação artificial", seria apenas uma divisão, ou então seria inteiramente absorvida, sem deixar vestígios. Desnecessário dizer que a Biologia assim concebida será uma ciência inteiramente nova, e as conseqüências disso apenas começamos a vislumbrar.

De qualquer modo, não é este o projeto de ciência mais interessante no ponto de vista do controle de todos estes sistemas, quer dizer, do ponto de vista de um poder que *compreende* a vida. Por esse prisma seria melhor que ocorresse o inverso, ou seja, que a Biologia se tornasse um campo especializado da computação. Seria um prolongamento da relação entre a Medicina e as formas jurídicas (Foucault, 1963), que atinge a sua perfeição no século XX, para num segundo momento encontrar a sua crise. Se quisermos adotar um ponto de vista novo, "materialista", deveremos compreender a todos os sistemas computacionais como parte de um único jogo ilimitado, que se desdobra em uma infinidade de jogos limitados. Podemos, apenas a título de exemplo, descrever um destes infinitos jogos possíveis.

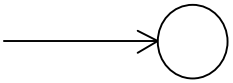


Digamos, para começar, que o espaço em que se movimentam os *tokens* seja um gradeado, uma coleção de lugares, e que alguns movimentos possíveis entre estes lugares sejam viáveis, outros não. Os fluxos interferem uns com os outros, uma vez que uma posição no gradeado comporta apenas um *token* de cada vez, de modo que se dois *tokens* concorrerem para ocupar um lugar, apenas um deles terá sucesso. Além disso, podemos supor que em nosso jogo há a possibilidade de que um fluxo interfira em outro como regra, ou seja, que o *controle*. Isso ocorre quando eles se interceptam, e desse encontro um dos dois surge como determinante do outro. Na figura a seguir, por exemplo, o fluxo de um *token* por *b* somente pode se dar a cada vez que um *token* passar por *a*. Chamamos ao fenômeno de *interrupção*.



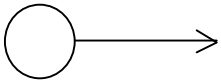
Uma interrupção

Dois fluxos se interceptam.



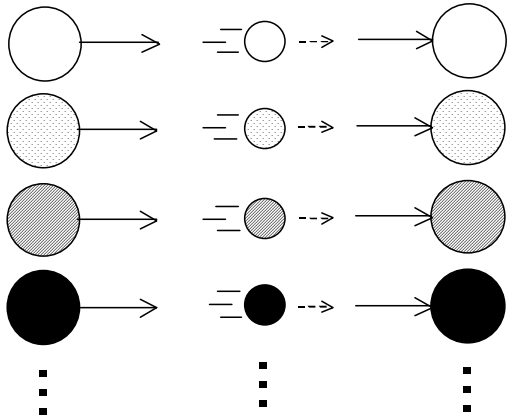
Portas

Fluxos cuja origem (ou destino) é indeterminada.



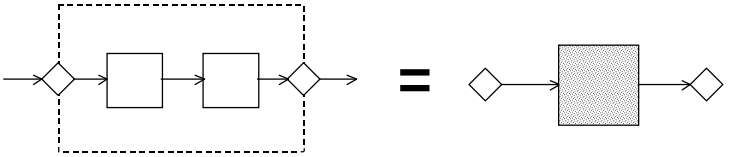
Com o objetivo de observar um comportamento local, podemos recortar um conjunto de lugares e observá-los em separado, como se fosse um sistema à parte. Se, ao fazê-lo, cortarmos pela metade fluxos que não quisermos desprezar, poderemos entendê-los como fluxos cuja especificidade da origem (ou do destino) não é levada em consideração, por não ser determinada. Eles aparecem nos diagramas como *portas*. É interessante notar que o grafismo adotado já começa a dar ares de uma representação abstrata, dado que o campo empírico não está sendo especificado. Adverte-se contra esta interpretação: que se mantenha o ponto de vista de quem observa, não o de quem imagina, ainda que se tenha que imaginar a observação.

Acrescentando mais um elemento de sofisticação ao jogo, podemos assumir que há *tokens* de diferentes tipos, e também que a ação das portas pode ser condicionada a produzir (ou aceitar), *tokens* de tipos determinados.[16] Uma objeção possível: se as portas indicam componentes que estariam em realidade fora do sistema em observação, pode parecer estranho supor-lhes um comportamento, sem conhecer a sua real natureza. Ocorre que esta é a condição imponderável da existência dos sistemas computacionais, nem sempre assumida em toda a sua extensão. Por mais que seus fluxos de entrada e saída sejam mapeados e por mais que se tente impor-lhes uma disciplina, restará sempre um grau de indeterminação no seu comportamento, e, portanto, no funcionamento do sistema como um todo, isso sem considerar as indeterminações que se dão no próprio momento da observação, com seus erros e imprecisões.[17] Em outras palavras, o funcionamento do sistema, enquanto observável, remete a uma virtualidade.

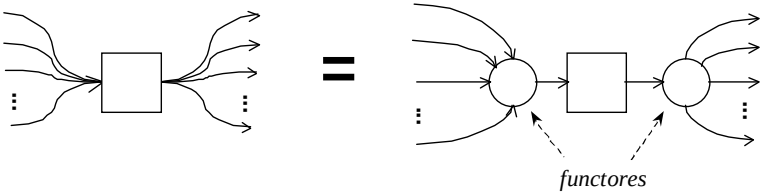


Portas diferentes produzem (e aceitam) tokens diferentes

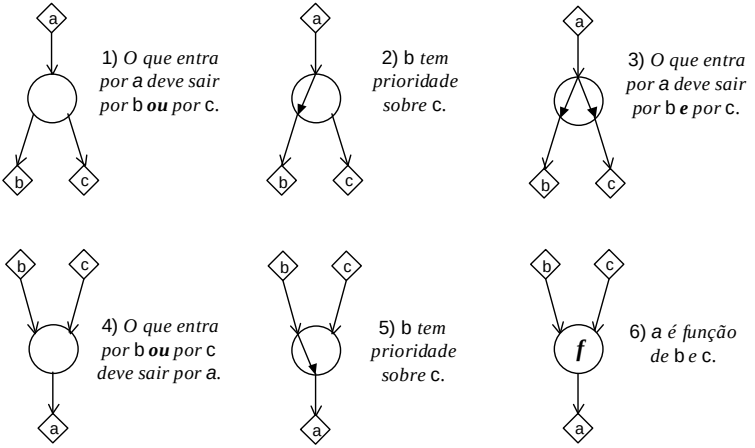
Quando nos deparamos com um sistema composto por uma quantidade muito grande de elementos, é útil distinguir níveis de complexidade, regular o escopo do que está sendo observado. Se for interessante, por exemplo, pode-se agrupar um conjunto de elementos como constituintes de um sistema local, ou subsistema.



Uma outra simplificação interessante é o agrupamento de todos os fluxos que concorrem por uma determinada entrada de um componente – ou pela sua saída –, por meio da suposição de um functor que os reúna em um só. Este functor deve "decidir", a cada vez, qual fluxo ganhará acesso a esta entrada (ou receberá a saída).

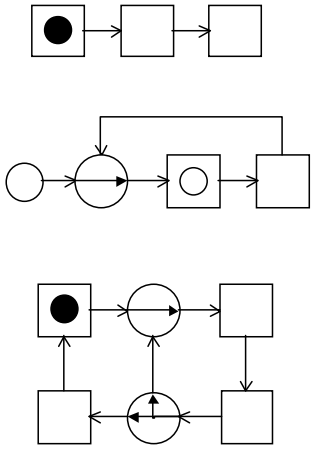


As chances que tem cada fluxo concorrente, seja pela entrada ou pela saída, de conquistar o acesso ao componente em questão, dependerão das circunstâncias. Podemos pensar em algumas variedades que permitiriam conceber uma diversidade ilimitada de sistemas. Por exemplo, o functor mais simples seria aquele que não interferisse na escolha, deixando-a ao acaso (figura a seguir, casos 1 e 4). Uma outra possibilidade seria a de um functor que estabelecesse uma preferência entre os fluxos concorrentes (casos 2 e 5, a flecha indica a saída preferencial). Um fluxo de entrada também pode ser multiplicado, cópias idênticas podem ser distribuídas nas diversas saídas (caso 3). Finalmente, vários fluxos podem dar origem a uma saída que é uma função das entradas (caso 6). Nesse último caso a função deve ser uma composição de elementos mais simples, portas e interruptores inclusive.

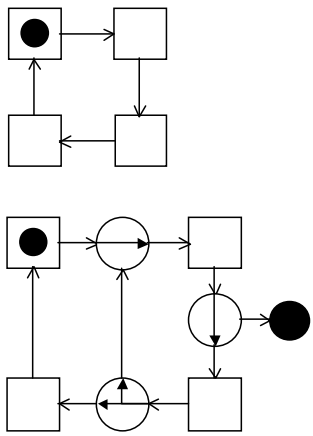


Chamamos aos sistemas completos de *sistemas fechados*. Ainda que contenham portas, seu funcionamento não sofre influência variável de elementos externos. Os subsistemas, se tomados como uma totalidade local, podem ser considerados *sistemas abertos*. Sistemas fechados dividem-se em dois grupos: aqueles que apresentam um funcionamento que termina, ou seja, que chega a uma situação a partir da qual não haverá mais movimentos, e aqueles que produzem ciclos repetitivos, sem parar nunca. As repetições não serão necessariamente idênticas, se o sistema contiver functores que produzam algum tipo de escolha aleatória, mas será possível sempre ver emergirem padrões de repetição.

Sistemas fechados



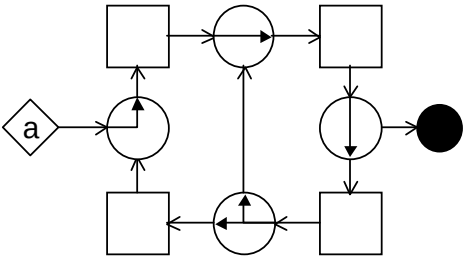
Funcionamentos que terminam



Funcionamentos cíclicos

Já os sistemas abertos exibem uma variedade de comportamentos intrigantes, que despertou o interesse dos teóricos da computação desde o início. Há sistemas abertos que sempre terminam, há aqueles que sempre entram em ciclos, mas há também aqueles que, dependendo do que surgir das entradas (ou puder ser consumido pelas saídas), ora entram em ciclos, ora terminam. Ainda que tomemos a seqüência de entradas como definida *a priori*, e consideremos as saídas como sendo indefinidamente receptivas a tudo o que lhes seja oferecido, ainda assim é impossível conceber um método

universal que permita discriminar formalmente entre os sistemas abertos que são cíclicos, os que terminam e os que às vezes terminam e às vezes não.[18]



Há também funcionamentos abertos que, ora são cíclicos, ora terminam.

O conhecimento da computação carrega consigo as marcas da experiência vivida. A teoria matemática da computação, ao desfazer-se da espontaneidade do fenômeno, é obrigada a desfazer-se também de qualquer outro tipo de exercício empírico, mesmo aquele que visa ao controle do fenômeno.[19] Deve, portanto, divorciar-se inteiramente da Física, o que só é possível diante da pressuposição de uma teoria semântica esvaziada do sentido. A Física de Newton dá acabamento definitivo à colocação do processo matemático como pivô da ciência empírica. Num movimento complementar a Matemática, ao reinstituir-se como ciência, acompanhando a febre do século XIX, é obrigada a se voltar sobre si mesma, uma vez que, por definição, não pode reivindicar objeto empírico. Nasce a Metamatemática, que é a Lógica da Matemática, e também a Matemática da Lógica. A teoria matemática da computação, filha deste movimento, vem a ser então uma Física não-empírica, e não por ser hipotética ou abstrata (justamente ao contrário: pode ser tão concreta e factual quanto se deseje), mas por retirar do movimento toda a característica de conflito, de choque, de dissipação, de conversão de energia, enfim, todo sentido de realidade. Tragicamente, ao tentar produzir instituição, ou seja, ao pretender afirmar-se no espaço dos saberes, o que só se pode fazer mediante inscrição em algum jogo de forças, a Matemática sucumbirá a uma fenomenologia subterrânea, que não ousa dizer o seu nome.

Não há quem encontre algo de profundamente real no funcionamento de um programa que funciona bem. A realidade insiste ali exatamente porque eles não cessam de funcionar relativamente mal, ou apenas quase bem, ou ainda porque isso se torna irrelevante, uma vez que resistem ao aperfeiçoamento planejado. Estranhamente, nos acostumamos a tomar os mal-funcionamentos como resíduos, como ocorrências marginais e desimportantes, ou então como desastres. Ora, desimportantes são justamente as computações que *não* produzem efeitos colaterais, das quais o único produto é o texto do programa e o seu funcionamento esperado, um cadáver. Se nas engenharias de *hardware* o acaso faz parte dos projetos, sob a forma do imponderável que reside no elemento fortemente empírico do trabalho, a única fonte de imprevisibilidade – de inovação genuína – dos projetos de programas é a própria deriva do pensamento, a sua falibilidade. Nos funcionamentos imperfeitos – que apesar disso se sustentam, permanecem viáveis, suficientes – há uma intriga, uma provocação. Ecos de Freud: "o ego não é senhor da sua própria casa", e não haverá agenciamento coletivo que o conduza a uma posição de controle. Seríamos ingênuos se esperássemos que a experiência da computação se desse como um retorno ao âmbito do visível, ao "macroscópico". Ali não há espaço para o ritmo seguro e repousante do nominalismo.

Esta parece ser uma caracterização particularmente negativa do que é real nos funcionamentos computacionais. É importante acrescentar que há certamente uma virtuosidade positiva dos funcionamentos, que é vivida propriamente como contato, ou como afecto. Ela tem lugar na experiência eventual de satisfação, que é a adequação entre o que se esperava e o que de fato se deu no funcionamento (adequação não supõe subordinação, bem entendido). Na duração de um sistema muitas vezes devem ocorrer momentos frágeis de adequação plena, mas estes – ainda que lhe sirvam de consolo, o que não é pouco importante – não são a causa da insistência da ação do projetista, que se sustenta a partir da sua insatisfação, da sua relativa insuficiência.

NOTAS

[1] Papert é pioneiro ao postular que as crianças reinventam o computador quando o encontram (1994). Indo um pouco mais adiante, diríamos que os sistemas computacionais vivem da espontaneidade, aparecem quando o planejamento se mostra impotente, surgem das suas fissuras. Se for assim, comicamente, a principal função do planejamento é falhar, e falhar com generosidade.

[2] Falamos aqui em uma forma histórica de agenciamento dos objetos técnicos, implicada pela existência de um determinado tipo social (o consumidor) e não em uma lei universal das relações entre homens e seus artefatos.

[3] "Em suma, todo mundo que eu conheço (inclusive eu mesmo) sente a vontade de atirar aquela máquina irritante pela janela, pelo menos uma vez por semana. (E agora, graças aos recentes avanços em miniaturização, isso é possível)" (Kapor, 1996).

[4] "Maurice Wilkes, diretor do projeto EDSAC de Cambridge, lembra-se do exato momento em junho de 1949 quando 'hesitando ao virar a curva na escada,' ele percebeu que 'uma boa parte do resto da minha vida seria gasta encontrando erros em meus próprios programas'" (Kohanski, 1998, p. 160).

[5] Há exemplos bem conhecidos deste tipo de estranheza em situações corriqueiras: nem todas as pessoas escolarizadas conseguem sentir-se à vontade diante de um simples terminal bancário automático, justamente porque o tipo de interação que ele pede deve (por razões de segurança) ser muito semelhante ao ato de programar. Apesar de muito restrita, não deixa de ser uma interação sofisticada do ponto de vista da linguagem. Não que requeira propriamente novas habilidades, mas sim uma aceitação de novos pressupostos formais.

[6] Esta história começa com a dos minicomputadores, já nos anos 60, quando os computadores passaram a habitar um outro nicho de mercado (Kidder, 1981).

[7] Muitas vezes em tentativas de constituição de um novo pensamento tecno-científico, de dentro para fora e de baixo para cima, na melhor tradição do iluminismo (Gamma, 1995).

[8] Contra estas limitações surge o movimento em defesa do *software* livre (ver http://www.fsf.org), que se propõe a criar uma nova ecologia para o desenvolvimento de sistemas pelo relaxamento dos direitos de propriedade intelectual sobre os programas (entre outros objetos) na medida em que isso representaria, no mínimo, um cerceamento da liberda-

de de expressão. Nas circunstâncias em que vivemos, ainda é impossível ir além do conceito de programa como produto, justamente porque o usuário leigo ainda é deixado totalmente de fora do processo de concepção (e talvez com ainda mais firmeza), mesmo com a adoção de *software* livre. Seria necessário, para que assim não fosse, um novo regime simbólico, e não apenas uma outra política, e os novos regimes simbólicos não são produto de apenas um movimento isolado. O movimento pelo *software* livre estaria restrito, por enquanto, a desempenhar o papel de uma heresia, ou de uma revolta de artesãos (os *hackers*): um neoludismo.

[9] É preciso fazer a diferença entre linguagem comum e padrão *de facto*. A engenharia de *software* sempre teve a necessidade de definir padrões e normas técnicas comuns, mas nunca produziu um denominador comum para os conceitos que utiliza na definição destes padrões. Brian Smith sinaliza para o fato de que os conceitos fundamentais da computação são na verdade fruto de um consenso irrefletido e tácito (1996, p. 28).

[10] Como de resto o desenvolvimento da pesquisa científica como um todo, desde que se tornou sistemática a integração entre ciência e novas tecnologias, particularmente a tecnologia da informação, especialmente desde o último grande esforço de guerra mundial (DeLanda, 1991).

[11] Assim como minha história não é sempre compatível com a minha memória (e a memória é insubstituível), também a história dos sistemas computacionais jamais substituirá uma memória que se forma no lugar em que eles funcionam.

[12] A palavra, na língua inglesa, refere-se a algo que circula (uma ficha de pôquer, ou um passe de metrô, por exemplo), mas que não tem a mesma generalidade do dinheiro. Um *token* refere-se, então, a um objeto circulante qualquer, mas que não entra em relação de troca, é apenas algo que afeta a outros objetos pelo próprio trânsito. Assim são, nos computadores, as variações físicas de cargas ou pulsos de diversas naturezas (elétricos, magnéticos ou luminosos são os mais comuns) que representam o próprio movimento para outras variações físicas de mesma natureza.

[13] Nesse ponto, o esforço da ciência da computação é o de construir uma teoria dos movimentos e lugares concretos, o que não deixa de ser uma empreitada que inverte o propósito de Galileu, cujo projeto de descrição matemática das leis da natureza (pelo menos do modo como chega até nós, pelos olhos da modernidade) pretende ser cientificamente elegante e simples, renunciando ao circunstancial, ao eventual.

[14] Observar não pode ser apenas identificar-se, reconhecer: "O afecto não é a passagem de um estado vivido a um outro, mas o devir não humano do homem. (...) Ahab não imita Moby Dick e Pentesiléia não 'se comporta como' a cadela: não é uma identificação imaginária. Não é a semelhança, embora haja semelhança" (Deleuze, 1992. p. 224).

[15] A noção é de Allen Newell e Herbert Simon (Newell, 1980. p. 135-183), mas a tomamos sem descartar o problema ontológico, ou seja, sem colocar a ponta seca do compasso no atributo "físico" e a ponta livre no atributo "símbolo" dos sistemas.

[16] Estas diferenças de tipos entre *tokens* permitirão ao observador a contemplação *in natura* de diversas noções não-simbólicas importantes, como a de classe e a de quantidade, por exemplo. Por isso, todo jogo computacional deve conter algum tipo de diferenciação real (não-simbólica) entre elementos em fluxo como elemento constitutivo e gerador. Esta diferenciação também pode ser realizada pela oposição mais elementar entre presença e ausência de um *token* em um lugar, que não foi adotada no exemplo do jogo em questão por privilégio da clareza, sem perda de generalidade.

[17] E há indeterminações de segunda ordem, de terceira ordem, e assim por diante, se pensarmos que o próprio projeto de um sistema terá sido, em algum momento, submetido a um outro sistema (ou processo) de desenvolvimento como entrada, o qual deverá ter passado um dia por um processo semelhante, etc. Esta observação não se restringe aos sistemas considerados "artificiais", embora uma interpretação "naturalista" desta caracterização possa parecer um tanto forçada.

[18] Este é o famoso problema da parada, o *halting problem* ou *Entscheidungsproblem* dos teóricos da computação, que apresentamos aqui numa versão fenomenológica, ou "experiencial".

[19] O que torna mais intrigante o projeto de Stephen Wolfram, que traz de volta o empírico à Matemática – embora diga que se trata de uma "outra ciência", talvez por não querer encarar frontalmente a barreira dogmática que se lhe oporia – pela via aparentemente mais estranha, a da contemplação pura (2002).

REFERÊNCIAS

BAUDRILLARD, Jean. *O sistema dos objetos*. São Paulo: Perspectiva, 1989.

BROOKS JR, Frederick P. *The mythical man-month:* essays on software engineering. Reading, Massachusetts: Addison-Wesley, 1975.

DELANDA, Manuel. *War in the age of intelligent machines*. Nova York: Zone Books, 1991.

DELEUZE, Gilles. *O que é a filosofia*. Rio de Janeiro: Editora 34, 1992.

FOUCAULT, Michel. *Naissance de la clinique*. Paris: P.U.F., 1963.

GAMMA, Erich et al. *Design patterns:* elements of reusable object-oriented software. Reading, Massachusetts: Addison-Wesley, 1995.

KAPOR, Mitchell. "A software design manifesto". In: WINOGRAD, Terry. *Bringing design to software*. Nova York: ACM Press, 1996.

KIDDER, Tracy. *The soul of a new machine*. Nova York: Avon Books, 1981.

KOHANSKI, Daniel. *The philosophical programmer*. Nova York: St. Martin's Press, 1998.

LATOUR, Bruno. *Ciência em Ação.* Como seguir cientistas e engenheiros sociedade afora. São Paulo: Unesp, 2000.

NEWELL, Allen. "Physical symbol systems". *Cognitive science*, 4, 1980. p. 135-183.

PAPERT, Seymour. *A máquina das crianças*. Porto Alegre: Artes Médicas, 1994.

SIMONDON, Gilbert. *Du mode d'existence des objets techniques*. Paris: Aubier, 1958.

SMITH, Brian C. *On the origin of objects*. Cambridge, Massachusetts: MIT Press, 1996.

ULLMAN, Ellen. *The dumbing-down of programming*. Disponível em: http://www.salon.com/21st/feature/1998/05/cov_12feature.html. Acesso em: 2000.

WOLFRAM, Stephen. *A new kind of science*. Champaign, Illinois: Wolfram Media, 2002.