UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Inteligência de Máquina: Esboço de uma Abordagem Construtivista

por

Antônio Carlos da Rocha Costa

Tese submetida como requisito parcial para
a obtenção do grau de
Doutor em Ciência da Computação

Prof. Dr. José Mauro Volkmer de Castilho
Orientador
Prof. Dr. Dalcidio Moraes Claudio
Co-orientador

Porto Alegre, 11 Outubro de 1993.

Resumo

Este trabalho introduz uma definição para noção de *inteligência de máquina*, estabelece a possibilidade concreta dessa definição e fornece indicações sobre a sua necessidade – isto é, dá-lhe um conteúdo objetivo e mostra o interesse e a utilidade que a definição pode ter para a ciência da computação, em geral, e para a inteligência artificial, em particular.

Especificamente, toma-se uma particular leitura da definição de inteligência dada por J. Piaget e se estabelecem as condições para que essa definição possa ser interpretada no domínio das máquinas. Para tanto, uma revisão das noções fundamentais da ciência da computação se faz necessária, a fim de explicitar os aspectos dinâmicos de variabilidade, controlabilidade e adaptabilidade subjacentes a tais conceitos (máquina, programa, computação, e organização, regulação e adaptação de máquina).

Por outro lado, uma mudança de atitude face aos objetivos da inteligência artificial também é requerida. A definição dada supõe que se reconheça a *autonomia operacional* das máquinas, e isso leva a abandonar, ou pelo menos a colocar em segundo plano, o ponto de vista que chamamos de *artificialismo* – a busca da imitação do comportamento inteligente de seres humanos ou animais – e a adotar o ponto de vista que denominamos de *naturalismo* – a consideração da inteligência de máquina como fenômeno natural nas máquinas, digno de ser estudado em si próprio.

O trabalho apresenta os resultados da reflexão através da qual se tentou realizar tais objetivos.

Palavras-chave

Epistemologia da inteligência artificial, fundamentos da ciência da computação, inteligência de máquina, epistemologia genética

Abstract

This work introduces a definition for the notion of *machine intelligence*, establishes the concrete possibility of that definition and gives indications on its necessity – that is, it gives that notion an objective content and shows the interest and utility that the definition may have to computer science, in general, and artificial intelligence, in particular.

Specifically, we take a particular reading of the definition of intelligence given by J. Piaget, and we establish the conditions under which that definition can be interpreted in the domain of machines. To achieve this, a revision of the fundamental notions of computer science was necessary, in order to make explicit the dynamical aspects of variability, controlability and adaptability that are underlying those concepts (machine, program, computation, and machine organization, regulation and adaptation).

On the other hand, a change in the attitude toward the objetives of artificial intelligence was also required. The given definition pressuposes that one recognizes the *operational autonomy* of machines, and this implies abandonning the point of view we call *artificialism* – the search for the imitation of the intelligent behavior of human beings and animals – and adopting the point of view that we call *naturalism* – which considers that machine intelligence is a natural phenomenon in machines, that should be studied by its own.

The work presents the results of the reflexion through which we tried to realize those goals.

Keywords

Epistemology of artificial intelligence, foundations of computer science, machine intelligence, genetic epistemology

Para Carlos e Clara
Com amor e reconhecimento

Agradecimentos

É grande a lista de pessoas e instituições de quem recebi as mais diversas formas de apoio durante a elaboração deste trabalho, e mesmo antes de iniciá-lo. A todos sou grato. Aqui, gostaria de nomear os que estiveram envolvidos mais diretamente com a elaboração técnica e as condições materiais de sua realização.

Em primeiro lugar, aos amigos, conselheiros e orientadores José Mauro Volkmer de Castilho e Dalcidio Moraes Claudio, que nunca permitiram que o esforço esmorecesse. Trabalhos de natureza epistemológica se desenvolvem segundo um ritmo e uma forma de organização que difere dos trabalhos estritamente técnicos. A eles e ao CPGCC/UFRGS, que foi receptivo à proposta de tese e ao aval dos orientadores, devo a flexibilidade de condições institucionais de trabalho de que dispus durante esses cinco anos.

O Instituto de Informática da UFRGS e seus Departamentos de Informática Teórica e de Informática Aplicada, a Propesp/UFRGS, o CNPq, a CAPES, o CNPq/RHAE, a CAPES/COFECUB, a Finep e a SID/Informática, em épocas diversas e em diferentes formas, forneceram os vários recursos financeiros e materiais que tornaram o trabalho viável. Em especial, o apoio dos professores Ricardo Reis, Clésio Saraiva dos Santos, José Mauro Castilho, Dalcidio Claudio, Carlos Alberto Heuser, Philipe Navaux, Ingrid Porto e Rosa Maria Viccari, que durante o período da tese dirigiram organismos do Instituto de Informática e/ou a interação do II com organismos externos, permitiu que o trabalho se realizasse em condições adequadas. À Rosa Maria Viccari, pelo interesse e apoio continuados, e pela amizade, devo um agradecimento especial.

Muitas outras pessoas ouviram, comentaram, opinaram, incentivaram, criticaram e me ensinaram aspectos diversos de diferentes tópicos do trabalho, em diferentes fases do seu desenvolvimento. Quero citar especialmente:

Benedito Mello Acioly, Roberto Lins de Carvalho
Helder Coelho, Martín Hötzel Escardó
Léa Fagundes, João Carlos Gluz, Peter Greenfield
Robert Hendley, Ana Teresa de Castro Martins
Paulo Roberto Ferrari Mosca, Flávio Moreira de Oliveira
Vanderlei Moraes Rodrigues, Wagner Sanz, Aaron Sloman
Laira Vieira Toscani, Simão Sirineo Toscani
Renata Vieira, Raul Sidnei Wazlawick

e todos os alunos que, ao longo deste tempo e mesmo antes, nas mais diversas disciplinas, trabalhos individuais e dissertações de mestrado, frequentemente acompanharam o desenvolvimento de idéias cujo sentido geral eu ainda não podia lhes transmitir, pois só consegui expressá-lo no texto desta dissertação. A eles agradeço a paciência, a boa vontade, a tolerância e o interesse por essas conversas, algumas durando vários semestres.

Duas pessoas devem receber referência em separado: Benedito Mello Acioly e Martín Hötzel Escardó. A influência, apoio e participação deles na organização matemática de muitas idéias apresentadas aqui, bem como das intuições computacionais que levaram à sua formulação, foi decisiva e mesmo essencial. Que

o texto apresentado não seja formal, isso não deve levar a dúvida sobre a presença de tal organização em sua base. É evidentemente, no entanto, que o fato de eles terem colaborado com minhas formulações não os torna responsáveis, nem sequer co-responsáveis, por qualquer desmando conceitual ou teórico que eu possa ter cometido.

Quero também fazer referência à questão da epistemologia. Engenheiro de formação, fui autodidata em filosofia e epistemologia até cursar dois anos do curso de graduação em filosofia. Evidentemente, não foi suficiente para tornar-me um filósofo, nem legal nem efetivamente. Foi o suficiente, porém, para reorganizar radicalmente minha visão de mundo, flexibilizar minha atitude intelectual e principalmente, para os fins deste trabalho, dar-me o mínimo de instrumental técnico sem o qual não se pode fazer a epistemologia de qualquer ciência. Pois se é certo que no terreno da reflexão sobre fundamentos não é necessário ser filósofo profissional, é certo também que não basta ser especialista na matéria. Aos professores do IFCH/UFRGS, o reconhecimento pela minha desdogmatização ideológica e filosófica.

Finalmente, no plano pessoal, quero agradecer aos meus filhos Juliana e André e à minha esposa Maria Elisabeth, que sei o quanto precisaram abdicar e desistir, ceder e dividir; aos meus pais, a quem dedico este trabalho como reconhecimento do amor e do apoio de que nunca se cansaram; e a Fernando Linei Kunzler, que soube conduzir a nau com segurança durante a tempestade.

Sumário

Capítulo 1

Introdução

A revisão epistemológica de uma ciência consiste primeiramente no questionamento do sentido de suas noções básicas. Este trabalho se propõe as primeiras etapas de uma revisão da ciência da computação e da inteligência artificial, visando evidenciar o papel que pode cumprir em ambas a noção de inteligência de máquina.

A ciência da computação tem como as mais básicas de suas noções as de máquina, programa e computação, que se combinam na noção de máquina programável. A inteligência artificial, por seu lado, tem como noção básica, adicionalmente àquelas três, a noção de inteligência.

Procuramos fundamentar a crítica epistemológica desenvolvida neste trabalho na epistemologia genética de J. Piaget, e tomamos de sua psicologia da inteligência a noção de inteligência que adotamos – através de uma leitura que procuramos fosse sempre essencialmente computacional.

Como tal noção de inteligência está baseada nas noções de organização, regulação e adaptação, estas três se acrescentam às noções referidas anteriormente, para compor o suporte conceitual em que pode se assentar a noção de inteligência de máquina.

A procura de um papel para o conceito de inteligência de máquina dentro das duas áres de conhecimento – ciência da computação e inteligência artificial – deve começar, portanto, pela crítica sistemática das noções de máquina, programa, computação, organização de máquina, regulação de máquina, adaptação de máquina e inteligência, e pelo consequente estabelecimento de um conteúdo real, objetivo, para aquele conceito.

1.1 Objetivos do trabalho e resultados alcançados

Na proposta de tese [30], o problema de encontrar um ponto de vista pelo qual se pudesse atribuir conteúdo real ao conceito de inteligência de máquina foi caracterizado como o problema preliminar do trabalho em inteligência artificial, dentro daquilo que denominamos de enfoque realista da inteligência artificial.

A resolução desse problema preliminar é o objetivo da presente tese, a qual consiste essencialmente em uma reflexão sobre as noções referidas, e na produção de interpretações computacionais adequadas para elas. Os capítulos que seguem resumem a reflexão realizada e apresentam as interpretações produzidas.

Cabe aqui salientar os principais resultados alcançados:

1. tomando os trabalhos de Piaget como quadro conceitual de referência, definimos a inteligência de máquina como o termo final do desenvolvimento da estrutura de regulação das interações funcionais da máquina com o ambiente;

2. estabelecemos condições organizacionais que nos parecem necessárias e suficientes para que a regulação funcional possa realizar-se. Para tanto, produzimos uma reconstrução sistemática da hierarquia clássica de níveis de sistema introduzida por A. Newell [8], [84], [85], o que permitiu ultrapassar não apenas o grau de detalhamento da hierarquia original, mas também avançar sistematicamente para níveis de organização que ela não contempla;

3. estabelecemos o quadro conceitual em que as noções de estrutura, comportamento e função de uma máquina em um ambiente podem ser formalmente caracterizadas e distinguidas umas das outras, o que é necessário para o adequado tratamento das noções de organização, regulação e adaptação de máquina;

4. elucidamos a noção de teleonomia, dentro do quadro conceitual recém referido, e mostramos como a noção de desenvolvimento de máquina – tornada possível pela reconstrução sistemática da hierarquia de níveis de organização e pela explicitação do espaço da regulação de máquina – se define como a regulação da construção do fecho teleonômico da máquina. Mostramos o lugar da noção de valor na organização e funcionamento de máquina, especialmente em relação à noção de desenvolvimento de máquina;

5. revisitamos os princípios da teoria da computação, distinguindo entre teoria da calculabilidade de entidades matemáticas e teoria dos sistemas de computação. Mostramos como a chamada "tese" de Church se constitui, de fato, apenas em uma explicação da noção de algoritmo matemático e não em uma caracterização da funcionalidade total das máquinas computadoras. Para tanto, distinguimos entre máquinas calculadoras e máquinas computadoras – estas sendo capazes de operar com objetos e ações parciais (no sentido da teoria dos domínios de Scott [110]). Mostramos como só as máquinas calculadoras se enquadram na "tese" de Church e no modelo da máquina de Turing, as máquinas computadoras escapando dessa caracterização por causa de sua interatividade e de sua consequente capacidade de modificação estrutural. Definimos computação não algorítmica e mostramos as condições em que é possível reconhecer máquinas realizando

computações não algorítmicas. Estabelecemos que as computações não algorítmicas constituem o campo específico onde atua a inteligência de máquina;

6. mostramos como a teoria dos sistemas dinâmicos é instrumento adequado da análise das computações em geral, e das computações não algorítmicas em especial, e como o estudo da inteligência de máquina é, então, uma aplicação da teoria de controle às máquinas computadoras;

7. finalmente, mostramos como a inteligência de máquina tem suas raízes na arquitetura de máquina. Revisamos a noção usual de programação, mostrando como só a modificação da estrutura material da máquina é capaz de atribuir-lhe novos funcionamentos e mostramos como a inteligência artificial pode colocar-se como disciplina fundamental da ciência da computação, na medida em que puder sistematizar o trabalho em arquitetura de computadores a partir dos princípios da inteligência de máquina.

1.2 Trabalhos futuros

Esta tese restringe-se a realizar uma análise conceitual das noções fundamentais da ciência da computação e da inteligência artificial e, com isso, estabelecer os princípios do trabalho em inteligência de máquina. Ela não elabora a teoria da inteligência de máquina, nem antecipa a variedade de trabalhos experimentais que o desenvolvimento dessa estrutura pode comportar. Pensamos, portanto, que todo o trabalho propriamente sintético, relacionado à noção de inteligência de máquina, está por fazer (em particular, o tratamento detalhado da construção do nível do conhecimento, do qual só damos algumas indicações).

Por outro lado, o presente trabalho – na medida em que fomos capazes de assimilar sem distorções a epistemologia e a psicologia genéticas – se constitui na utilização da epistemologia e mesmo da psicologia de Piaget na revisão conceitual da inteligência artificial e da ciência da computação. Em um sentido, portanto, é uma aplicação da teoria de Piaget à computação. A outra via, a da aplicação da computação e da inteligência artificial à teoria de Piaget também é possível e já foi iniciada (ver [44], que propõe o termo *inteligência artificial construtivista* para designar essa segunda direção de interação).

Então, um segundo tipo de trabalho que se apresenta como trabalho futuro é o de investigar as conexões entre essas duas vias, em especial o problema de estabelecer as relações apropriadas entre a organização da inteligência de máquina e a organização dos modelos (psicológicos e outros) que podem se valer dela.

1.3 Estrutura do texto

O capítulo 2 (Duas visões da inteligência artificial) estabelece as relações, e as diferenciações, entre o trabalho apresentado aqui e os diversos trabalhos clássicos voltados ao problema dos fundamentos da inteligência artificial.

Os capítulos seguintes tratam, cada um, de uma das noções fundamentais da ciência da computação e da inteligência artificial: Máquina (cap. 3), Programa (cap. 4), Computação (cap. 5), Organização de Máquina (cap. 6), Regulação de Máquina (cap. 7), e Adaptação de Máquina (cap. 8). No conjunto, eles estabelecem os resultados obtidos no trabalho e sintetizados na seção acima.

O capítulo 9 (Inteligência de Máquina) faz as amarrações da noção de inteligência de máquina com os resultados da análise realizada nos capítulos anteriores. O apêndice A (Construção dos níveis de organização algorítmicos) apresenta a reconstrução sistemática da hierarquia de níveis de organização obtida no capítulo 6. O apêndice B (Um quadro formal para o estudo da inteligência de máquina) traz o esboço de formalização que estabelecemos para iniciar o trabalho formal em inteligência de máquina.

Capítulo 2

Duas visões da inteligência artificial

2.1 A visão artificialista

> *The goal of work in artificial intelligence is to build machines that perform tasks normally requiring human intelligence.*

Com essa frase, N. Nilsson inicia o prefácio de seu "Problem-solving Methods in Artificial Intelligence" [90], talvez o livro-texto mais importante na área quando considerado do ponto de vista histórico, por ter sido o principal livro-texto existente sobre o assunto durante quase uma década.

Ela fixa de modo emblemático o ponto de vista (i.é, o modo de pensar e o modo de fazer inteligência artificial) que denominamos de pragmatista, porque mede seu sucesso pelo sucesso prático dos programas, máquinas e sistemas contruídos com os métodos e as técnicas ditas "de inteligência artificial".

2.1.1 As abordagens pragmatista e de simulação do pensamento

É sabido que o ponto de vista pragmatista surgiu na década de 60 em oposição ao ponto de vista proposto por A. Newell e H. Simon, o da "simulação do pensamento humano" [1]. Este último colocava como objetivo do trabalho em inteligência artificial o de construir programas que realizassem tarefas mentais de maneira semelhante à maneira como elas são realizadas pelos seres humanos, para através daqueles programas estabelecer modelos da inteligência humana [2].

[1] Ver a coletânea [82], p.ex., que relembra esse aspecto, ao mesmo tempo que consagra a expressão "programação heurística" para resumir a idéia da inteligência artificial pragmatista.

[2] O artigo [86] contém a formulação original dessa proposta. [24] tenta uma reconstrução do contexto epistemológico e histórico em que ela surgiu.

É interessante notar que esta proposta tinha, explicitamente, um cunho psicológico: visava revisar as teorias psicológicas da inteligência à luz da metodologia do processamento mecânico de informações. Como consequência, porém, transformava o trabalho em inteligência artificial em caudatário da psicologia [3].

A reação pragmatista a tal proposta, da qual a frase do livro de Nilsson é uma expressão significativa, aparece sintetizada na já citada coletânea publicada por M. Minsky [82] [4]. Na introdução a esse livro, Minsky define o objetivo do trabalho em inteligência artificial como sendo o de *construir máquinas inteligentes, sem nenhum preconceito em relação a fazer o sistema simples, biológico ou humanóide* (i.é, sem preconceitos em relação a métodos e formas finais).

2.1.2 Outras abordagens

Além da abordagem de simulação do pensamento e da abordagem pragmatista, duas outras abordagens clássicas também conquistaram seu espaço na área da inteligência artificial: o logicismo, através dos trabalhos de J. McCarthy [77], e o neurofisiologismo, através das redes neurais [108] [5].

Mais recentemente – [14], [73], [51], p.ex. – as abordagens dos agentes autônomos e da vida artificial indicaram uma nova alternativa para a inteligência artificial, qual seja a de voltar-se para a ação corpórea do ser humano e para o comportamento animal, ao invés de voltar-se ao pensamento representativo.

2.1.3 A essência da visão artificialista

Em todas as correntes de idéias citadas acima, há uma marca fundamental, que as une num objetivo comum, para além das divergências metodológicas: todas se valem de uma comparação com uma inteligência natural, humana ou animal, para medirem o sucesso de suas construções.

Desse modo, a construção de máquinas, programas e sistemas é colocada numa perspectiva que podemos chamar de *substituição estrutural com preservação funcional*, isto é, numa perspectiva de substituir a entidade natural realizadora de determinada tarefa, por uma entidade artificial, que funcione de modo a que a tarefa considerada continue sendo realizada sem descontinuidade funcional, do ponto de vista de quem se vale dela.

Em resumo, independentemente das diversas intenções com que aqueles trabalhos são feitos, a intenção mais geral que termina dominando todos eles é a de buscar *uma artificialização de entidades*, isto é, de fazer com que onde antes funcionava um ser vivo, agora passe a funcionar uma máquina.

Para os fins da presente tese, importa antes de tudo reconhecer que as divergências metodológicas entre as abordagens citadas não as retiram dessa perspectiva comum, e que é essa a visão geral sintetizada na frase do livro de N.

[3] Veja-se, por exemplo, o caráter nitidamente psicológico de [87].

[4] Um retrato da reação que se seguiu dentro da área da psicologia aparece em [63].

[5] [78] é considerada a "história oficial" do surgimento da IA, quando se fixou como dominante a tendência pragmatista.

Nilsson, mesmo que a frase se refira apenas a seres humanos e a tarefas mentais. A essa visão chamamos de **visão artificialista** da inteligência artificial.

2.2 A visão naturalista

Propomos um objetivo alternativo para o trabalho em inteligência artificial, onde não aparece a idéia de artificializar entidades realizadoras de tarefas: propomos que o objetivo do trabalho em inteligência artificial seja o de, inicialmente, investigar a possibilidade de dar um conteúdo real e específico à noção de inteligência de máquina; e posteriormente, uma vez constatada a existência de tal conteúdo real, investigar as características da inteligência de máquina e sua relevância para o uso, o funcionamento, a especificação e a própria construção das máquinas. A essa visão chamamos de **visão naturalista** da inteligência artificial.

2.2.1 Inteligência de máquina como fenômeno natural

Por "conteúdo real e específico da noção de inteligência de máquina" entendemos um fenômeno real e específico que ocorre nas máquinas, assim como a inteligência humana é um fenômeno real e específico que ocorre em seres humanos. Com isso, também queremos propor que a atitude a ser tomada no trabalho em inteligência artificial com relação ao estudo da inteligência de máquina é uma atitude de investigação empírica e experimental, com forte carga de formalização, dada a natureza do trabalho de pesquisa sobre máquinas computadoras, porém não uma atitude de engenharia artificialista.

Em outros termos, a atitude artificialista em inteligência artificial é uma atitude de engenharia, visando a produção de equipamentos. Já a atitude naturalista em inteligência artificial é científica, fundamentadora da visão artificialista, visando a identificação do fenômeno da inteligência de máquina e a análise de suas características e de suas relações com o projeto, a construção e o uso das máquinas [6].

Tal como sintetizado na resenha [35], trata-se de ver a inteligência artificial como o estudo de um fenômeno natural nas máquinas, sujeito a leis específicas que precisam ser explicitadas.

A atitude naturalista implica, portanto, afastar-se da idéia dominante de que a atividade principal em inteligência artificial é o desenvolvimento técnico de um meio artificial de substituir a inteligência humana ou animal, mas por outro lado implica também poder esperar que surja entre os dois tipos de atividades a relação de cooperação que sempre surge entre qualquer engenharia e a ciência que a fundamenta.

[6] Um tema que não desenvolvemos na presente reflexão é o do sentido da noção de ciência quando aplicada a fenômenos artificiais [116]. Apenas assumimos que é possível uma atitude científica naturalista em relação ao domínio das entidades artificiais (ver também [21]).

2.2.2 A visão naturalista do uso dos computadores

Pensamos que nossa proposta não se encontra isolada no contexto da inteligência artificial. Pensamos que ela se vincula, por exemplo, à proposta que T. Winograd e F. Flores desenvolvem em [136], que contém por outro lado uma crítica do ideal da IA artificialista que consideramos definitiva.

Eles adotam três referenciais filosóficos e teóricos: a análise existencial de M. Heidegger, a epistemologia de inspiração neuro-fisiológica e imunológica de H. Maturana e F. Varela, e a teoria dos atos de fala de J. Austin e J. Searle. Seu objetivo inicial é analisar criticamente o que chamam de a "tradição racionalista", isto é, o conjunto de idéias que embasam não só a visão artificialista da IA, mas toda a visão moderna e contemporânea da ciência e da técnica.

As idéias principais da tradição racionalista são resumidamente as seguintes: conhecer é representar por meio de símbolos; aprender é construir representações simbólicas; pensar é processar representações; falar é denotar por meio da linguagem; e, conversar é trocar sentenças de uma linguagem. Em síntese, a tradição racionalista pretende a idéia de que a cognição humana se organiza em torno de representações simbólicas.

Winograd & Flores indicam como a análise do modo de ser do ser humano, elaborada por M. Heidegger, expõe a falta de fundamento da idéia de que o conhecimento humano do mundo está baseado em representações: Heidegger mostra em sua análise existencial que é o próprio existir no mundo, antes mesmo da consideração de qualquer representação, que dá ao ser humano a possibilidade de construir representações significativas, e que esse existir no mundo já é, em si mesmo, uma forma de compreensão. Portanto, é o compreender que fundamenta a representação, e não o contrário como pretende a tradição racionalista.

Por outro lado, Winograd & Flores mostram como a epistemologia de Maturana & Varela - elaborada a partir dos estudos neuro-fisiológicos que o primeiro realizou nas décadas de 60 e 70, e também dos estudos em imunologia que o segundo realizou nas décadas de 70 e 80 - obriga a rejeitar a idéia da objetividade das significações das representações, porque a significação atribuída a uma representação é relativa ao estado cognitivo, e na verdade ao estado biológico, de quem lhe atribui tal significação.

Finalmente, Winograd & Flores se valem da teoria dos atos de fala de Austin & Searle para mostrar que o conceito de linguagem como um meio de troca de significações é um conceito incompleto e que na verdade a linguagem, através dos atos de fala, é um meio de coordenação das ações dos indivíduos que conversam através dela [7].

Com essa tripla fundamentação, Winograd & Flores voltam-se para revisar as formas de uso dos computadores estabelecidas pela ciência da computação em geral e pela inteligência artificial em particular, com base na tradição racionalista. Segundo essas formas de uso, os computadores são máquinas manipuladoras de símbolos, que registram e processam dados e conhecimentos cuja significação é (idealmente) inequívoca, e que são usados para armazenar e tornar

[7] Maturana e Varela [76] também apontaram esse aspecto.

disponíveis a pessoas e organizações esses dados e conhecimentos, bem como os modos de processá-los.

A crítica da tradição racionalista mostra o quanto essas formas de uso não são gerais como pretendem. Mostra que são formas particulares que tem sua efetividade restrita a situações especiais dotadas de um certo caráter de fechamento semântico e pragmático, que viabiliza a objetividade aproximada das significações. Porém, em situações em que há uma abertura para formas inesperadas de ação e comportamento linguístico dos seres humanos envolvidos com o sistema (i.é, em situações de sistemas abertos [58], [10] e [99]), a objetividade aproximada das significações desaparece e aquelas formas de uso se mostram insuficientes.

Winograd & Flores desenvolvem então a idéia de que os computadores devem ser vistos de um modo alternativo, não como processadores de dados e de conhecimentos, mas como *ferramentas da ação humana*, especialmente como *equipamentos para a linguagem* e a *ação linguística* [136, p.76–79], isto é, como instrumentos dos atos de fala pelos quais os seres humanos organizam suas ações e seu trabalho.

Cremos que Winograd & Flores buscam o que podemos caracterizar como uma forma mais natural de pensar o modo de inserção dos computadores no conjunto da ação humana, um modo de inserção em que os computadores aparecem como eles são em si mesmos, e não como meios de simulação de fenômenos que lhes são externos. É nesse sentido que enquadramos tal proposta na visão naturalista da inteligência artificial.

A consequência metodológica final derivada por Winograd & Flores diz respeito, no entanto, à noção de projeto técnico, e a defesa dessa consequência derivada é, na verdade, o objetivo do livro. Na tradição racionalista, a idéia de projeto é a idéia de uma atividade de planejamento do agenciamento de recursos técnicos disponíveis para realizar uma construção, a qual visa resultados funcionais pré-estabelecidos. Sendo planejamento, essa atividade é realizada preliminarmente ao tal agenciamento de recursos.

Frente à crítica à tradição racionalista e frente a uma nova maneira de entender os computadores, a proposta dos autores é que se pense na idéia de projeto como a de uma atividade permanente. Porém, Winograd & Flores propõem que um projeto seja mais que um planejamento continuado de construções que se vale de uma avaliação retrospectiva destas construções para que os resultados funcionais buscados sejam continuamente renovados em função dos que vão sendo alcançados.

Os autores propõem que se pense em projeto – principalmente – como uma continuada indagação e revisão crítica do próprio posicionamento do ser humano frente àqueles resultados funcionais, visando sempre a explicitação das novas possibilidades de ação e compreensão que vão se abrindo com as diversas contruções técnicas, as quais continuamente recolocam o ser humano diante de novas perspectivas.

Para os fins da presente tese, não é necessário elaborar um posicionamento frente a tal concepção de projeto. O que é essencial observar é que essa consequência metodológica embora naturalista, no sentido em que usamos este

termo, diz respeito apenas ao modo de uso dos computadores, portanto à visão externa (ou melhor, extrínseca) que se pode ter deles, e não a um visão interna (ou melhor, intrínseca) dos mesmos.

Quanto a isso, é curioso notar que Winograd & Flores, talvez justamente por centrarem seu trabalho na análise do uso dos computadores e não nos fundamentos dos mesmos, se mantiveram estritamente aderidos à visão dos computadores como máquinas manipuladoras de representações, que é precisamente o tipo de visão que justifica a adoção da tradição racionalista não só como pano de fundo da ciência da computação e da inteligência artificial, mas também como seu fundamento teórico explícito [138] [8].

É imediato que a tradição racionalista da ênfase nas representações e a idéia do computador como processador de representações se casam perfeitamente para fundamentar a visão artificialista da inteligência artificial. Elas se implicam mutuamente nessa empreitada e, por isso, ao rejeitar-se uma é preciso rejeitar também a outra. É surpreendente ver Winograd & Flores deixarem intactas noções tradicionais da computação, tais como as de que programas são representações simbólicas de procedimentos, de que máquinas são interpretadores de estruturas simbólicas, e de que o programador determina o funcionamento do computador através dos programas que escreve, sem acrescentar a essas idéias nenhum comentário crítico.

Não podemos crer que tal situação, em que a revisão crítica permaneceu apenas parcial, tenha permitido a Winograd & Flores tirarem todas as consequências do ponto de vista que defendem. Para que tal aconteça, cremos que é preciso proceder também à parte que faltou àquela revisão, isto é, à análise crítica da visão usual dos computadores. A isso procedemos nos diversos capítulos do presente trabalho.

Percebemos portanto que nossa proposta se desenvolveu na direção deixada intocada por aqueles autores dentro de seu próprio trabalho, e é nesse sentido que dizemos que ela não está isolada no contexto da IA.

Nossa proposta não derivou, porém, da proposta de Winograd & Flores, nem visou complementá-la. Ao contrário, desenvolveu-se a partir de outras referências teóricas e filosóficas, como indicado ao longo do texto que segue, embora a hermenêutica e a análise existencial de Heidegger também estivessem presentes aqui, como parte do pano de fundo. Por outro lado, pensamos que a crítica da noção de máquina é epistemologicamente anterior à crítica do uso das máquinas, pois evidentemente condiciona esta última de modo determinante ao definir aquilo que as máquinas são e o que podem fazer [9].

Não podemos indicar, por ora, se os resultados que apresentamos são com-

[8] Por outro lado, é importante notar que também Varela [127], [128] assume essa visão simbólica da computação, embora no seu caso se trate de um imunologista e não de um cientista da computação.

[9] Em relação ao tema clássico do que os computadores podem ou não fazer [45], cremos que o trabalho de Winograd & Flores estimula a reconhecer a necessidade de uma distinção entre o problema do que os computadores *podem fazer* e o problema de como os computadores *podem ser usados*. O primeiro problema só pode ser resolvido pela análise da noção de máquina, o segundo só pela análise da ação humana em combinação com os resultados dessa primeira análise.

patíveis com os deles, nem o que é preciso fazer para compatibilizá-los, nem se é interessante fazê-lo. Apenas indicamos o que nos parece ser uma certa proximidade [10].

[10] Essa impressão foi reforçada mais recentemente pela constatação de uma certa proximidade também com os trabalhos mais recentes de F. Varela [128].

Capítulo 3

Máquina

3.1 Observações iniciais

Esta dissertação tem por objetivo estabelecer os princípios básicos da visão naturalista da inteligência artificial. Começamos aqui, pela noção de máquina. Com respeito a ela, é curioso observar nos livros-textos clássicos da teoria da computação [107], [40], [65] e mesmo em livros textos mais recentes – [12], [11], [119], [132], p.ex. –, a ausência de qualquer indagação sobre o significado da palavra máquina [1]. Nem sequer Turing [124], [125] fez esse questionamento, o que é surpreendente no caso de [125], diante do propósito declarado de analisar e definir a noção de inteligência de máquina [2].

Tentando entender tal situação a partir da idéia de ciência normal, de T. Kuhn [68], a conclusão simples e imediata que se tira é a de que a comunidade científica da computação assume que todos que se iniciam na teoria da computação já sabem o que é uma máquina e que, por isso, não há a necessidade de ensinar nada sobre tal noção.

Mas como na verdade nenhum conhecimento específico é exigido dessas pessoas em relação à noção de máquina, isso quer dizer que o que se espera de quem lida com a teoria da computação é que saiba sobre máquinas o que corresponde ao sentido comum da palavra máquina, portanto um sentido ingênuo, não crítico.

Se a situação é assim, então é porque a própria teoria da computação aceita a noção ingênua de máquina e não submete essa noção a uma crítica [3].

[1] Na introdução a [28] se relembra a herança etimológica negativa da palavra máquina: no latim como no grego antigo, o termo máquina tem um conteúdo de engano, de artimanha, de maquinação, e no período medieval as artes mecânicas eram as *artes sordidae*.

[2] No capítulo 9 mostra-se, por outro lado, que a recusa explícita de Turing em analisar a noção de inteligência se insere dentro de uma sólida tradição da inteligência artificial.

[3] Do ponto de vista epistemológico, isso significa que a teoria da computação não pode ser instrumento da crítica da noção intuitiva de máquina, justamente por estar baseada nela, e que a revisão dos princípios da inteligência artificial requer, paralelamente, a revisão dos princípios da própria teoria da computação.

Nos livros-textos de teoria dos autômatos – [5], [62], entre outros – a situação é semelhante.

Do lado da inteligência artificial também – e talvez mais espantosamente ainda – o quadro de aceitação implícita da noção de máquina do senso-comum não muda. Vejam-se [91], [137] e [106] (para citar só os livros-textos clássicos): em todos eles, aceita-se a noção de máquina tal como ela vem do senso comum, sem fazer nenhuma revisão crítica, e age-se como se a inteligência artificial não estivesse inserida numa tradição de investigação que visa revisitar tal noção, para extrair dela o que ainda não foi extraído.

Mesmo Winograd & Flores [136], como indicado na seção 2.2.2, preocupados que estão em questionar o uso dos computadores, não questionam a descrição do computador como máquina de funcionalidade exclusivamente algorítmica.

A. Newell, porém, em função de seu trabalho de análise de computadores reais [8] e de seu esforço epistemológico de análise direta e semi-formalizante das atividades práticas da IA [4], apresenta em [84] e [85] uma noção de máquina da qual tiramos elementos essenciais para a tese (ver capítulo 6).

A exceção mais notável à lista de livros de teoria da computação indicada acima é o livro de M. Minsky [81], cuja primeira seção é intitulada "O que é uma máquina?". Infelizmente, porém, o propósito de Minsky sendo o de expor a teoria clássica da computação, e não o de revisá-la críticamente, sua resposta àquela pergunta não persegue propriamente a crítica da noção de máquina.

3.2 Crítica da noção de máquina

Minsky [81], a exemplo de todos os que realizam a exposição da teoria clássica da computação, assume que essa teoria dá um retrato acabado das máquinas computadoras e que, tal como estipulado por ela, a natureza destas máquinas seria essencialmente matemática. Ele declara, por exemplo, que *algumas de nossas melhores idéias da teoria das máquinas são realmente inerentemente matemáticas* – **ou sobre a própria matemática** (p.3, grifo no original). Outro exemplo é Arbib [5], que declara que *"A teoria dos autômatos é a matemática pura da ciência da computação"* (p.4, entre aspas no original).

Ora, essa parece ser uma interpretação inadequada das idéias de Church e Turing, expostas respectivamente em [17] e [124], interpretação inadequada que Minsky compartilha com os que identificam a teoria dos sistemas de computação com a teoria da calculabilidade das funções matemáticas – desenvolvida por Church, Turing, Kleene e outros – à qual chamamos de teoria clássica da computação. A essência de tal interpretação inadequada parece ser a versão construída para a assim chamada "tese" de Church [5].

[4] Ver [101] para as noções piagetianas de análise epistemológica direta e formalizante. Ver [31] para uma tentativa de compreensão do uso da epistemologia como instrumento auxiliar da ação científica.

[5] Para evitar essa interpretação inadequada, propusemos em [26], e especialmente em [34], distinguir entre a *teoria da calculabilidade* das funções matemáticas, ou **teoria clássica da computação**, que diz respeito à calculabilidade das entidades matemáticas em geral e das funções matemáticas em particular, e a **teoria dos sistemas de computação**, que diz

3.2.1 A explicação de Church

A. Newell [84] formula de um modo muito usual a "tese" de Church [6]:

> *Church's statement is called a* thesis *because it is not susceptible of formal proof, only to the accumulation of evidence. For the claim is about ways to formalize something about the real world, i.e., the notion of machine or determinate physical mechanism. Self-evidently, formal proof applies only after formalization.* (p.150).

Porém, na verdade, em nenhuma passagem de [17] Church se refere a algum "mecanismo físico determinado", nem deixa transparecer que está falando sobre modos de formalizar "algo sobre o mundo real".

Ao contrário, Church explicitamente manifesta que seu artigo se refere a entidades e processos matemáticos, que visa exatamente *propor uma definição de calculabilidade efetiva* (p.346) de funções matemáticas, e que os métodos formais que ele indica para tanto *constituem uma caracterização tão geral dessa noção quanto é consistente com o entendimento usual intuitivo dela* (p.346).

Vê-se assim que a calculabilidade efetiva é considerada explicitamente como uma propriedade das funções matemáticas, que podem tê-la ou não, mas Church não diz, e não se vê como poderia ser assim, que uma propriedade de um ente matemático possa conter, positivamente, algo sobre um "mecanismo físico" [7].

Quer dizer, a chamada "tese" de Church não é uma tese, no sentido que esse termo tem usualmente em epistemologia, de uma proposição teórica fundamental, um axioma não-lógico, possivelmente não verificável concretamente, a partir do qual se estrutura uma teoria sobre uma realidade dada.

Ao contrário, a proposta (e não proposição!) de Church, é uma proposta metodológica consciente e deliberadamente colocada na perspectiva daquilo que R. Carnap chamou de explicação (*explication*): a troca de um conceito familiar, mas vago, por um conceito novo e exato [16, p.7]. No caso de Church, uma troca realizada no âmbito matemático, com finalidades matemáticas, sem intenção de falar "sobre o mundo real".

Foi Turing, em [124], quem falou de máquinas. No entanto, como veremos a seguir, falou delas apenas num sentido funcional, no sentido de um dos papéis que podem desempenhar quando em interação com um ser humano, e não no sentido essencial das máquinas como "mecanismos físicos" [8].

---

respeito à estrutura, comportamento e funcionalidade dos sistemas de computação.

[6] Tendo em vista que o presente trabalho diz respeito à inteligência de máquina, é importante procurar apontar o quanto a interpretação equivocada da análise de Turing e Church pode marcar a noção de computação utilizada pelos que atuam na área da IA. Só por isso tomamos essa citação de A. Newell, para exemplificar com ela a presença daquele equívoco no desenvolvimento de um dos trabalhos mais fundamentais da área. Citações semelhantes poderiam ter sido buscadas junto a outros autores, de outras áreas.

[7] Por outro lado, a "tradição racionalista" trata como "mecânica" toda manipulação simbólica envolvida nos métodos formais. Em [17], porém, essa tendência não parece estar presente. Antes, Church parece estar preocupado com a compatibilidade do formalismo com a naturalidade do que se pode atribuir ao pensamento matemático algorítmico.

[8] Turing abordou as máquinas como "mecanismos físicos determinados" em [125], no contexto de um estudo sobre a inteligência de máquina. Ver sobre isso o capítulo 9.

Consciente de que sua máquina captava o que havia de essencial na noção de algoritmo, enfatizou a idéia de que com ela, através dos programas que ela era capaz de executar, se captava formalmente o conjunto de todos os elementos matemáticos calculáveis efetivamente (números, funções, predicados, etc.), no sentido mesmo que Church atribuía à noção de efetividade. Esse esforço para demarcar o alcançável pelo seu modelo está em Turing [124] de modo muito claro, e dá a exata medida da compreensão que ele tinha de seu trabalho como estabelecimento dos "limites" da computabilidade mecânica, no sentido que se pode associar ao trabalho de Church.

Por outro lado, E. Post [103] insistiu na necessidade de ver a identificação da noção intuitiva de calculabilidade algorítmica com a noção formal de recursividade e com os modelos mecânicos de computação [9], como estando dotada de uma possível "fidelidade psicológica", no sentido de ela estabelecer uma "lei natural", uma "limitação do poder de matematização do Homo Sapiens", pois seria essa – segundo Post – a única de forma de estender tal identificação a "todas" as lógicas simbólicas e "todos" os métodos de resolução, conforme lhe pareceu ser a intenção de Church.

Cremos então que talvez tenha sido daqui, a partir de uma leitura não matemática de afirmações como estas de Post, em associação com a ênfase colocada por Turing no problema dos limites, mas extrapolando do problema da explicação da efetividade algorítmica para alcançar o "poder de matematização" e mesmo todos os "poderes mentais" do Homo Sapiens, que se popularizou a idéia, já esboçada em Turing [124], de que a "tese" de Church alcançava os limites da computabilidade efetiva em todas as suas formas possíveis, humanas ou mecânicas, com fortes repercussões filosóficas sobre os limites da inteligência artificial [10].

Assim, procurando evitar leituras apressadas dos resultados da teoria da calculabilidade efetiva, preferimos abandonar a expressão *tese de Church* sobre os limites da computabilidade efetiva, e adotamos a expressão mais neutra, mas também técnica e historicamente mais correta, de **explicação de Church** para a noção de calculabilidade efetiva.

Precisamos ainda, porém, mostrar em que a análise mecânica de Turing é de caráter apenas funcional, e não estrutural.

3.2.2 Turing e a noção funcional de máquina

Turing, em [124], faz a análise da noção de computador e considera como objeto inicial de exame um ser humano realizando um cálculo matemático. Mas, de modo algum, Turing parece pretender estar realizando a análise da estrutura

[9] No caso, o modelo de "processos combinatórios finitos" que ele propõe no artigo, e que é uma formulação de uma máquina semelhante à de Turing, mas desenvolvida independentemente por Post.

[10] Esse sentido de captar limites, presente no trabalho de Turing e reforçado por Post, parece ter passado diretamente para Kleene [65], onde a "tese" de Church é apresentada exatamente nessa versão, e provavelmente esse livro, pela sua importância, foi a principal alavanca de divulgação de tal interpretação.

intrínseca da própria natureza humana, ao analisar tal situação. Ao contrário, o uso de uma pessoa realizando a tarefa de um calculador mecânico parece ter o sentido de enfatizar que se trata justamente de uma análise funcional num sentido estrito, para a qual o que importa é o conjunto delimitado de funções realizadas na situação dada, e não a potencialidade funcional total de quem as realiza [11].

Por isso a estrutura de computador (humano) apresentada no início da análise pode ser substituída depois pela estrutura de computador (mecânico) da máquina calculadora. A rigor, portanto, a noção de máquina calculadora de Turing é uma noção funcional: trata-se de um papel funcional que componentes diversos podem desempenhar em uma dada situação concreta.

Dessa forma, parece correto dizer que Turing procedeu à análise da noção de máquina computadora apenas em relação ao seu funcionamento como máquina calculadora, e não – como parecem pretender os que fazem a identificação da teoria clássica da computação com a teoria dos sistemas de computação – à análise da máquina computadora em relação à sua capacidade funcional total.

Por isso, parece correto dizer também que a contribuição decisiva de Turing para o esclarecimento da noção de efetividade embutida na noção de algoritmo, – primeiramente estudada por Gödel [54] em razão do problema da decisão colocado por Hilbert [59], e que se constituiu na noção central da teoria da calculabilidade das funções elaborada por Church, Kleene e o próprio Turing – foi exatamente a de formular a constatação de que os funcionamentos algorítmicos podem ser reduzidos a funcionamentos de máquina, quando esta noção é tomada no sentido funcional que ele determinou.

Porém, dizer que a análise de Turing é funcional, que é relativa apenas a um particular modo de funcionamento do computador, e que assim não capta a potencialidade funcional total de uma máquina computadora, é dizer em outros termos que as máquinas computadoras são capazes de funcionamentos não algorítmicos, pelos quais então a teoria clássica da computação nunca se interessou.

Esse ponto, que é de importância central para a presente tese, requer, é claro, que se faça a distinção nítida entre funcionamento algorítmico e funcionamento não-algorítmico.

Como a caracterização positiva dos funcionamentos algorítmicos é justamente o tema da teoria clássica da computação, captada na explicação de Church, o que é preciso fazer aqui é então caracterizar o que se deve entender por funcionamento computacional não-algorítmico.

3.2.3 Funcionamentos computacionais não algorítmicos

A teoria clássica da computação costuma denominar de *oráculo* à máquina cujo funcionamento pode não ter caráter algorítmico ([12],p.ex.). Com isso, ela distingue dois tipos de funcionamentos, os funcionamentos algorítmicos e os funcionamentos oraculares, ou possivelmente não-algorítmicos.

[11] A noção de função é uma noção semanticamente bastante carregada, cuja análise permite destacar diversos sentidos implícitos. Ver seção 3.4.4.

Propomos aqui especializar um pouco mais essa distinção. Propomos subdividir a classe dos oráculos em duas subclasses: a dos oráculos operatórios e a dos oráculos não-operatórios.

Chamamos de **estrutura operatória** a um conjunto bem definido de operações com propriedades bem determinadas. Chamamos **oráculo operatório** ao oráculo cujo funcionamento se dá, a cada momento, dentro de uma estrutura opertória bem determinada.

Isto é, um oráculo é operatório quando, a cada momento, seu funcionamento ocorre através da execução de operações tiradas de um conjunto de operações bem definido, com propriedades bem determinadas, mas sem a necessidade de essa estrutura operatória ser fixa ao longo do funcionamento do oráculo.

Chamamos de **oráculo não-operatório** ao oráculo que não apresenta estrutura operatória determinada em algum dos momentos de seu funcionamento [12].

Com isso, distinguimos três classes de possíveis funcionamentos de uma máquina: os algorítmicos, os oraculares operatórios e os oraculares não-operatórios. A teoria clássica da computação rejeita o caráter efetivo aos funcionamentos das duas últimas classes e só admite como propriamente computacionais os funcionamentos algorítmicos.

Aqui, propomos que esse ponto de vista seja modificado, que só os funcionamentos não-operatórios sejam considerados não-computacionais, e que tanto os funcionamentos algorítmicos quanto os oraculares operatórios sejam considerados efetivos. Por outro lado, propomos conservar a designação *algorítmicos* para os funcionamentos assim caracterizados na teoria clássica da computação, e designar de *não algorímicos* os funcionamentos oraculares.

O resultado dessa classificação se exprime nos seguintes tipos possíveis de funcionamentos de uma máquina:

funcionamentos computacionais algorítmicos (ou funcionamentos algorítmicos, simplesmente)

funcionamentos computacionais não-algorítmicos (ou funcionamentos não-algorítmicos, simplesmente)

funcionamentos não-computacionais

A principal diferença entre funcionamentos computacionais algorítmicos e não-algorítmicos é, então, a de que os algorítmicos se dão dentro de uma estrutura operatória fixa, ao passo que os não-algorítmicos se dão dentro de uma estrutura operatória variável, isto é, uma estrutura cujo conjunto de operações

[12] O termo estrutura operatória tem em Piaget, [96] p.ex., um sentido pré-formal de estrutura algébrica. Conservamos, por enquanto, o uso pré-formal do termo operatório, reservando o termo algébrico para uma formalização posterior das idéias apresentadas aqui. Evidentemente, fica a questão de se o sistema cognitivo do ser humano se qualifica, ou não, como estrutura operatória bem determinada, no sentido exigido por essa definição. Na medida em que o fizer, a inteligência humana estará mais próxima da inteligência de máquina, tal como ela é definida nesta tese.

e propriedades varia ao longo da execução de suas operações [13]. Os funcionamentos computacionais não-algorítmicos, isto é, funcionamentos que se dão em uma estrutura operatória variável, consitituem então – como se verá –, o resultado direto da existência e da atuação da inteligência de máquina. Eles abrem a possibilidade da noção de desenvolvimento de máquina, discutida na seção 3.4.7 e no capítulo 9 (ver também [27]) [14].

A possibilidade dessa existência concreta de funcionamentos não algorítmicos em computadores reais também é apontada nas seções 4.2 e 5.4.1. O que importa notar aqui, entretanto, é que o reconhecimento teórico dessa existência só é logicamente possível a partir da conclusão obtida acima de que a análise conceitual de Turing tem um caráter funcional e não um caráter estrutural (ver [34] para uma primeira discussão).

Na presente tese, para adotar uma terminologia facilitadora para os dois sentidos funcionais possíveis para a palavra computador, fixamos que os termos **máquina calculadora** e *calculadora automática* são usados para designar o computador quando ele é considerado executando um procedimento de características algorítmicas, e que os termos *computador* e **máquina computadora** são usados para designar o computador quando ele é considerado em relação à sua capacidade funcional total, incluindo a possibilidade de executação de procedimentos não-algorítmicos.

3.2.4 As duas limitações da noção usual de máquina

Ao reconhecer a diferença entre os dois sentidos possíveis para o termo computador (*máquina calculadora* e *máquina computadora*), salta aos olhos o engano contido na afirmação paradigmática de Minsky sobre a natureza matemática das máquinas computadoras, pois ela implica em essas máquinas terem uma funcionalidade restrita à funcionalidade de uma máquina calculadora. Isto é, a afirmação faz as máquinas computadoras perderem parte significativa de seu conteúdo concreto simplesmente por que um dos seus possíveis modos de funcionar é um determinado modo de funcionamento, o funcionamento algorítmico.

Por outro lado, na segunda seção do livro [81], seção intitulada *Sobre definições*, Minsky declara que a máquina *não pode ser definida de modo útil como um membro de uma certa classe de objetos físicos* (p.3), isto é, que o conceito de máquina não pode ser definido "de modo útil" por gênero e diferença específica.

A razão para tanto é uma outra idéia de senso comum, que Minsky também adota paradigmaticamente: que a decisão sobre se algo é ou não uma máquina dependeria *do para que ela é usada, e não só de sua composição ou estrutura*

[13] Que os procedimentos computacionais a estruturas operatórias variáveis não se reduzem a procedimentos computacionais a estruturas operatórias fixas, é uma questão essencial, tratada no capítulo 5.

[14] A consideração de funcionamentos dentro de estruturas operatórias variáveis no tempo, coloca em pauta obrigatoriamente a questão clássica da dualidade do *tempo objetivo* (físico, newtoniano, reversível), e do *tempo subjetivo* (biológico, bergsoniano, irreversível), questão levantada por N. Wiener em relação às máquinas já em [134, cap.I], mas a partir da consideração da existência de processos estocásticos em máquinas complexas, e não por razões construtivistas, como na presente tese (ver capítulo 5).

(p.3). Isto significa a idéia de que uma máquina não é só uma estrutura, uma dinâmica e uma funcionalidade, mas é também uma parte da ação humana, de forma que a decisão sobre se algo é ou não uma máquina depende de uma análise da própria ação humana.

Com toda essa carga semântica, e mais a carga matemática indicada anteriormente, é claro que a noção de máquina tem de se tornar excessivamente complexa para poder receber uma definição "útil", e nesse sentido o ponto de vista adotado por Minsky não oferece alternativa a não ser a de considerar que *uma definição intuitivamente adequada teria de ser muito complicada* (p.3).

Desenvolvemos a respeito da noção de máquina um ponto de vista crítico que resultou ser antagônico aos expostos acima, em um certo sentido. Cremos poder dar uma definição adequada e útil da noção de máquina. Uma definição que permita superar as duas limitações citadas – a da restrição aos funcionamentos algorítmicos, e a da dependência de um uso exterior. Uma definição que – por isso mesmo – serve melhor para colocar a natureza das máquinas, e suas relações com a ação humana, numa perspectiva teórica essencial e não meramente utilitária e circunstancial (ver [22]).

3.3 Definição de máquina

Partimos de uma distinção muito conhecida, que foi explicitada com clareza por P. Latil em um texto clássico da literatura cibernética [70]. Na seção intitulada "Que é ferramenta? Que é máquina?", Latil estabelece que uma ferramenta é um artefato que prolonga a ação do homem que o utiliza, ao passo que uma máquina executa ela própria a ação visada, quando dispõe da energia adequada.

Nota-se que há também, nessa conceituação, uma referência ostensiva ao uso da máquina, à sua inserção no conjunto da ação humana, mas o aspecto importante dela é a referência à autonomia da máquina na execução de seu funcionamento.

Com base nisso, estabelecemos:

> **Máquina** *é o artefato dotado de autonomia operacional.*

Esta definição apresenta as seguintes características, que são comentadas na próxima seção [15]:

1. a definição se dá por gênero e diferença específica;
2. o termo artefato está sendo usado no sentido de objeto construído pelo homem segundo um projeto consciente;

[15] As definições de máquina, programa e inteligência de máquina que apresentamos neste e nos próximos capítulos foram formuladas primeiramente na proposta de tese [30]. Na proposta de tese [30] usamos o termo *autonomia funcional* para definir máquina. Porém, com a elaboração de [36] e [37] esclarecemos as noções de função e de operação (tal como mostrado na seção 3.4.4), de modo que a formulação adequada da noção de máquina é a adotada aqui, utilizando o termo *autonomia operacional*.

3. o termo operacional implica a realização temporal de ações e a produção de resultados;

4. o termo autonomia indica que as formas de funcionamento de uma dada máquina são determinadas por ela própria, e não por um agente externo qualquer que lhas impõe sem resistência; indica também que as máquinas não funcionam isoladamente, mas sim que estão colocadas em um ambiente e que funcionam em interação com ele;

5. a definição deixa em aberto o que chamamos de questão da *autonomia axiológica*, isto é, a questão de cada máquina poder ou não desenvolver valores e normas particulares para decidir sobre a conveniência e a finalidade de sua ação;

6. a definição deixa em aberto também a questão da *autopoiese* [76] ou da autonomia energética e material, isto é, a questão de as máquinas terem ou não meios próprios para conseguirem no seu ambiente os elementos necessários para suprir suas necessidades de energia e para reporem o desgaste material devido aos seus funcionamentos;

7. a definição não faz referência explícita a processos de desenvolvimento, que no entanto se explicitam durante a análise dos aspectos referidos anteriormente.

3.4 Comentários à definição de máquina

Sobre as características da definição de máquina apresentada acima podemos fazer os seguintes comentários:

3.4.1 Finalidade e artificialidade

O fato de a definição não fazer referência à finalidade da máquina – ao para que ela é usada, e ao quanto ela atende a essa finalidade – possibilita simplificar exatamente o que Minsky [81] considerava complicado: decidir sobre se algo é ou não é uma máquina apenas com base no exame desse algo em si mesmo, independentemente de seu modo de inserção no conjunto das ações humanas.

A definição dada troca a referência à finalidade – que não diz respeito exclusivamente à máquina, pois refere também quem a usa – pela referência à autonomia do dinamismo, que diz respeito exclusivamente à máquina e portanto à sua classe e diferença específica.

A referência à finalidade da máquina, deixando de ser intrínseca e essencial para ser extrínseca e conjuntural, passa a poder ser considerada também mais objetivamente, pois então a variação da finalidade conforme o contexto e o modo de uso não pode mais afetar a classificação dos objetos como máquina, e a estabilidade de uma classificação é condição de sua objetividade.

Por outro lado, a exigência da utilidade da definição, colocada explicitamente por Minsky, pode ser entendida em dois sentidos: o primeiro, teórico, relativo à

utilização da definição de máquina na elaboração de uma teoria das máquinas e de sua inteligência; o segundo, prático, relativo à utilização da definição de máquina no estudo do papel das máquinas na ação humana.

A verificação da utilidade teórica da definição só pode ser feita após a elaboração da teoria que se baseia nela (referida na seção 1.2), portanto, não pode ser antecipada neste ponto da pesquisa. O segundo sentido, relativo à verificação da sua utilização no estudo da compreensão do papel das máquinas na ação humana, não cai no escopo do presente trabalho, embora esteja intimamente relacionado com ele (ver seção 2.2.2 e [22]).

Em nenhum desses caso a utilidade da definição de máquina depende em princípio da inclusão nela de uma referência ao uso prático da máquina na ação humana. No nosso caso, como se verá nos capítulos que seguem, a exclusão da referência ao uso prático da máquina sugere a utilidade teórica da definição, que é o que interessa para a fundamentação da inteligência artificial.

Quanto à forma da definição propriamente dita, por gênero (artefato) e diferença específica (autonomia operacional), ela supõe que esteja dada, de pleno, a definição de artefato. Por gênero e diferença específica, contudo, a noção de artefato só pode ser alcançada, no conjunto das coisas existente, por contraposição à noção de objeto natural, o que também foge ao escopo deste trabalho [16].

O problema imediato da presente tese passa ser então o da distinção entre autônomia e heterônomia no campo da operacionalidade dos objetos artificiais. Felizmente, quanto a isso, podemos nos socorrer de J. Piaget [99] e F. Varela [127] e sua noções de *sistemas aberto-fechados* e de *fechos operacionais*, respectivamente [17]. É o que faremos em diversas passagens, no que segue, especialmente na seção 3.4.4.

3.4.2 Transparência

A existência do projeto garante a transparência relativa do objeto projetado, isto é, a possibilidade de uma descrição intuitivamente satisfatória e completa dele. Uma máquina, portanto, não deve conter mistérios.

Nem por isso, uma descrição precisa e exaustiva dela deve ser necessariamente possível. Em [29] tentamos uma reflexão sobre a insolubilidade do problema da parada, que pretendeu colocar aquela impossibilidade de descrição precisa e exaustiva como resultante não de uma limitação dos formalismos computacionais ou da própria noção de procedimento efetivo, como é usual (ver [69],p.ex.), mas sim como resultante da natureza mesma das máquinas, pela carência essencial de uma transparência absoluta em sua constituição.

A implicação epistemológica dessa interpretação que atribui uma essencial falta de transparência às máquinas parece ser a seguinte: não só em relação aos

[16] Quanto a isso, estabelecemos apenas que um objeto é artificial na medida em que sua elaboração se realiza com a intervenção continuada ou da ação humana intencional ou de outros artefatos, e ele é natural na medida inversa. Isto é, *natural* e *artificial* são distinções graduais e não, categóricas.

[17] Ver [33] para uma primeira elaboração, baseada em [99].

fenômenos humanos, mas também em relação aos fenômenos que ocorrem em quaisquer sistemas complexos – mesmo mecânicos – abertos ao funcionamento autônomo, não é o paradigma da explicação reducionista que se aplica, mas o paradigma da compreensão globalizadora e histórica [120].

Por outro lado, a existência do projeto pode dar a sensação de que o problema da teleologia se resolve à partida, pela presença da ação do projetista comandando a articulação dos meios e fins.

Pensamos que essa sensação é enganosa, por dois motivos. Em primeiro lugar porque, se ela parece resolver o problema da origem das articulações de meios e fins em artefatos que não apresentam desenvolvimento autônomo, não resolve esse problema tão logo o desenvolvimento autônomo se torna uma possibilidade dos artefatos.

Em segundo lugar porque, mesmo no caso que exclui a possibilidade do desenvolvimento autônomo de artefatos, não se deve confundir a leitura que o projetista faz da estrutura que criou, desde um ponto de vista extrínseco e relativo ao uso que pretendia dar a ela, com a leitura que o analista pode fazer, na medida em que está livre de usos pretendidos para a estrutura, podendo assim voltar-se para um ponto de vista intrínseco relativo apenas à estrutura e ao funcionamento dela [18].

Daí, exatamente porque os funcionamentos possíveis de um sistema complexo são em geral múltiplos, a visão do projetista termina sendo uma entre muitas possíveis (e frequentemente não antecipáveis por ele mesmo, mesmo que relativas ao sistema que ele próprio projetou!), e só a visão intrínseca do analista pode abarcar o conjunto dos funcionamentos possíveis.

Só que, nessas circunstâncias, as dificuldades com as causas finais se repõe pelo menos parcialmente, pois trata-se novamente de explicar – não mais geneticamente, dado que se supõe a ausência de desenvolvimento, mas operacionalmente – um todo dinâmico coerente que mantém sua integridade operacional ao longo do tempo, e que pode admitir variadas formas funcionais, mesmo que tiradas de um conjunto fixo.

3.4.3 Efetividade

A noção de efetividade é central para a compreensão do conceito de máquina computadora, pois engloba duas características fundamentais desse conceito: finitarismo e produtividade.

Por **finitarismo** entendemos a condição de que as máquinas só se valem, em cada instante dos seus funcionamentos, de um conjunto finito de recursos internos ou externos – mas não a condição de que os próprios funcionamentos sejam necessariamente finitos no tempo ou no espaço.

Por **produtividade** entendemos a condição de que as máquinas produzem seus resultados ao longo do seu funcionamento, partes finitas dos resultados sendo produzidas durante tempos finitos de funcionamentos – mas não a condição

[18] Isso procuraremos demonstrar na seção 4.4, em concordância aparente com Maturana & Varela [76].

de que os próprios resultados sejam necessariamente finitos no espaço e no tempo.

Tradicionalmente ([40], p.ex.), as condições de finitarismo e produtividade são fundidas simplificadamente em uma só condição de efetividade, que chamamos aqui de **exigência de parada**, pela qual uma máquina só é capaz de produzir seus resultados após ter terminado o seu funcionamento [19]. É a exigência de parada que dá origem ao *problema da parada* referido na 3.4.2.

Sem a exigência de parada, quer dizer, quando uma máquina pode ser produtiva sem precisar terminar sua computação, a questão da parada se torna, em geral, secundária, e nas máquinas cujo funcionamento não deve parar, isto é, que devem se manter *vivas* no sentido da teoria da concorrência ([53], p.ex), a parada se torna precisamente uma condição a evitar.

A exigência de parada surge de uma concepção simplificadora do poder de interação das máquinas com seus meios externos, pela qual uma máquina só é capaz de intercambiar objetos e ações acabados, completos, e não é capaz de intercambiar objetos e ações parciais, i.é, parcialmente elaborados. Chamamos a essa concepção de **exigência de totalidade**, e se vê que ela pode ser compreendida como a razão da exigência de parada.

A compreensão de que a exigência de totalidade em relação aos objetos e ações intercambiados pela máquina com seu ambiente não é uma exigência realística, e de que na verdade as máquinas funcionam em uma condição de intercâmbio de objetos e ações parciais, é um passo essencial para a compreensão do papel da *interação* na organização e funcionamento das máquinas computadoras [20].

Por outro lado, um deslocamento conceitual importante que se faz ao se passar da noção de efetividade submetida às exigências de totalidade e de parada para a noção de efetividade como *produtividade + finitarismo*, é o de deixar em aberto a direção da produtividade da máquina. Na primeira situação, a direção é a de uma produção para o exterior – portanto na perspectiva extrínseca da utilização da máquina por um usuário externo. Já na segunda situação, abre-se a possibilidade de a produção da máquina ser destinada a ela mesma, através dos processos de interação interna de seus componentes – portanto numa perspectiva intrínseca de auto-conservação e desenvolvimento.

É de esperar que a abolição da condição de parada tenha repercussão importante sobre a noção de efetividade, com uma ampliação da sua significação e uma correpondente ampliação da noção de computabilidade algorítmica, mas aqui só nos ocuparemos de suas contribuições para a noção de computabilidade não-algorítmica [21].

[19] Rogers [107], mesmo distinguindo entre parada e produtividade (p.12), adota esta fusão simplificadora.

[20] O movimento inicial na direção dessa compreensão foi feito por Scott [110] ao fornecer uma análise do conceito de máquina que explicitava a existência e manipulação interna de objetos parciais em máquinas calculadoras. Para uma primeira formulação da definição de interação e de seus impactos sobre os funcionamentos das máquinas ver [34] e [33]. Ver também [48], [49] e os capítulos 6 e 7, a seguir.

[21] Para o impacto sobre a noção de computabilidade algorítmica ver [49], [50].

3.4.4 Funcionalidade e teleonomia

O fato de uma máquina ser um sistema dinâmico [22] dotado de uma organização complexa aproxima a idéia de máquina da idéia de organismo biológico e leva necessariamente a pensar os componentes das máquinas e suas ações em termos de finalidade, isto é, da finalidade que cumprem no sistema total da máquina.

A aproximação entre organismo e máquina não é recente [23], mas o trabalho inaugural importante para a tecnologia e ciência contemporâneas foi o artigo de Rosenblueth, Wiener e Bigelow [109], trabalho iniciador da cibernética como área de estudo autônoma (há uma análise de [109] em [1]).

As dificuldades das explicações teleológicas, desde logo reconhecidas na própria biologia, são também agudas nas ciências do artificial – principalmente em função de uma fácil e enganosa possibilidade de atribuir toda finalidade à ação do projetista –, mas não por isso devem levar, como não levam na biologia, exclusão do uso da noção de finalidade como categoria explicadora.

Se a explicação teleológica é insatisfatória por ser incompatível com a ordenação temporal exigida pela causalidade eficiente, a explicação em termos teleonômicos ([99], [83], p.ex.) é a alternativa existente para compatibilizar a noção de finalidade com a relação de causa-efeito dos processos materiais.

Na concepção teleonômica do organismo, que introduzimos a seguir, há uma dependência mútua entre estrutura, comportamento e função, com influências mútuas também sobre o desenvolvimento de cada um desses aspectos do organismo, com a função e o comportamento estando baseados na estrutura e seu funcionamento, assim como o funcionamento está baseado na estrutura e na reação do meio externo ao comportamento que se aplica sobre ele, mas com a estrutura, seu funcionamento e o comportamento voltados para a preservação da função (ver apêndice B).

Para organizar esse entrelaçamento de influências, fixamos então a seguinte terminologia (ver [36] e, principalmente, [37]):

estrutura material (de uma máquina)
: a coleção de componentes materiais da máquina, e suas interconexões [24];

funcionamento (de uma máquina)
: o conjunto dos processos que ocorrem na máquina, incluindo os estritamente internos e os que envolvem interações com o ambiente;

comportamento (de uma máquina)
: a parte do funcionamento da máquina relativa interação com o ambiente externo.

[22] Em [32] se procura uma relação direta entre os conceitos de máquina e programa e os conceitos tradicionais de sistemas dinâmicos normalmente utilizados para sistemas físicos. O capítulo 5 apresenta a elaboração posterior dessas idéias.

[23] Ver [75] para tal aproximação em Kant, p.ex.

[24] O uso do termo *componente material* aqui é análogo ao que se faz dele em física quando se fala, por exemplo em mecânica, de *ponto material*, ou em eletricidade e magnetismo, de *carga puntual*. Quer dizer, um componente material é um construto teórico determinando um jogo de propriedades operatórias, e para o qual se estabelecem correspondentes reais nas máquinas reais, em cada caso estudado (ver capítulo 6).

Com essa base terminológica estabelecemos as noções seguintes, relativamente aos componentes de uma máquina [37]:

variação estrutural (de uma máquina ou de uma parte dela, devido a um componente da máquina ou ao ambiente)
uma variação na estrutura material da máquina, devido às interações da máquina, ou de parte dela, com o componente considerado ou com o ambiente;

ligação operacional (ou, laço operacional)
ligação entre dois ou mais componentes, produzida pela realização conjunta de ações interativas, ou pelo estabelecimento de relações de causalidade entre as ações de ambos;

variação operacional (de uma parte da máquina, ou de toda a máquina devida a um componente ou ao ambiente)
variação das ligações funcionais da parte considerada, ou de toda a máquina, devido às interações que ela tem com o referido componente ou o ambiente;

papel operacional (de um componente em uma parte da máquina, ou na máquina como um todo)
o conjunto de ligações operacionais do componente com os componentes de uma parte da máquina, ou com todos os demais componentes da máquina;

estrutura operacional (de uma máquina)
a estrutura de processos dentro da qual se dá o funcionamento da máquina, conforme ele é determinado pela estrutura material da máquina em função da interação com um dado ambiente;

variação operacional (de uma máquina, ou de uma parte dela, devida a um componente)
variação na estrutura operacional da máquina ou de uma parte dela, devida às interações de seus componentes com o componente considerado;

fecho operacional (numa estrutura operacional)
uma subestrutura circular na estrutura operacional (ver [127] para a formulação original de fecho operacional) [25];

base dinâmica (de uma máquina em um determinado ambiente)
a estrutura material da máquina e a estrutura operacional que ela apresenta em um determinado momento de seu funcionamento, no ambiente considerado.

Em relação à máquina como um todo, podemos então estabelecer as seguintes noções funcionais:

[25] Como em [127] e em [99], usamos o termo circular em um sentido amplo, admitindo formas circulares múltiplas e variadas, com repetições estruturais, encaixamentos, etc.

função (realizada por um componente para uma parte da máquina, ou para a máquina como um todo)
a ação desse componente sobre a parte considerada, ou sobre o todo da máquina, isto é, o efeito dos papéis operacionais que ele desempenha para essa parte ou o todo da máquina;

ligação funcional (entre um componente e uma parte da máquina ou a máquina como um todo, devida a uma funçãoque ele realiza)
ligação entre o componente dado e os componentes existentes nessa parte da máquina, devida à funçãoconsiderada;

estrutura funcional (em geral)
um conjunto de funções e as ligações funcionais entre os componentes envolvidos em sua realização;

estrutura funcional interna (de uma máquina)
o conjunto das funções existentes na máquina e as ligações funcionais que elas determinam;

estrutura funcional externa (de uma máquina em um determinado ambiente)
o conjunto das funções que ela realiza para o ambiente e o conjunto de ligações funcionais, entre a máquina e o ambiente, que elas determinam;

estrutura funcional global (de uma máquina em um determinado ambiente)
a estrutura funcional que abarca a estrutura funcional interna e a estrutura funcional externa da máquina nesse ambiente;

fecho funcional (em uma estrutura funcional)
uma subestrutura circular na estrutura funcional;

suporte operacional (da estrutura funcional de uma máquina, pela base dinâmica da máquina, em um determinado ambiente)
a correspondência entre funções da estrutura funcional e componentes materiais da estrutura material, e entre ligações funcionais da estrutura funcional e processos na estrutura operacional, de modo que, em cada momento, para cada função exista exatamente um componente material, ou conjunto de componentes materiais, que a realize, e para cada ligação funcional exista um processo na estrutura operacional que a produza.

Outras noções funcionais importantes podem também ser formuladas, como por exemplo [37]:

dinâmica estrutural (de uma máquina)
a totalidade de suas variações estruturais;

dinâmica operacional (de uma máquina)
a totalidade de suas variações operacionais;

dinâmica funcional (de uma máquina)
a totalidade de suas variações funcionais;

função sistêmica (em uma máquina)
estrutura funcional fechada, juntamente com os componentes do sistema que a realizam;

função geral (em uma máquina)
estrutura funcional fechada, suportada por todos os componentes da máquina;

função externa (de uma máquina em relação ao seu exterior)
a função da máquina na estrutura funcional do meio em que ela está funcionando.

Como as variações estruturais, operacionais e funcionais se dão através do processo de interação dos componentes, isto é, através da execução conjunta de ações interativas, surge o problema de saber se para uma dada máquina, sua base dinâmica é suficiente para garantir suporte operacional adequado às estruturas funcionais que resultam de sua dinâmica funcional.

Isto é, surge o problema da existência ou não de um equilíbrio entre as dinâmicas estrutural, operacional e funcional da máquina, e o problema correlativo do modo de estabelecimento e conservação desse esquilíbrio. Então, definimos:

equilíbrio teleonômico (entre a estrutura funcional e a base dinâmica de uma máquina em um determinado ambiente)
a condição em que a estrutura funcional da máquina tem suporte operacional adequado e as *perturbações* (variações estruturais, operacionais ou funcionais) produzidas por algum componente da máquina ou do ambiente podem ser *compensadas* por eventuais conjuntos de outras variações (estruturais, operacionais ou funcionais), resultando na conservação da atividade global da base dinâmica e no equilíbrio teleonômico entre a nova estrutura funcional e a nova base dinâmica resultantes desse processo.

A compensação de alguma variação (estrutural, operacional ou funcional) pode não garantir a manutenção do equilíbrio teleonômico no mesmo nível que o anterior. Então definimos:

estabilidade do equilíbrio teleonômico (de uma máquina em um determinado ambiente)
a situação em que o estado de equilíbrio teleonômico da máquina é preservado frente às diversas perturbações que pode sofrer funcionando naquele ambiente.

A estabilidade do equilíbrio teleonômico é garantida pelo:

fecho teleonômico (de uma máquina frente a um ambiente determinado)
uma articulação entre a estrutura funcional e a base dinâmica da máquina, capaz de garantir a estabilidade do equilíbrio teleonômico da máquina no ambiente considerado.

Dizemos, então, que o **problema teleonômico** em máquinas consiste em determinar o quanto elas podem avançar no sentido da construção do fecho teleonômico e, portanto, da estabilidade teleonômica.

Se vê então o quanto o problema teleonômico de uma máquina está próximo do problema do projeto da máquina. Mas, o problema do projeto é um problema que se coloca do ponto de vista do uso da máquina, ao passo que o problema teleonômico se coloca desde um ponto de vista intrínseco.

Essa diferença fica ofuscada quando se trata de sistemas em que não ocorrem processos de desenvolvimento autônomo, mas fica evidente em sistemas onde ocorrem desenvolvimentos autônomo, pois aí se manifestam estruturas e funcionamentos não projetados, cujo surgimento pode afetar – positiva ou negativamente – a estabilidade teleonômica.

A proximidade do problema do projeto e do problema teleonômico de um sistema é tanta que Matura & Varela [76] e Varela [127] tratam-nos como se fossem um só, e interpretam a noção de teleonomia como a noção de funcionalidade, tal como ela aparece ao observador externo, ou ao projetista.

Por isso afirmam que a questão da teleonomia não diz respeito ao domínio de funcionamento do sistema, mas sim ao domínio de observação adotado pelo observador externo (ou, pelo projetista) do sistema. Da mesma forma, identificam função com finalidade externa, e retiram a noção de função do domínio do funcionamento.

Consideramos que tal identificação não se justifica. O ponto de vista do observador externo, e o do projetista, são pontos de vista centrados no comportamento da máquina, que é apenas parte de seu funcionamento (a parte exterior) e no uso que se pretende fazer dela, o que envolve já valores que são exteriores ao funcionamento da máquina.

A abordagem adotada aqui possibilita uma formalização da noção de função, elaborada primeiramente em [36] e [37], de forma paralela à formalização dos aspectos dinâmicos da máquina, o que torna o conceito de *fecho funcional* – estabelecido aqui – um conceito complementar do conceito de *fecho operacional* – estabelecido por Varela em [127], mas já elaborado por Piaget, em toda sua obra, em termos do primado da função de assimilação.

Finalmente, então, podemos definir com precisão a propriedade central diferenciadora das máquinas no conjunto dos artefatos e, com isso, precisar a própria noção de máquina:

autonomia operacional (de uma máquina em um determinado ambiente)
a condição em que a máquina está dotada de um fecho teleonômico frente ao ambiente considerado.

A noção de função como ação de um componente sobre uma parte da máquina ou sobre toda a máquina (que tiramos da noção de função biológica em

Piaget [99]), e a noção de teleonomia como equilíbrio estável entre a estrutura funcional e a base dinâmica da máquina (primeiramente esboçada em [27]), são noções que não apresentam as deficiências presumidas por Maturana & Varela, pois são relativas exclusivamente à totalidade da máquina, sem envolver valores não derivados dessa estrutura e de sua base.

Disso resulta que a noção de teleonomia, centrada agora no conceito de *autonomia operacional* definido aqui – derivado e complementar do conceito de *autonomia operacional* de Varela –, pode ser entendida como dizendo respeito ao jogo de articulação entre o fecho funcional e o fecho operacional dinâmico, jogo colocado numa perspectiva temporal própria pela questão do desenvolvimento [26].

Vê-se então que, tal como foi definida a noção de autonomia operacional, em termos de estabilidade de equilíbrio teleonômico, ela é uma noção limite a que as máquinas só podem aceder ou pela ação direta de um projetista externo, ou por um desenvolvimento paulatino, correlativo de um funcionamento continuado no meio.

Por outro lado, seguimos Piaget [99] reconhecendo um poder causal às estruturas funcionais de um sistema, o que coloca alguns problemas cuja solução o próprio Piaget procurou encaminhar.

Em primeiro lugar, como o plano funcional é distinto dos planos operacional e material-energético, a causalidade de uma função não pode ser a mesma que existe nesses planos. A causalidade material-energética é a causalidade física, reconhecida de há muito. A causalidade operacional é aquela chamada de *relação de precedência* e exaustivamente estudada na ciência da computação, na área da teoria da concorrência ([13] e outros).

A causalidade funcional, porém, não pode ser a física, pois não há conexões materiais-energéticas entre funções, nem a causalidade operacional, pois funções não são eventos espaço-temporais no sentido que adotamos para construir a noção de processo em máquina.

Funções, na definição que tomamos de Piaget ([99], [37] e 3.4.4) são efeitos do funcionamento de um componente sobre o funcionamento de outros componentes. Portanto, funções dizem respeito a variações em processos. Em outros termos, funções referem estruturas conjunto-teoréticas (espaço de funções, conjunto potência, etc.) de processos, e sua causalidade então tem de ser uma causalidade lógico-operatória própria.

3.4.5 Valores e normas

A definição dada exclui do conjunto das máquinas todo artefato que não seja dotado de funcionamento interno, e do conjunto dos que tem funcionamento interno, todo aquele cujo funcionamento interno seja governado por elementos

[26] Em [27] estabelecemos uma esquematização preliminar para a abordagem de noções teleonômicas e de suas relações com o desenvolvimento. Nos capítulos 6 e 7 apresentamos novos resultados desse trabalho, assim como um aprofundamento dos tópicos tratados aqui. No apêndice B, apresentamos sua vinculação à noção de inteligência de máquina.

externos. Portanto, a definição impõe um limite inferior à noção de máquina, abaixo do qual os artefatos não são considerados como máquinas.

Mas a definição deixa em aberto a questão do limite superior do conceito, de modo que, na forma como está formulado, o conceito se aplicaria a um eventual ser vivo que tivesse sido sintetizado artificialmente. Porém, a intenção que preside a formulação do conceito não pretende, em princípio, contemplar esse caso [27].

A discussão feita anteriormente em 3.4.1 a propósito do papel das finalidades na compreensão e constituição das máquinas aponta que a exclusão do caráter de ser vivo não implica a exclusão de todos os caracteres da vida.

No capítulo 9, procuramos mostrar que, além das finalidades, também os valores e as normas deles decorrentes podem ocorrer em uma máquina, e podem ser explicados por suas estruturas material, operacional e funcional, sem que seja preciso recorrer ao eventual uso que alguém possa fazer da máquina, nem muito menos a uma atribuição de "vida" à máquina. Em particular, procuramos mostrar que a autonomia operacional é o suporte de uma autonomia axiológica, necessária para a construção do fecho teleonômico.

Por outro lado, é nítida a distância que há entre as máquinas existentes hoje em dia e as máquinas dotadas de fecho teleonômico. Pensamos que, desde um ponto de vista histórico, o caminho do desenvolvimento tecnológico como um todo, e o das ciências que o fundamentam, é justamente o da realização desse fecho teleonômico das máquinas, do qual a construção da inteligência de máquina é etapa essencial [22], [21].

3.4.6 Autopoiese e autonomia

Autopoiese é o termo criado por H. Maturana e F. Varela [76] para caracterizar a forma geral da organização dos seres vivos.

Por *sistema autopoiético* entende-se aquele sistema físico-material cujo funcionamento consiste precisamente em engendrar a reposição material e energética de sua constituição física, que se desgasta precisamente por causa desse funcionamento de reposição, o que tem como resultado garantir a conservação do sistema como sistema funcionante. A tese de Maturana & Varela, então, é a de que ser um sistema autopoiético é condição necessária e suficiente para ser um sistema vivo.

Em [127], Varela destaca a estrutura formal da autopoiese, captando-a na forma de um fecho operacional, isto é, na forma de um ordenamento de operações onde se reconhece um processo recorrente, um ciclo em sentido amplo. A tese

[27] Ao caracterizar um objeto pelas palavras *artificial* e *máquina*, pretendemos excluir deliberadamente os seres vivos do conceito desse objeto. Na presente situação do desenvolvimento tecnológico não se faz necessário realizar essa exclusão explicitamente. Na eventualidade de uma síntese da vida a partir de elementos não vivos, essa exclusão deverá então ser feita explicitamente de algum modo, se for o caso de preservar a diferença entre os seres vivos e os não-vivos tal como a mantemos hoje em dia, os seres não-vivos sendo estudados pelas ciências físicas e os seres vivos pela biologia (ainda que uma biologia de seres vivos artificiais). Caso contrário, a distância entre a inteligência de máquina e a inteligência de seres vivos será reduzida.

de Varela é a de que o fecho operacional é a caracterização formal da dinâmica de todo sistema autônomo. A autopoiese, então, é um caso particular de autonomia, a autonomia no plano físico-material.

Na definição de máquina, adotamos em parte a tese de Varela, reconhecendo nas estruturas de fecho o fundamento da noção de autonomia. Porém, seguindo nossa leitura de Piaget [99], distinguimos estrutura material, estrutura operacional e estrutura funcional de um sistema, o que nos leva a postular três tipos possíveis de autonomia e três tipos respectivos de fechos:

i) **autonomia material-energética** e
fecho material-energético

ii) **autonomia operacional** e
fecho operacional

iii) **autonomia funcional** e
fecho funcional

Além disso, pensamos que há certas dependências entre esses tipos de autonomia: a autonomia material-energética exige a autonomia operacional e esta a autonomia funcional, mas as exigências inversas não se dão.

Assim, por exemplo, uma estrutura conceitual qualquer é sempre funcionalmente autônoma, embora não tenha autonomia (nem sequer existência!) operacional e material. Já um computador de von Neumann tem autonomia operacional e, portanto, autonomia funcional, mas não autonomia material-energética. Os sistemas vivos, por fim, como estabelecem Maturana & Varela, tem autonomia material-energética e, portanto, as outras duas.

Por isso, a definição de máquina está aberta à inclusão de artefatos dotados de autonomia material-energética, isto é, está aberta à inclusão de sistemas vivos artificiais, embora não seja essa sua intenção.

3.4.7 Desenvolvimento

Em relação à questão do desenvolvimento, tomamos de Piaget sua renovação da noção aristotélica de movimento: um desenvolvimento é uma transformação de formas, desde uma forma inicial até uma forma final, mas com a condição de que, ao invés de a transformação da forma se dar com uma pré-determinação teleológica da trajetória das modificações a ser seguida, ela se dá com uma liberdade de direções, mas condicionada à conservação de propriedades funcionais invariantes e a uma ampliação do campo de interações com o ambiente.

No caso do desenvolvimento de máquina, a forma transformada é a das estruturas material, operacional e funcional da máquina, os invariantes funcionais são os que forem determinados na sua organização inicial, e a ampliação do campo de interações implica a ampliação de seu equilíbrio teleonômico.

Em outros termos, definimos que **desenvolvimento de máquina** em um ambiente determinado é a variação da base dinâmica, com conservação de invariantes funcionais iniciais e ampliação do equilíbrio teleonômico da máquina no ambiente considerado.

A articulação entre o desenvolvimento e a teleonomia se dá, então, de modo imediato: *o desenvolvimento é o processo de construção do fecho teleonômico.*

Capítulo 4

Programa

4.1 Observações iniciais

Ao contrário da noção de máquina, a noção de programa constitui um tema que a ciência da computação procura analisar detalhada e formalmente. Começando já com Turing [124] e suas "descrições–padrão" de máquinas particulares – descrições que se tornam programas para serem interpretados pela máquina universal – a teoria da computação nos seus diversos livros-textos clássicos [65],[40], [107] ou posteriores [82], [11], [12], [132], [119] procurou dar um tratamento cuidadoso à noção de programa e suas relações com as noções de algoritmo e de procedimento efetivo.

Porém, em todos esses estudos, o modelo de máquina adotado foi o da *máquina calculadora* e isso influiu na noção de programa que tais estudos elaboraram, porque influiu na noção de processo de computação.

Turing [124] definiu computação de um modo que se tornaria padrão [1]: uma máquina tem configurações (estados) e pode sofrer transformações dessas configurações. As transformações decorrem de ações internas da máquina, as ações possíveis sobre uma dada configuração sendo determinadas pela própria configuração [2].

Além disso, para cada computação há uma configuração inicial e uma possível configuração final, atingida somente se a computação terminar [3].

[1] Esse modo corresponde a definí-la como uma trajetória no espaço de estados de um sistema dinâmico [72] que é a máquina. No entanto, as relações entre programas e equações diferenciais ou de diferenças finitas, parece ainda não terem sido adequadamente explicitadas (ver seção 5.4.2).

[2] Turing [124, p.232] define dois tipos de máquina: as *automatic machines*, que só admitem ações de origem interna, e as *choice machines*, que admitem a escolha externa da próxima ação que a máquina vai executar (quer dizer, de outro ponto de vista, elas são não determinísticas). Porém, ele só se valeu das *automatic machines* no artigo.

[3] É interessante notar que a exigência de parada não foi instituída por Turing. Em [124], a efetividade é uma produção continuada do resultado pois, como as máquinas que Turing estuda ali devem imprimir a expansão binária de um número real, a máquina é produtiva se envereda pelo caminho sem fim dessa impressão, e não produtiva se imprime apenas uma

O algoritmo executado pela máquina é representado por uma *tabela* que relaciona configurações com ações internas, isto é, por aquilo que se chamou posteriormente de *diagrama de estados* da unidade de controle da máquina de Turing [40].

O algoritmo executado é fixo, isto é, se mantém inalterado durante a execução da computação. Em definições de máquina que adotam a sugestão de Wang [131] de representar o programa executado pela máquina através de um fluxograma ou outra estrutura qualquer em que as intruções vão sendo apontadas pela unidade de controle, essa estrutura se torna "externa" à unidade de controle da máquina, mas o algoritmo continua fixo [4].

Então se vê que nessas definições (ou, mais claramente, nessas opções conceituais) estão consolidadas amarrações importantes da noção de máquina (já referidas no capítulo 3). São as seguintes as amarrações que fazem dessa noção de computação a realização operacional da noção de algorítmo matemático, e que fazem da máquina de Turing uma máquina calculadora [34]:

i) um estado inicial de controle, fixo, pré-determinado para todas as computações;

ii) um processo elementar de interação da máquina com seu ambiente: há uma operação inicial de entrada de dados e uma possível operação final de saída de dados, realizada se e quando a computação termina;

iii) invariância do algoritmo de funcionamento da máquina durante a computação.

A intenção de aproximar as noções de programa de computador e de algoritmo matemático, que se seguiu ao surgimento dos computadores programáveis, na década de 40, induziu na noção de programa de computador traços restritivos derivados das noções de algoritmo e de máquina calculadora elaborados pela teoria clássica da computação.

Essa teoria resultou ser plenamente satisfatória desde o ponto de vista matemático do estudo das funções calculáveis efetivamente, tal como o demonstram os livros-textos citados anteriormente.

Porém, do ponto de vista concreto da inteligência artificial, aquela aproximação com os algoritmos resultou insatisfatória. As noções usuais de programa, computação e computador contém deficiências cuja supressão é necessária para que elas possam mediar adequadamente as noções de máquina e de inteligência.

Pensamos que do ponto de vista da inteligência artificial não é possível alcançar uma análise satisfatória da máquina programável sem que se parta de

parte finita da expansão. Turing utiliza a expressão *circle-free machine* para o primeiro caso e *circular machine* para o segundo, a "circularidade" indicando tanto a parada do funcionamento quanto um funcionamento improdutivo. Por outro lado, em [65] a exigência de parada já estava consagrada, com a distinção entre *estados passivos* (ou, finais) e *estados ativos* (ou, não finais), sem que Kleene fizesse referência a essa mudança de perspectiva.

[4] Já no modelo RASP, o programa é armazenado na memória, e a possibilidade de.modificá-lo extende a capacidade da computação do modelo para além das funções recursivas clássicas, ditas *ordinárias* [46].

uma análise satisfatória da própria noção de máquina, tal como foi feito no capítulo 3. Assim, pensamos que o reconhecimento explícito da autonomia operacional das máquinas é indispensável para que se possa alcançar a idéia de variabilidade estrutural que consideramos estar implícita na noção de programa introduzida por von Neumann – tal como mostraremos a seguir – e que nos parecer ser o fundamento da inteligência de máquina.

Consideramos indispensável também reconhecer que a interatividade das máquinas com o seu ambiente é uma característica essencial delas, para poder apreciar o forte impacto de tal característica sobre o que podemos entender por procedimento efetivo.

Finalmente, pensamos que é preciso reconhecer um caráter histórico-temporal às máquinas computadoras e abandonar a idéia de um estado inicial de controle fixo e pré-determinado para todas as computações.

4.2 Crítica da noção de programa

4.2.1 A noção algorítmica de programa

O termo programa de computador pode ser usado em diversos sentidos. Em primeiro lugar, e originalmente [56], os programas foram chamados de códigos. Nesse sentido, programas são constituídos de sequências de dígitos binários armazenados na memória do computador, para serem lidos e executados como instruções pela unidade de controle. São então, concretamente, representações das sequências de operações que a unidade de controle comanda dentro do computador. Chamamos de **programas de von Neumann** a esses programas.

A essa primeira noção de programa associa-se uma segunda: a noção de programa como uma representação simplificada daquelas sequências de operações, em uma linguagem "algorítmica" – expressão do conceito elaborado pela teoria clássica da computação – e que seria uma "linguagem de alto nível", em que os detalhes da codificação estão implicitamente colocados na representação, facilitando o trabalho do programador.

Pretensamente, linguagens algorítmicas tradicionais como Fortran, Algol, Pascal e seus descendentes e variantes não seriam, então, mais que melhorias notacionais da linguagem binária efetivamente utilizada pela unidade de controle do computador. Como se verá a seguir, esse não é o caso, pois tais linguagens adotam como noção de programa a noção de algoritmo definida pela teoria clássica da computação, o que faz com que a noção usual de programa seja muito restrita quando comparada à noção de programa proposta originalmente por von Neumann para computadores reais [5].

[5]Em outros termos, dizemos que o desenvolvimento técnico e teórico da ciência da computação obscureceu aspectos da noção de programa de von Neumann que são nucleares para a inteligência artificial. Até onde percebemos, somente [46, sec.7] indicou que o modelo de computação da máquina de von Neumann tem maior competência computacional que o modelo de computação da máquina de Turing, mas aquele artigo não explorou a totalidade do potencial do modelo de von Neumann (para a noção de competência computacional, ver seção 5.4.6).

Por outro lado, tem sido usual ([7],p.ex.) fazer a crítica tanto da noção de programa de von Neumann como da noção algorítmica de programa, apontando que tais noções são dependentes da arquitetura de computador chamada de *arquitetura de von Neumann*. Em oposição a elas apresenta-se a noção de programa declarativo, que seria independente de arquiteturas, e portanto mais geral.

Pensamos que tal colocação envolve um equívoco, proveniente de uma visão empobrecedora da noção de programa de von Neumann, possivelmente induzida pela noção restritiva de programa algorítmico, e por uma visão simplificadora das relações entre máquina, programa e usuário [6].

A noção declarativa de programa pretende derivar sua generalidade do fato de centrar-se na dinâmica externa da máquina, isto é, no comportamento de entrada/saída, sem fazer referência à dinâmica interna.

Esse desvinculamento da dinâmica interna, no entanto, não é suficiente para dar ao programa declarativo um significado absoluto, como parece pretender a visão declarativista, pois o fato é que essa visão conserva, na análise de um programa, uma referência ao uso que se fará dele, e isso já relativiza tal significado.

A referência ao uso que se faz do programa aparece explicitamente quando se produz um *manual de uso* para o programa, porque todo manual de uso pode ser visto como uma *especificação da interação* do programa com o ambiente, isto é, uma especificação do comportamento que o ambiente deve ter para que o programa possa funcionar do modo previsto e, assim, adquirir o significado pretendido. Mas, então, o programa aparece nitidamente como não declarativo, pois fica explícito que seu significado é relativo a um comportamento, o comportamento do ambiente.

Uma constatação importante, portanto, é a de que enquanto a dinâmica interna da máquina admite uma descrição independente do uso que se pretende fazer dela, a dinâmica externa tende a exigir uma referência a esse uso, e essa referência é suficiente para invalidar, a priori, o ideal de uma semântica pura para linguagens de programação [7].

Toda descrição semântica de um programa (visto como representação de um processo computacional, dado em uma linguagem de programação) pressupõe um certo modo de uso do programa, e portanto não é mais semântica e sim, pragmática (para uma discussão sobre alguns desses aspectos, ver [26] e [48]), trazendo consigo todos os problemas advindos da referência ao uso, já discutidos no capítulo 3 [8].

[6] Como mostraremos a seguir, o essencial da noção de programa de von Neumann não está nem em ela ser um veículo para comunicação de algorítmos para uma máquina que vai executá-los, nem em ela ser uma expressão de uma particular arquitetura de máquina computadora – a arquitetura de von Neumann, no caso – mas sim em ela ser a expressão de uma combinação da dinâmica interna e da dinâmica externa da máquina, capaz de fazer a dinâmica interna passar por um processo de desenvolvimento.

[7] Em particular, um manual de sistema que também fosse puramente declarativo seria simplesmente inútil do ponto de vista de informar sobre *como usar* o sistema.

[8] R. Queiroz, em [104], faz uma análise wittgensteiniana da noção de significado como uso, no contexto da programação algorítmica. Para a presente tese, um aspecto importante dessa análise está em que a noção de uso que ela utiliza está despida dos aspectos finalísticos,

Entretanto, a impossibilidade de desconsiderar os aspectos de interação da máquina com o ambiente externo – devido ao seu caráter essencial para o próprio funcionamento interno – faz com que a descrição da dinâmica interna seja insuficiente como descrição da dinâmica global da máquina e, portanto, insuficiente para embasar uma noção adequada de programa.

O que é preciso fazer então, para encontrar tal noção, é encontrar uma forma de integrar as duas dinâmicas numa só, mas sem referência aos aspectos utilitários [9].

4.2.2 A noção de programa de von Neumann

Em [56], o tema é o do planejamento e o da codificação, isto é, o problema da elaboração de um código (programa binário) capaz de determinar um funcionamento interno do computador de modo que, visto externamente, tal funcionamento apareça como a realização mecânica de um método para resolução de determinado problema (em [56] o problema é sempre de cálculo numérico).

Como tal, a solução do problema da codificação e do planejamento, e os códigos resultantes, podem ser vistos sem dúvida como uma amarração entre a dinâmica interna da máquina computadora e os modos de uso pretendidos para ela – e essa consituti exatamente a essência da noção usual de programa, que [56] ajudou a instituir.

Porém, a descrição das características dinâmicas internas da máquina e do processo de execução de programas apresentadas naquele relatório [56] tem uma generalidade e uma amplitude que transcendem não só a particular aplicação visada pelos autores naquele momento, mas também a noção de programa algorítmico que se elaborou por simplificação, a partir daquela descrição mesma. Ela só encontra paralelo nas descrições das dinâmicas dos chamados programas de sistemas, especialmente sistemas operacionais e interpretadores, e nas propostas de sistemas de inteligência artificial baseadas de algum modo numa idéia de sistema aberto ([58],[14],p.ex.).

Podemos colher a síntese de tal idéia na formulação seguinte:

> "*coding is not a static process of translation, but rather the technique of providing a dynamic background to control the automatic evolution of a meaning.*" [56, p.83].

Sobre essa formulação podemos fazer as seguintes observações:

---

teleológicos, normalmente embutidos na noção corriqueira de uso. A utilização de tal método analítico para as finalidades da inteligência de máquina, no entanto, depende aparentemente de se poder enquadrar no conceito de *jogo de linguagem* um jogo em que o conjunto das regras pode mudar com o desenvolvimento do próprio jogo.

[9] Isso implica dissociar não só a dinâmica interna, mas também a dinâmica externa, de tais aspectos. É o que pensamos ter feito no capítulo 5. Quanto a isso, note-se que a noção de *uso* de uma máquina por um ambiente diz respeito ao funcionamento global do ambiente, e que o processo de interação da máquina com o ambiente pode ser tomado, por seu lado, como o elemento de ligação entre a dinâmica interna da máquina e os usos que se podem fazer dela, isto é, os funcionamentos do ambiente a que a máquina pode ser submetida.

- *translation* refere-se a uma tradução da linguagem do método de cálculo em questão para a linguagem binária da unidade de controle do computador;
- *static process of translation* pode ser entendido como um processo direto de tradução por correspondência um-a-um entre expressões do método e elementos do código;
- *background* é o próprio código, isto é a sequência de instruções binárias que é examinada pela unidade de controle e que ela usa para organizar o funcionamento da máquina;
- *automatic evolution of a meaning* é o processo de funcionamento da máquina, operando sobre as representações de informações contidas dentro dela;
- *dynamic background*, finalmente, é a indicação de que a estrutura do código não é estática, mas que se modifica ao longo da sua própria execução.

Com isso, a formulação indica também o que é um processo de computação aos olhos de von Neumann e seus colegas. Ela mostra que para eles uma computação é mais que uma sequência de transições de estados, tal como definido por Turing. Ela é na verdade o funcionamento organizado de um sistema dinâmico que tem essa característica de que seu desenrolar no tempo é controlado por um código binário, mas que também tem esse aspecto essencial de que o código pode ser transformado dinamicamente – e transformar-se a si mesmo – em função do próprio processo que controla.

Uma outra formulação dessa idéia, expressa de um modo talvez mais claro, pode ser encontrada na seguinte passagem:

> "*A coded order* [uma instrução] *stands not simply for its present contents at its present location, but more fully for any succession of passages of C through it* [C é a unidade de controle]*, in connection with any succession of modified contents to be found by C there, all of this being determined by all other orders of the sequence* [por todas as outras instruções do programa] *(in conjunction with the one now under consideration). This entire, potentially very involved, interplay of interactions evolves successively while C runs through the operations controlled and directed by these continuously changing instructions*" [56, p.82].

Que a principal operação de modificação de instruções considerada em [56] seja a de alteração dos endereços de memória referidos por elas, modificação necessária devido à falta de formas de endereçamento de memória denominados indireto ou indexado (que seriam inventados só posteriormente) – que tal fosse a principal modificação considerada e que não se considerassem explicitamente naquele relatório as modificações mais cruciais dos códigos de operações das instruções do programa (portanto, modificações da natureza das intruções) nem o acréscimo ou remoção dinâmicos de intruções – não é um fato relevante.

Conceitualmente, a descrição dada admite a priori, como legítima, qualquer modificação na estrutura dos programas, inclusive nos códigos de operações das instruções.

Pensamos que esta interpretação das idéias de von Neumann e seus colegas pode ser justificada pelo seguinte:

- em [55] (seção 7: *The input-output organ*) é admitida a modificação dinâmica de instruções pela intervenção de um operador externo, isto é, a modificação de instruções durante uma parada forçada da execução do programa;
- em [56] (seção 7: *General principles of coding and flow-diagramming*) define-se explicitamente uma forma de conexão variável entre partes de programas, chamada *variable remote connection*, para indicar ao nível das descrições de fluxogramas a modificação de instruções de desvio;
- em [15, p.78] prevê-se a futura utilização de teletipos para facilitar a intervenção direta do operador na execução do programa;
- e, principalmente, em [55] (seção 8: *The memory organ*) admite-se a ambiguidade dados/instruções/endereços, isto é, a possibilidade de uma mesma sequência de dígitos binários ser interpretada ou como um dado, ou como um código de operação, ou como um endereço de memória – dependendo do momento do ciclo de funcionamento da unidade de controle C em que ela é examinada. Em consequência apresenta-se a noção de *programa armazenado* como a característica mais importante da arquitetura proposta [10].

Como o conjunto dessas condições torna imprevisível, em princípio, o resultado de uma alteração de programa, posto que ela pode ser consituída em parte por uma combinação de ações de origem externa, não há como negar ser parte integrante da noção original de programa de computador a possibilidade conceitual de os programas poderem sofrer qualquer tipo de alteração.

Essa possibilidade não deve ser vista como uma casualidade técnica da arquitetura de von Neumann, a ser evitada com o uso das "boas" técnicas de programação que não permitem a modificação dinâmica de programas, buscando com isso uma maior facilidade de depuraçãoe de verificação formal de sua correção.

O ponto de vista da depuração e da verificação da correção é o ponto de vista do ajustamento do funcionamento da máquina a um uso pretendido. Todo elemento perturbador ou complicador da depuração é um elemento que opera contra a segurança da correção do uso, por isso é tecnicamente secundário e mesmo prejudicial, para esse ponto de vista.

[10] É interessante notar que essa ambiguidade já está presente na máquina de Turing. von Neumann se valeu dela, certamente, para conceber a noção de programa armazenado, mas introduziu também a interatividade, que a máquina de Turing não tem.

Pensamos que não é esse o conteúdo fundamental da noção de programa de von Neumann. Apesar de von Neumann também estar preocupado com um uso do computador (cálculos numéricos) e ter cuidados com a "correção" dos programas [66], a presença de um dinamismo amplo, envolvendo a própria modificação dos programas, é claramente aceita em princípio e utilizada instrumentalmente de modo positivo, para além do fato de representar um modo de superar limitações de uma arquitetura ainda primitiva.

Isso pode ser observado com toda a clareza na seguinte passagem:

> [the machine] *can change the contents of the memory – indeed this is its normal* modus operandi. *Hence it can, in particular, change the orders (since these are in the memory !) – the very orders that control its actions. Thus all sorts of sophisticated order-systems* [programas] *become possible, which keep successively modifying themselves and hence also the computational processes that are likewise under their control.* [130, p.20]

Portanto, um programa de von Neumann é uma estrutura essencialmente dinâmica, como a estrutura do processo que ele determina.

4.2.3 Sistemas físicos de símbolos

Em [88], A. Newell e H. Simon introduziram a noção de sistemas físicos de símbolos para captar o essencial de uma estrutura de máquina computadora. Em [84], Newell realizou uma análise mais detalhada do conceito, apresentando os sistemas físicos de símbolos como constituindo o *nível de programa* da hierarquia de níveis dos sistemas de computadores que ele estabeleceu em [8].

Podemos entender, então, que a síntese da noção de sistema físico de símbolos é a de que ela corresponde à noção de máquina com arquitetura de von Neumann. Claro, não diretamente nos termos em que essa expressão costuma ser usada [11]. É para além dos detalhes da arquitetura de von Neumann que as duas noções se aproximam, e com certeza essa aproximação está no espírito do artigo de Newell, através da noção genérica de sistema físico de símbolos [84, p.154].

A aproximação se dá nas duas característica essenciais da arquitetura de von Neumann, da qual derivou a arquitetura da linguagem e da máquina Lisp: *programa armazenado*, e *interatividade* durante a execução dos programas [12].

Com isso, amarramos a noção de programa de computador de von Neumann à noção de estrutura física de símbolos, e amarramos a noção de sistema

[11] Por exemplo, o sistema físico de símbolos paradigmático, chamado `SS`, descrito em [84] é em essência uma variação de uma arquitetura de máquina Lisp (*a garden variety, Lisp-ish sort of beast*, p.177), e sabe-se que Lisp é uma linguagem com um núcleo que segue o paradigma de programação funcional, ao contrário da linguagem da arquitetura de von Neumann, que segue o paradigma de programação imperativa.

[12] A rigor, pela definição original de sistema físico de símbolos, a aproximação deveria se dar pela idéia de *universalidade* computacional, que diz respeito às características de comportamento externo, mas deixamos a questão da universalidade para tratar no capítulo 5 e adotamos agora a aproximação pelas características do funcionamento interno.

físico de símbolos às duas propriedade que consideramos essenciais: programa armazenado (com ambiguidade dados/instruções/endereços) e interatividade do processamento.

Cremos então que, para alcançar o limite das possibilidades conceituais criadas pela arquitetura de von Neumann e pelos sistemas físicos de símbolos, devemos considerar que os programas de máquinas computadoras se caracterizam por serem estruturas operacionais capazes de auto-modificação e desenvolvimento, por influência de fatores internos e externos à máquina, mas sempre de acordo com a articulação global de suas partes (*all of this being determined by all other orders of the sequence*) [13].

Por isso, pensamos que a definição operacional de programa que apresentamos a seguir capta o que a noção original de programa dada por von Neumann e a noção de sistema físico simbólico tem de mais fundamental relativamente ao funcionamento de máquinas computadoras reais, para além da existência dessa contraparte física que são os programas simbólicos [14].

4.3 Definição de programa

Partimos da idéia já citada anteriormente: *um programa é o background dinâmico que controla a evolução automática de um significado* [56, p.83]. Porém, interpretamos os termos dessa caracterização de modo a poder utilizá-la como mediadora entre a noção de máquina, dada na seção 3.3, e a noção de inteligência de máquina a ser dada na capítulo 9.

Como dito antes, entendemos que nessa frase a expressão *background dinâmico* designa a própria estrutura mutável do programa simbólico interpretado pela unidade de controle do computador de von Neumann. Na Introdução a [28], porém, desenvolvemos a idéia de que os programas simbólicos executáveis em uma máquina são apenas *índices de funcionamentos* que são pré-formados nessa máquina pela sua estrutura material e, portanto, não são determinações exógenas controladoras desses funcionamentos. Por isso, na definição abaixo, desenfatizamos o índice (o programa simbólico) e enfatizamos o conteúdo (o funcionamento) [15].

Consoante o que foi dito na seção 3.4.7 sobre desenvolvimento, por *evolução de um significado* entendemos o desenrolar temporal de operações de uma estrutura operacional, mas um desenrolar aberto a transformações daquela própria

[13] Em [34], com base em [111], chamamos de **programas parciais** a tais estruturas (ver seção 4.4.2).

[14] A principal consequência dessas amarrações conceituais é a seguinte: a natureza simbólica física dos programas e a interatividade das máquinas são a base sobre a qual se assenta a dinâmica dos estados de atividade (comentada na próxima seção e estudada no capítulo 7), e as estruturas operacionais dessa dinâmica constituem exatamente as estruturas reguladoras das trocas funcionais da máquina com o ambiente, nas quais podem se enraizar eventuais estruturas operatórias da inteligência de máquina (tal como discutido no capítulo 9).

[15] Notamos então que, em consequência, temos duas noções de programa: o *programa físico simbólico* (ou programa simbólico, simplesmente) que consitui um particular estado ou arranjo da estrutura material física da máquina, e o *programa* propriamente dito, no sentido que definiremos abaixo, que determina as possibilidades operacionais globais da máquina.

estrutura. Por outro lado, por *evolução automática de um significado*, em função do reconhecimento da autonomia operacional da máquina, entendemos a evolução autônoma no sentido que a palavra autonomia recebeu na seção 3.4.6.

Também, tornamos essencial à noção de programa a idéia de que a máquina que o executa é interativa e que portanto o programa admite em princípio a execução de ações externas ao longo de todo o seu processamento, e não só no início e fim deste, assim como consideramos que ele pode indicar trocas com o ambiente, seja de dados sobre os quais operar, seja de conjuntos de instruções a executar, e que como considerado na seção 3.2.3 esses dados e conjuntos de instruções podem ser parciais.

Além disso, deixamos em aberto a possibilidade de o programa poder indicar trocas de elementos materiais e energéticos com o meio, seja no sentido da autopoiese, exposto na seção 3.4.6, seja no sentido da mera variação da estrutura material da máquina.

Assim, adotamos a seguinte definição:

> **Programa** *é um conjunto organizado de ações executadas por uma máquina.*

Esta definição tem as seguintes características, que serão comentadas na próxima seção:

1. a palavra organização está sendo usada no sentido de estrutura operacional, indicando que entre as ações se estabelece uma relação de causalidade operacional;

2. o fato de as ações serem ações de uma máquina indica que elas podem ser tanto ações internas como trocas externas da máquina com o ambiente;

3. a palavra conjunto está sendo usada num sentido informal: a coleção de ações que constitui a estrutura operacional de um programa não é necessariamente fixa, e o fato de o conjunto de ações não ser fixo se acompanha do fato de a própria estrutura operacional não ser fixa.

4.4 Comentários à definição de programa

4.4.1 Estrutura operacional global e estados de atividade

A definição diz que as ações são elementos que passam por uma execução. Ela pressupõe, portanto, uma diferença entre uma ação e um evento que é uma particular ocorrência dessa ação.

Nas chamadas teorias da concorrência ([80], [105], p.ex.) chama-se *relação de causalidade* a uma certa vinculação entre ações que dá às ocorrências dessas ações a estrutura de uma ordem parcial, constituindo o que se denomina de processo . É exatamente essa noção de causalidade que chamamos de **causalidade**

operacional e, portanto, nossa noção de programa coincide em parte com o que aquelas teorias chamam de processo [60],[80] ou de sistema [105] [16].

A **estrutura operacional** de um programa é, portanto, a estrutura de eventos [89] ou de ocorrências [105] que esse programa propicia que ocorra na máquina. Cada caminho nessa estrutura de eventos, quer dizer, cada subestrutura de eventos compatíveis entre si a cada momento, constitui um **funcionamento** da máquina admitido pelo programa.

Entretanto, por causa da interatividade da máquina, dois aspectos devem ser precisados. Por um lado, como o programa admite ações interativas, cujo resultado depende do comportamento concreto do ambiente no momento da sua ocorrência, o programa não pode antecipar em geral quais serão as consequências para a máquina de uma particular troca interativa com o ambiente.

Ele só pode antecipar o conjunto das possíveis consequências dessas trocas (ou o conjunto das consequências que ele admite que ocorram). Nesse sentido, todo programa interativo faz a máquina se comportar como uma *choice machine* de Turing.

Por outro lado, como os dados trocados pela máquina com o ambiente são dotados da ambiguidade dados/instruções/endereços, poderia parecer razoável pensar que uma máquina está executando determinado programa até o momento em que, por efeito de uma troca externa, o programa simbólico que corresponde ao programa em execução indica uma transformação em si próprio, o que acarretaria uma transformação na estrutura de eventos que a máquina executa, e faria com que ela passasse a executar *outro* programa.

Contudo, como nenhuma transformação no programa simbólico pode ser feita sem que ele próprio a indique (ficando o ambiente apenas com a indicação de se e como ela vai ocorrer, no caso de a tranformação depender de algum modo das trocas externas) aquela maneira de pensar não parece ser totalmente adequada.

De fato, se todas as tranformações que um programa simbólico pode sofrer são determinadas por ele então já estão pré-formadas na estrutura de eventos cuja execução ele determina e, portanto, parece mais correto entender que na verdade o que ocorre na máquina quando ocorre uma transformação no programa simbólico que a controla, é uma passagem da máquina, desde uma certa *região de funcionamento* para outra, dentro de uma mesma **estrutura global de eventos**, que se conserva fixa então.

É possível entender assim, que não há propriamente sentido em falar de programas simbólicos como entidades isoladas, que vão sendo substituídos na máquina e que constituem uma "linguagem" no sentido conjunto-teorético, isto é, um conjunto sem estrutura. Do ponto de vista proposto aqui, parece mais correto pensar que o conjunto dos programas simbólicos forma o **espaço de programas simbólicos** da máquina, onde um programa simbólico está relacionado com outro se e somente se pode se transformar nesse outro.

[16] Notamos porém, que nossas transformações de programas não coincidem com as transições que "transformam" processos por realização de ações, naquelas teorias [60],[80]. Nossas transformações de programas são transformações da estrutura de causalidade, portanto transformações de sistemas de [105], e não sua realização temporal, que chamamos *funcionamento*.

Da mesma forma, podemos pensar que o que se chama usualmente de *conjunto das execuções possíveis* de um programa simbólico deve ser entendido como uma *região de funcionamento* no espaço interconectado de todas as regiões de funcionamento pelas quais a máquina pode passar dentro da sua *estrutura global de eventos*, e que a *transformação* de um programa simbólico em outro deve ser vista como um *deslocamento* entre dois pontos nesse espaço [17].

Chamamos de **estados de atividade** a essas regiões de funcionamento particulares, dotadas de uma caracterização local em termos de estruturas de eventos, como indicado acima, mas colocadas todas como *subestruturas* dentro da estrutura global de eventos, que abrange todos os funcionamentos possíveis (todos os estados de atividade possíveis) da máquina.

Cada transformação de programa simbólico é então uma **transição de estados** no **espaço de estados de atividade** da máquina [18]. Com isso, o espaço de atividades como um todo, pré-formado e determinado pela estrutura material da máquina, constitui o único e verdadeiro programa que a máquina executa. Vale então o slogan: *em cada máquina, um só programa, vários estados de atividade* [19].

Desse ponto de vista, portanto, invertemos a relação usual de determinação entre programas simbólicos e estados de atividade: usulamente se pensa que são os programas simbólicos que determinam os estados de atividade de uma máquina; agora se vê que são os estados de atividade que, ao serem observados em suas estruturas de símbolos, determinam a existência de programas simbólicos como índices caracterizadores. Desse modo, falar de programas simbólicos é só um modo simplificado e indireto de falar de estados de atividade.

Cabe, por fim, o seguinte comentário. A idéia recém exposta, corresponde ao caso da máquina real que satisfaz uma condição– usualmente assumida pelo senso comum – que é a de que o funcionamento normal não altera de modo significativo a estrutura material da máquina.

Se considerarmos o caso da máquina em que ocorre variação da estrutura material, podemos notar uma modificação no espaço de estados de atividade da máquina, com possível perda de alguns funcionamentos e acréscimos de outros. O senso comum, porém, que enxerga as máquinas apenas do ponto de vista do uso, considerará que, para além de uma possível melhoria funcional por "amaciamento" do funcionamento da máquina, todo novo funcionamento acrescentado

[17] Duas regiões de funcionamento estando conectadas se o programa simbólico que indica a execução dos funcionamentos contidos na primeira pode ser transformado no programa simbólico que indica a execução dos funcionamentos contidos na segunda.

[18] Encontramos o termo *estado de atividade* em Maturana & Varela [76], formulado intuitivamente. Tentamos aqui a indicação de uma possível expressão formal para ele. Para além de uma mera diferença no modo de falar, essa diferença entre falar da estrutura global de eventos e falar de espaço de estados de atividade expressa um aspecto concreto do funcionamento da máquina relativamente à sua estrutura funcional: cada estado de atividade admite um conjunto próprio de estruturas funcionais, internas ou externas, às quais ele pode dar suporte adequado e que não são significativas do ponto de vista da estrutura global de eventos.

[19] A primeira formulação desta idéia encontra-se na Introdução a [23]. A maneira mais imediata de encontrá-la é começar reconhecendo que um computador real, com uma estrutura de memória fixa, é um autômato finito com diagrama de estados fixo. Sloman [118] aponta a possibilidade deste enfoque, mas não o valoriza positivamente.

tem sentido degenerativo, com caráter patológico, não normal em relação ao uso pretendido da máquina.

Isso é assim porque no sentido do senso comum uma máquina não é capaz de realizar mudanças na sua estrutura material que lhe tragam novos funcionamentos de caráter normal, não patológicos. Só os seres vivos são capazes disso, as máquinas são incapazes de inovações, se diz.

Uma máquina em que ocorre degeneração da estrutura material durante seu funcionamento, assim como uma máquina capaz de aquisição energética e material, e de consequente crescimento em sua estrutura material com base nessas aquisições, certamente não obedecem ao slogan *em cada máquina, um só programa, vários estados de atividade*, pois seu espaço de estados de atividade não é fixo, varia com a estrutura material, e isso corresponde a elas executarem vários programas. Porém, só nessas condições uma máquina pode fazer variar seu programa, isto é, seu espaço de estados de atividade [20].

Em qualquer desses casos, o que cabe à teoria, inicialmente, é determinar como se dá a variação do espaço de estados de atividades em função da variação da estrutura material. O juízo sobre a qualidade positiva ou negativa dessas variações é então uma questão posterior e depende ou do uso pretendido para a máquina – portanto de uma avaliação extrínseca –, ou de uma avaliação intrínseca que considere a utilidade da variação para a própria máquina – por exemplo, seu equilíbrio teleonômico (sec. 3.4.4).

Em resumo, é só assim que uma máquina real, materialmente finita, poderia desenvolver infinitos funcionamentos: variando de modo infinito a forma de sua estrutura material finita. Mesmo assim, em nenhum momento, teria a potencialidade simultânea de infinitos funcionamentos (infinitos estados de atividade), pois uma infinidade de funcionamentos potenciais diferentes só pode ocorrer, numa estrutura finita, ao longo do tempo [21].

4.4.2 Interatividade e trajetória da atividade

O caso das máquinas em que ocorre degeneração da estrutura material não nos interessa aqui. Aqui nos interessam apenas os casos em que a estrutura material se conserva ou é aumentada.

Em relação ao caso em que a estrutura material se conserva, que pode ser visto como o caso estático da situação dinâmica mais geral de variação da estrutura material, interessa examinar como se dá o deslocamento da máquina no seu espaço de estados de atividades (ou no correspondente espaço de programas simbólicos) [22]. Em particular, interessa examinar como esse deslocamento

[20] A única maneira de continuar a pensar, nessas condições, que a máquina dispõe de um programa único e fixo é adotar a atitude platônica, por vezes comum, de considerar como elementos reais já existentes os elementos que ainda apenas são possíveis em uma situação dada. Não é o caso da presente reflexão.

[21] Com isso se vê a impossibilidade concreta de existência de uma máquina real com essa potencialidade de infinitos estados de atividade diferentes, pois para tanto ela teria de ser uma máquina eterna.

[22] Evidentemente, há um quarto caso a considerar, que é o da conservação dinâmica da estrutura material, isto é, o caso em que ela não é estática, mas estacionária, com o problema

depende de, e condiciona, a interação da máquina com o ambiente.

Em relação ao caso em que há crescimento da estrutura material, interessa examinar como esse crescimento pode ser controlado pela própria máquina e ao mesmo tempo tornar-se um crescimento positivo, isto é, um desenvolvimento e não uma degradação de sua estrutura operacional.

Nesta seção, nos ocupamos do caso estático e deixamos o caso dinâmico do desenvolvimento para a próxima seção 4.4.3 e seus seguimentos. Notamos, no entanto, que o caso dinâmico não pode ser compreendido sem a compreensão do caso estático, pois os instrumentos do desenvolvimento estrutural são em parte os do deslocamento no espaço de funcionamentos, mas aplicados à estrutura global da máquina, e não só à sua estrutura operacional.

Realizamos a seguir a análise conceitual do espaço de estados de atividade, e dos deslocamentos da máquina nesse espaço de estados de atividade, para o caso da estrutura material estática [23].

Seja uma máquina dotada de estrutura material fixa e de uma estrutura operacional global em que se distinguem regiões localizadas de funcionamento (estados de atividade) para os quais se podem identificar índices (programas simbólicos) na estrutura material. Chamamos de **transformação de programa simbólico** à transformação de índice que faz a máquina transitar de um estado de atividade para outro. Chamamos de **trajetória da atividade** da máquina ao deslocamento que ela realiza no seu espaço de estados de atividade, à medida que seu programa simbólico vai se transformando.

Chamamos de **programa simbólico parcial** ao programa simbólico capaz de indicar possíveis transformações de si próprio. Chamamos de **programa simbólico total** ao programa simbólico que não é capaz de indicar transformações de si próprio. Um programa simbólico parcial pode indicar expansões, substituições e contrações em sua própria estrutura simbólica, como se define abaixo. Um programa simbólico total não pode auto-transformar-se.

Dizemos que um programa simbólico total indica um **estado final** no espaço de estados de atividade, isto é, um estado de atividade que não pode sofrer transições. Dizemos que um programa simbólico parcial indica um **estado não final** no espaço de estados de atividade, isto é, um estado de atividade capaz de sofrer transições.

Por **interação** de uma máquina com seu ambiente entendemos o entrelaçamento temporal de suas ações internas e externas [34]. É pela interação que a máquina condiciona as transformações de seu programa simbólico à história das

---

da conservação material podendo ser visto como um problema de tolerância a, e recuperação de, falhas. Também não consideramos este caso na presente reflexão. Além disso, o caso estático é o limite do caso estacionário.

[23] Nota-se que a análise consiste simplesmente na consideração do espaço de estados de atividades como um sistema de transições [64] cujas configurações são estados de atividade da máquina. Isso não é mais, então, que o estudo da dinâmica do chamado meta-nível de programação da máquina [52], que é parte do seu nível simbólico, mas cuja estrutura operacional origina o nível de conhecimento (ver seção 9.3.4). A idéia de que o nível do conhecimento deve ser abordado desde um ponto de vista de sistema dinâmico foi esboçada primeiramente em [25].

reações que o ambiente fornece às *ações* que ela exerce sobre ele [24].

As transformações de um programa simbólico vão ocorrer se e quando acontecer de o programa simbólico em execução indicar ou uma transformação de si mesmo que independa das condições do ambiente, ou uma transformação de si mesmo que dependa das condições do ambiente e acontecer de ele encontrar que as condições efetivamente existentes no ambiente estão na forma requerida pela transformação.

No entanto, vale o raciocínio seguinte: se a transformação que o programa simbólico indica sobre si mesmo é independente da história da interação da máquina com o meio, então ela já estava pré-formada na estrutura operacional da máquina e, apesar de ser uma transformação na estrutura simbólica do programa, não é uma transformação no estado de atividade que ele indica. Só as transformações de programas simbólicos dependentes da história da interação da máquina com o meio é que efetivamente devem ser entendidas como transformações de estados de atividade.

Por outro lado, a respeito das ações externas da máquina sobre o ambiente, pensamos que todas devem ser vistas como tendo sempre uma produtividade interna e uma possível produtividade externa [25], e sendo usadas pela máquina para o condicionamento de novas ações internas ou externas, com o efeito de manter a máquina realizando o funcionamento correspondente ao estado de atividade em que se encontra.

Na medida em que as transformações da estrutura do programa simbólico (e das estruturas simbólicas operadas como dados por ele) só podem ser indicadas pelo próprio programa, fica clara a idéia de que uma *operação de entrada* não é simplesmente um transporte para dentro da máquina de um objeto que o ambiente lhe entrega, transporte que é neutro em relação ao objeto transportado. Ao contrário, a operação de entrada deve ser vista como uma transformação do conjunto de estruturas simbólicas operadas pelo programa, transformação que é guiada internamente por ele, e que leva em consideração as condições do ambiente, mas que não é determinada pelo ambiente: uma operação de entrada dados é na verdade uma *elaboração interna* de dados, *condicionada* pela situação do ambiente externo.

Chamamos de **ação externa incondicional** à ação externa da máquina que requer incondicionalmente a participação adequada do ambiente externo na sua realização. Chamamos de **ação externa condicional** à ação externa da

[24] Para a teoria da inteligência de máquina, seguindo o ponto insistentemente referido por Piaget, é importante ver a máquina como o sistema ativo e o ambiente como o objeto (ou, estrutura de objetos) que responde às iniciativas de ação da máquina [33]. Como salientado por Varela [127], uma vantagem do modo de falar que enfatiza a autonomia do funcionamento da máquina, por oposição ao modo de falar que considera que a máquina responde a comandos, está em que ele evita a referência ao uso pretendido, que é sempre problemática. Outra vantagem é que ele abre espaço à idéia de valores intrínsecos à máquina (ver capítulo 9). Uma terceira vantagem é que o caso em que se vê a máquina dando respostas a comandos pode ser visto como caso particular do caso em que a máquina detém a iniciativa da ação (ver seção 4.4.3).

[25] Isto é, em princípio toda ação externa é sempre uma ação de entrada e possivelmente uma ação de saída. Esta idéia já estava na base da tentativa de modelagem da percepção em [20].

máquina que é capaz de ser realizada de um modo alternativo (possivelmente não produtivo para o ambiente), se o ambiente não se encontra em condições de participar adequadamente na sua realização.

Quando uma máquina executa uma ação externa incondicional e o ambiente externo não se acha em situação de poder participar adequadamente dessa ação, essa ação é bloqueada e a parte da máquina responsável por ela se torna incapaz de executar as ações posteriores do programa que dependem da realização de tal ação.

Quando uma máquina executa uma ação externa condicional e o ambiente externo não se acha em situação de poder participar adequadamente dessa ação, essa ação se frustra na sua forma preferencial, mas a ação não é bloqueada. Ela se realiza de um modo alternativo e isso faz com que o programa possa indicar para execução as ações posteriores que dependem da realização desse modo alternativo da ação. No que segue, sempre as ações externas consideradas serão ações externas condicionais [26].

Nessas condições, o processo de transformação de um programa simbólico em função da situação do ambiente externo simplesmente condiciona as operações de transformação do programa simbólico aos resultados da ação da máquina sobre o ambiente, de modo que para cada combinação de resultados o programa simbólico seja transformado de um modo determinado.

Um programa simbólico pode sofrer então três tipos básicos de transformações em sua estrutura simbólica. Uma primeira transformação possível é o aumento da sua estrutura simbólica, pela elaboração de novas instruções a partir de dados obtidos do ambiente, e tornadas disponíveis para execução logo após sua anexação às intruções existentes anteriormente. Neste caso a transformação pode ser chamada de **expansão** do programa simbólico.

Um segundo tipo de modificação do programa é a sua **contração**, pela eliminação de instruções, conforme decisão tomada a partir de uma elaboração de dados obtidos do ambiente.

O terceiro caso possível é o da **substituição** de uma parte do programa simbólico por outra, obtida pela elaboração de dados obtidos do ambiente.

Nos três casos se vê que a função do ambiente é a de fornecer elementos para a decisão sobre a transformação a tomar, mas não o de determinar o resultado da transformação.

No uso convencional do computador de von Neumann, esse processo de transformação de programas simbólicos é usado de modo essencial, porém muito tímido, pelo sistema operacional da máquina, que tem por função colocar em execução programas simbólicos aplicativos e de sistema. O sistema operacional pode ser considerado como um programa simbólico parcial, que se expande a cada vez que carrega e põe em execução um programa determinado, e que se

[26] Cremos que a inexistência da noção de ação condicional nas teorias da concorrência está na raíz do problema da eqüidade (*fairness*) estudada por elas, mas não temos, no momento, um tratamento formal para esta idéia. Por outro lado, parece claro que o problema do raciocínio não monotônico [135] pode ser entendido como o da utilização adequada de passos de inferência que são ações condicionais, enquanto o raciocínio monotônico se vale apenas de ações incondicionais.

contrai quando termina a execução desse programa. Em particular, todo computador tem de dispor de um *carregador bootstrap*, cuja função é apenas, e exatamente, de se expandir assumindo a forma de um sistema operacional, ou pelo menos de um núcleo de sistema operacional [34].

Outro tipo de programas que são parciais por natureza são os interpretadores, que buscam do ambiente as estruturas de símbolos que vão interpretar como programas simbólicos da linguagem que realizam.

A base operacional de todo computador de von Neumann, e a razão de esses computadores poderem ser ao mesmo tempo de *uso geral* e *automáticos* no controle da execução dos programas simbólicos, está no fato de seus programas simbólicos de sistema (sistemas operacionais e interpretadores) serem programas simbólicos parciais, capazes de expansão, contração e substituição.

O efeito dessas diversas transformações de programas simbólicos no computador de von Neumann é o de deslocar a máquina no seu espaço de estados de atividade, desde o funcionamento inicial que é o do seu carregador bootstrap, até o funcionamento presente, que é o do programa que está em execução sob a supervisão do sistema operacional. No contexto histórico das sucessivas transformações do programa simbólico da máquina há, portanto, uma evolução na estrutura do programa simbólico que está continuamente indicando o funcionamento da máquina, desde a forma inicial do *carregador bootstrap*, até a forma corrente do sistema como um todo, a qual nunca é definitiva.

Em nenhum momento, porém, há transformação no próprio espaço de estados de atividade. Ele é único e fixo, porque é determinado pela estrutura material da máquina, pela sua arquitetura, que é fixa. O que varia no tempo não é a estrutura operacional do computador, mas o estado de atividade em que se encontra e, portanto, a estrutura de programa simbólico que o caracteriza.

Em consequência, pode variar o modo com que o computador atua sobre o ambiente, considerando-se o ponto de vista externo dos diversos momentos da história de sua interação com o ambiente, podendo ocorrer inclusive de ele atuar de modos diferentes sobre ambientes com características idênticas, em função de ele estar em estados de atividade diferentes em cada momento, mas não varia o conjunto dos modos possíveis de ele atuar externamente, pois esse é um conjunto fixo, pré-determinado pelo conjunto de seus possíveis estados de atividade, que é fixo e pré-determinado pela estrutura material, que não varia.

4.4.3 Regulação da atividade e desenvolvimento

Uma máquina só se desenvolve fazendo desenvolver sua estrutura material. Uma máquina com estrutura material fixa, estática ou estacionária, admite apenas transformações nos seus estados de atividade – transformações indicadas pelo programa que ela executa, e não pelo ambiente.

Para determinadas máquinas, o programa contém um único estado de atividade (um único programa simbólico, se se trata de uma máquina que alcança o nível simbólico na hierarquia de [8]) e esse programa não indica, então, transformações de estados de atividade. Dizemos nesses casos que são **estados de**

atividade com programas simbólicos fixos, mesmo quando se tratam de máquinas que não alcançam o nível simbólico [27].

Outras máquinas tem um programa que contém estados de atividade os quais determinam processos de interação com o ambiente que admitem serem vistos como processos de interação em que o meio comanda o funcionamento da máquina (entrada de dados, transformação do programa simbólico, etc.). Dizemos nesses casos que são **estados de atividade com programas simbólicos variáveis externamente**, ou **estados de atividade programáveis** [28]. A condição necessária para esse tipo de funcionamento, no entanto, é a existência na máquina de um estado de atividade referencial (indicado por um programa simbólico que chamamos de **regulador de atividade**) a partir do qual os estados de funcionamento "programados" e "ativados" pelo ambiente possam ser alcançados, e dizemos, nesse caso, que há **regulação de atividade** no funcionamento da máquina (ver capítulo 7).

Há, finalmente, máquinas cujos programas contém estados de atividade que, uma vez alcançados, determinam processos de interação com o ambiente que só podem ser vistos como processos dotados de autonomia frente ao ambiente. Não é possível, nesses casos, compreender as ações do meio sobre a máquina como "comandos", porque a máquina não "responde" a essas ações através de funcionamentos com características de uma execução submissa de uma ordem. Ao contrário, o que se observa é uma interação em que a máquina manifesta ter objetivos determinados internamente, com seu funcionamento sendo constantemente voltado para a consecução desses objetivos, e utilizando a interação para esse fim. Dizemos nesses casos que há **auto-regulação de atividade** e que os correspondentes estados de atividade são **estados de atividade auto-regulados** [29].

Há, no entanto, dois tipos de regulação de atividade, correspondendo a dois tipos de estados de atividade auto-regulados: aqueles cujas ações – independentemente de eles serem estados pertencentes a máquinas com estruturas materiais fixas ou variáveis – só afetam os funcionamentos da máquina a que pertencem causando possíveis transições de estados de atividade, e aqueles que, pertencendo a uma máquina com estrutura material variável, podem afetar não só os funcionamentos da máquina, mas também sua estrutura material e, portanto,

[27] Em [32] expressamos uma classificação das máquinas, correlativa a essa classificação de estados de atividade, em função da noção de *sistema dinâmico programado* e usando o conceito de máquinas em categorias, de M. Arbib e E. Manes [6].

[28] É interessante notar que é o estado de atividade, pelas interações que determina com o ambiente, que dá à máquina sua aparência usual de uma máquina com um funcionamento em aberto, que necessita que um programa lhe seja dado "de fora" para então ter algum funcionamento determinado.

[29] É importante notar que a auto-regulação é um processo não só no sentido operacional de se constituir numa organização temporal de ações, mas também no sentido histórico-genealógico das máquinas, por que é apenas paulatinamente que elas, com base em sua autonomia operacional, vão estendendo o campo da sua auto-regulação. Vide a história evolutiva dos sistemas operacionais [122], passando dos monitores residentes para os sitemas operacionais em disco, uniprocessados e depois multiprocessados, até sua presente tendência à distribuição, história que pode ser compreendida como um processo contínuo de ampliação do campo de regulação da máquina.

o programa da máquina, isto é, o conjunto dos estados de atividade e sua estrutura de transições. No primeiro caso, dizemos que são máquinas dotadas de **regulação operacional**. No segundo caso, dizemos que são máquinas dotadas de **regulação estrutural** [30].

[30] A noção de computação como regulação de atividade e as correspondentes possibilidades de computação como regulação operacional e regulação estrutural são examinadas no próximo capítulo.

Capítulo 5

Computação

5.1 Observações iniciais

Do ponto de vista dos sistemas físicos de símbolos, uma *computação* é um processo de transformação de estruturas de símbolos, portanto uma trajetória num espaço de estados de estruturas simbólicas.

O fato de a computação ser vista como computação *de* um programa simbólico pressupõe, como vimos, que o programa simbólico se mantenha fixo durante sua execução, e que em consequência, uma computação "aconteça" dentro de uma estrutura operatória, sem que ela modifique essa estrutura, nem muito menos que a construa. É o que identificamos como sendo a noção algorítmica de programa (ver capítulo 4) [1].

Em sintonia com o que foi discutido no capítulo 3, cremos que é justamente essa idéia – de que uma computação é uma ativação, mas não uma construção de estruturas operacionais – que origina a dificuldade de compreensão do potencial de variação contido na noção de computação.

5.2 Crítica da noção de computação

No sentido usual do termo, portanto, uma computação é uma realização temporal de ações de uma estrutura operatória, conforme um programa dado.

[1] É curioso que, apesar desse suposto não construtivismo estrutural do processo computacional, seja justamente a noção de computação, enquanto processo finitário, que faça aproximar a ciência da computação da chamada corrente intuicionista da lógica e da matemática, que se coloca em oposição às chamadas correntes platonista e formalista: para o platonismo, as estruturas lógico- matemáticas (em especial, as estruturas infinitas) tem uma natureza própria, independentemente do modo particular do pensar matemático que se refere a elas. Para o intuicionismo, porém, tais estruturas lógico-matemáticas são construções do pensamento matemático, nunca acabadas no caso das estruturas infinitas, mas elaboradas a partir de certas intuições operacionais fundamentais referentes a essas construções mentais e de modo tal que, em certo sentido, as operações sobre elas podem ser vistas como computacionais [65], [126].

No capítulo 3, estabelecemos uma diferença entre *computação algorítmica* e *computação não algorítmica*: uma computação algorítmica é um modo particular de computação que ocorre em uma estrutura operatória fixa, conforme um programa fixo e com um processo de interação elementar da forma *entrada inicial → cálculo → saída final*.

Já uma computação não algorítmica é uma computação que consiste em realizar no tempo ações pertencentes a uma estrutura operatória que pode ela própria variar temporalmente, e segundo um programa variável, com um processo de interação em que ações internas e externas se sucedem temporalmente de modo essencialmente entrelaçado.

Vemos, porém, que essa dupla classificação das computações não é suficiente para esgotar o conjunto dos possíveis tipos de classificação: há ainda uma possibilidade intermediária de uma computação que ocorre em uma estrutura operatória fixa, mas segundo um programa que pode variar no tempo. Já que nas computações não algorítmicas o programa deve variar necessariamente, posto que varia a estrutura operatória cujas ações ele organiza, classificamos esse tipo intermediário de computação como computação não algorítmica e subdividimos as computações não algorítmicas em dois tipos: computações não algorítmicas a estruturas operatórias fixas e programas variáveis (ou, *computações não algorítmicas a estruturas fixas*, simplesmente) e *computações não algorítmicas a estruturas variáveis* (que são necessariamente a programas variáveis).

Temos então, como **resultado final** da análise da noção de computação, o seguinte conjunto de tipos possíveis de processos computacionais:

computação algorítmica
com estrutura opertatória fixa e programa fixo;

computação não algorítmica a estrutura fixa
e programa variável;

computação não algorítmica a estrutura variável
e programa variável.

Podemos agora definir com maior clareza a natureza dos processos computacionais.

5.3 Definição de computação

Partimos de uma interpretação das idéias gerais expressas por D. Scott em [110], e procuramos explicitar com base nelas um sentido para a noção de *construção* que entendemos estar implícita na noção de computação.

Um **domínio** de objetos computacionais é essencialmente um conjunto parcialmente ordenado de objetos parciais, fechado para supremo de cadeias ou conjuntos dirigidos, com um menor elemento, que é então completamente parcial [2]. As operações computacionais sobre esses objetos computacionais são

[2] Ver [57], [111], [102], p.ex., para apresentações variadas da teoria dos domínios.

funções parciais sobre esse domínio [3].

Pensamos que, para alcançar a noção de construção embutida na noção de computação, a *ordem de aproximação* entre objetos parciais deve ser vista como uma **ordem de construção**: $x \sqsubseteq y$ significa que x é objeto constituinte de y, $x \sqcup y$ é o menor objeto que tem x e y como objetos constituintes, e $x \sqcap y$ é o maior objeto comum à constituição dos objetos x e y.

Um **passo de construção** de um objeto qualquer x é qualquer aplicação de transformação f de objetos tal que para algum $y \sqsubseteq x$ vale $y \sqsubseteq f(y) \sqsubseteq x$. Uma **operação de construção** é qualquer transformação f de objetos tal que para todo objeto x vale $x \sqsubseteq f(x)$, isto é, uma operação de construção é uma transformação *ampliadora* de objetos.

Uma **construção** de um objeto de um domínio computacional é uma cadeia (ou um conjunto dirigido) de objetos do domínio tal que o supremo da cadeia é o objeto considerado.

Nessas condições, definimos:

> Uma **computação** é uma realização temporal de uma construção em um domínio de objetos computacionais.

Essa definição apresenta as seguintes características que serão comentadas na próxima seção:

1. por ser um conjunto organizado de operações, toda construção (logo, toda computação) se dá conforme um programa (fixo ou variável);

2. nem todo passo de construção necessita ser realizado por uma operação de construção;

3. toda construção (logo, toda computação) é uma trajetória no espaço de objetos parciais em que ela acontece;

4. as computações apresentam duas dimensões, que chamamos de *dimensão diacrônica* e *dimensão sincrônica*, ambas constituindo-se em construções em domínios computacionais (diferentes, porém relacionados), e isso permite estabelecer a noção de computação como regulação.

5.4 Comentários à definição de computação

Nesta seção apresentamos esboços do tratamento formal da noção de computação como realização temporal de uma construção, das noções de estrutura

[3] Note-se que, ao contrário de [110], aqui não se exige a monotonicidade das operações sobre objetos parciais. Em compensação, como se verá na seção 5.4.4, o uso das operações não monotônicas deve se dar conforme uma determinada disciplina – portanto conforme determinadas restrições de programação – que garantam a efetividade de sua utilização. Vê-se que, assim, se re-introduz na caracterização semântica dos processos computacionais um aspecto operacional. Cremos que isso apenas reflete o fato, já indicado anteriormente no capítulo 4, da impossibilidade de uma semântica puramente declarativa para os processos computacionais, toda semântica devendo conter, implicita ou explicitamente, um componente operacional.

operatória e estrutura operacional de uma máquina, a idéia da computação como trajetória em um espaço de objetos computacionais, e seus aspectos sincrônico e diacrônico.

Em todos os casos, sempre associamos uma máquina M com um domínio D de objetos parciais, que é o domínio que indica a natureza dos objetos que M potencialmente é capaz de conter e manipular. Escrevemos M_D para indicar isso, e dizemos que M_D *funciona em* D.

5.4.1 Computação como realização temporal de uma construção

Seja D um domínio de objetos computacionais. Representamos um *passo* na construção de $x \in D$, realizada através de uma transformação $p : D \to D$, pelo intervalo em D dado por $p_y = (y, p(y))$, para algum $y \sqsubseteq x$. Se p_y é um passo de construção de $x \in D$ então escrevemos $p_y \in P_D(x)$. O conjunto dos passos de construção dos objetos de D é $P_D(D)$. Se $p_y, q_z \in P_D(D)$ e $p_y = (y, z)$, então escrevemos $p_y \mathrel{—\!\!<} q_z$ [4].

Se $p_y = (y, z)$ é um passo de construção em $P_D(D)$ então $ob(p_y) = z$ é o objeto construído por p a partir de y. Se $P \subseteq P_D(D)$ então $ob(P)$ é o conjunto dos objetos construídos por todos os $p_y \in P$, com $y \in D$.

Seja P uma cadeia de passos de construção em $P_D(D)$ que constrói $x = \bigsqcup(ob(P))$. Seja $(T, \leq)$ o conjunto ordenado dos instantes de tempo. Uma **computação da construção** P é uma atribuição de passos da construção aos instantes de tempo $C_P : P \to T$ de modo que valham as seguintes condições: $p_y \mathrel{—\!\!<} q_z \Rightarrow C_P(p_y) \leq C_P(q_z)$ e $C_P^{-1}(t)$ é finito para todo $t \in T$ [5]. Com isso, escrevemos também: $x = ob(C_P)$.

Seja π um programa. Se os passos da construção P estão de acordo com a organização determinada por π então P é uma construção de acordo com π. Nesse caso, toda computação de P é uma computação de π. Assim, uma **computação de um programa** é uma computação de uma construção que está de acordo com esse programa [6].

Vemos então que as características atemporais de uma computação, especialmente a fixidez/variabilidade da estrutura operatória e do programa, são dadas pela construção que ela realiza no tempo. Assim se tem, de fato, que as construções, e não as computações, é que são realmente ou *algorítmicas*, ou *não algorítmicas a estrutura fixa*, ou *não algorítmicas a estrutura variável*, e que as computações são de um desses tipos apenas conforme a construção que realizam.

[4] A idéia aqui é construir $P_D(D)$ como uma estrutura de eventos para a relação $—\!\!<$ (ver [89]).

[5] Note-se que, assim, conjuntos finitos de passos podem ser realizados simultaneamente.

[6] Desse modo fica diferenciada a noção de *construção*, que requer apenas o ordenamento das operações estipuladas por um programa, e a noção de *computação*, que estipula a distribuição das operações da construção no tempo. Dito de outro modo, uma construção é uma entidade conceitual atemporal enquanto que uma computação é uma entidade real temporal.

Estrutura operatória de uma máquina

Seja D um domínio de objetos computacionais em consideração. Seja M_D uma máquina funcionando em D. Uma **estrutura operatória** para M_D é uma atribuição de sub-conjuntos $o \subseteq D \to D$ a M_D conforme os instantes do tempo $O_{M_D} : T \to 2^{D \to D}$. Se $O_{M_D}(t) = o$ então o é a estrutura operatória de M_D no instante t. Se M_D tem estrutura operatória o no instante t então escrevemos também $M_D^{o_t}$. Com isso se tem: $O_{M_D} = \{M_D^{o_t}\}_{t \in T}$.

Programa global e estrutura operacional de uma máquina

A **estrutura operacional global**, ou **programa global**, de M_D é uma atribuição de **estruturas operacionais** (estruturas de eventos) $\pi \subseteq P_D(D)$ conforme os instantes do tempo $\Pi_{M_D} : T \to 2^{P_D(D)}$ de modo que se O_{M_D} é a estrutura operatória de M_D então $\Pi_{M_D}(t) \subseteq O_{M_D}(t)$ para todo $t \in T$. Se $\Pi_{M_D}(t) = \pi$ então π é o programa de M_D no instante t e escrevemos também $M_D^{\pi_t}$. Então, $\Pi_{M_D} = \{M_D^{\pi_t}\}_{t \in T}$.

Vê-se que a idéia é a de que o programa global da máquina determina a estrutura operacional que a máquina tem em cada instante, isto é, a parte da estrutura operatória que a máquina pode por em execução naquele instante.

Vê-se assim que em cada instante uma máquina só pode mobilizar, isto é, realizar temporalmente, a parte de sua estrutura operatória que é realizada pela estrutura operacional que ela tem naquele instante, e que em cada instante sua estrutura operacional não pode estender-se para além de sua estrutura operatória.

Tipos de construção

Dado que é a natureza das construções que determina a natureza das computações, podemos esboçar em relação às construções a formalização das noções de algoritmicidade e não algoritmicidade. Seja D o domínio em que funciona a máquina M_D. Seja $x \in D$.

- **Construção algorítmica**

  Uma construção P de x pela máquina M_D é uma *construção algorítmica* (isto é, a estrutura e programa fixos) se e somente se:

  i) a estrutura operatória de M_D é fixa, isto é, $O_{M_D}(t) = O_{M_D}(t')$ para todo $t, t' \in T$;

  ii) o programa de M_D é fixo, isto é, $\Pi_{M_D}(t) = \Pi_{M_D}(t')$ para todo $t, t' \in T$;

  iii) a construção P de x é compatível com o programa fixo de M_D.

- **Construção não algorítmica a estrutura fixa**

Uma construção não algorítmica P de x pela máquina M_D é uma *construção a estrutura fixa e programa variável* se e somente se:

i) a estrutura operatória de M_D é fixa, isto é, $O_{M_D}(t) = O_{M_D}(t')$ para todo $t, t' \in T$;

ii) o programa de M_D é variável, isto é, existem $t, t' \in T$ tais que $\Pi_{M_D}(t) \neq \Pi_{M_D}(t')$;

iii) existe uma sequência de transformações de M_D que origina a sequência de programas $\pi_1, \pi_2, \ldots, \pi_n, \ldots$ dentro da estrutura operatória fixa de M_D, e existe uma sequência $P_0, P_1, \ldots, P_n, \ldots$ de etapas em que P se decompõe, de modo tal que P_n é compatível com π_n para cada $n = 0, 1, ...$

- **Construção não algorítmica a estrutura variável**

Uma construção não algorítmica P de x pela máquina M_D é uma *construção a estrutura variável* (e, portanto, a programa variável) se e somente se:

– existe uma sequência de transformações de M_D, determinando uma sequência de estruturas operatórias $o^0, \ldots, o^k, o^{k+1}, \ldots$, uma sequência de programas $\pi_0^0, \pi_1^0, \ldots, \pi_{n_0}^0, \ldots, \pi_0^k, \ldots, \pi_{n_k}^k, \pi_0^{k+1}, \ldots$ e uma sequência $P_0^0, P_1^0, \ldots, P_{n_0}^0, \ldots, P_0^k, \ldots, P_{n_k}^k, P_0^{k+1}, \ldots$ de etapas de P de modo tal que P_n^k é compatível com π_n^k, para cada $k = 0, 1, ...$ e $n = 0, 1, \ldots, n_k$.

Tipos de programas globais de uma máquina

Em relação aos programas globais conforme os quais as computações ocorrem, temos a seguinte classificação, correlativa à classificação das computações que foi dada na seção 5.2:

programa global fixo (estrutura operacional global fixa)
com conjunto de operações fixo e organização fixa;

programa global variável (estrutura operacional global variável)
com conjunto de operações fixo e organização variável;

programa global variável (estrutura operacional global variável)
com conjunto de operações e organização variáveis.

Assim, vemos como as três classes de computações se relacionam, de modo respectivo e desde o ponto de vista da construtividade, com as três classes de programas globais conforme os quais as computações ocorrem.

5.4.2 Computação como trajetória

Se D é um domínio de objetos computacionais, com $x, y \in D$ e $x \sqsubseteq y$, então podemos dizer que há uma trajetória em D indo de x até y: se P é uma construção de y, isto é, $y = \bigsqcup(ob(P))$, então $P\uparrow_x = \{p_{x'} \in P \mid p_{x'} = (x', x'') \wedge x \sqsubseteq x'\}$ é uma trajetória em D no sentido de que, começando em x, cada $p_{x'} \in P\uparrow_x$ faz a construção de y avançar em D deslocando-se de x' para x'', quando $p_{x'} = (x', x'')$. Então podemos dizer que D constitui um *espaço de fase* para a máquina que realiza P, e que P é um conjunto de trajetórias nesse espaço, que as máquinas podem percorrer quando realizam computações de P.

É interessante notar, por outro lado, que além dos paralelos entre máquina e sistema dinâmico, e construção e trajetória em espaço de fases, também é possível estabelecer uma correlação direta entre programa e sistema de equações: por exemplo, os sistemas de equações de diferenças (ver [72], p.ex.) são exatamente os sistemas de equações recursivas (ver [107], p.ex.) em que o caso básico foi removido do sistema e passou a ser tratado como condição de contorno [7].

Quer dizer, a abordagem de sistemas dinâmicos, usual no tratamento de processos físicos, não se retringe a ser aplicada à computação com redes neuronais, como já se tornou tradicional [3], mas também se estende à análise dos sistemas de computação simbólicos, tal como indicado já em [32] (ver também [79]).

5.4.3 Ação computacional e independência de uso

Nessas condições, podemos dizer que a **ação computacional** executada por uma máquina, num ambiente determinado, é exatamente a construção que ela realiza enquanto está funcionando nesse ambiente, isto é, a ação computacional da máquina é o conjunto dos efeitos produzidos, nela e no ambiente, pelo conjunto de operações que constituem tal construção.

Então, se vê que se uma máquina M_D funcionando em D realiza P, tanto seu funcionamento global como seu comportamento – expresso pelas operações de interação que ela executa – são independentes de qualquer uso que se possa dar a eles, e na verdade podem ser tomados como pontos de partida para a determinação dos usos possíveis que se pode dar à máquina.

5.4.4 Computação como regulação

Definimos uma computação como uma realização temporal de uma construção. Assumimos que o conjunto dos tempos tem uma estrutura linear. Então, independentemente de as construções serem ordenações totais ou parciais de operações, toda computação é uma organização linear das operações que constituem a construção que ela realiza, mas no sentido de uma pré- ordem, já que várias operações diferentes podem ser atribuídas a um mesmo instante de tempo [8].

[7] Também, é possível interpretar as operações de integração e diferenciação em domínios computacionais, quando as variáveis de tempo discreto são entendidas como listas [47].

[8] Alternativamente, uma computação pode ser pensada como uma atribuição de conjuntos de operações aos instantes de tempo.

Então, como conjunto de sequências de (conjuntos de) operações, o conjunto das computações pode ser pensado como um domínio de sequências e todo passo de construção pode ser pensado como uma operação que estende a sequência da computação em que se realiza.

Por outro lado, dada uma computação podemos considerar a *variação* que sofre o elemento que ela está construindo quando a computação passa de um instante para outro. Tradicionalmente, desde a análise pioneira de Scott [110], essa variação é considerada como um crescimento: se C_P é uma computação de P então $ob(C_P(t)) \sqsubseteq ob(C_P(t'))$ para todo $t, t' \in T$ com $t \leq t'$.

A razão para essa exigência resulta da análise conceitual da noção de interação entre uma máquina e seu ambiente, feita em [110] (ver também [121]): a produção de informações por uma máquina para seu ambiente é um processo monotônico no sentido do crescimento da informação produzida, pois uma vez a informação tendo sido produzida para o receptor, o emissor não tem mais como cancelar essa produção.

Pensamos, contudo, que tal análise faz uma amarração desnecessária entre dois aspectos distintos dos processos de interação. Nesses processos podemos distinguir, por um lado, o *aspecto factual* dos acontecimentos físicos que constituem a interação, e por outro lado, o *aspecto semântico-informacional* da interação, que se constrói sobre a base daqueles acontecimentos físicos.

Assim, é certo que os acontecimentos físicos, ocorrendo dentro da estrutura irreversível de sequenciamento do tempo físico, realmente não podem ser revertidos uma vez realizados, posto que os instantes de tempo em que ocorreram não podem mais ser materialmente alcançados, após terem ocorrido. Desde o ponto de vista físico, portanto, o processo de interação é necessariamente monotonicamente crescente, no que diz respeito ao comprimento das sequências que o compõem.

Por outro lado, os objetos físicos são despidos, em si mesmos, de toda significação – e, de fato, essa é a idéia que está na base da supressão dos compromissos teleológicos dos processos físicos, admitida pela ciência desde Galileu. Os significados de um objeto surgem, então, somente pela sua inserção em um sistema de operações realizadas real ou potencialmente por um agente, e só existem para esse agente e são relativos à medida e à maneira como ele os insere nesse seu sistema de operações [9].

Assim, não há razão para atrelar à irreversibilidade do crescimento físico do processo que constitui a interação de uma máquina com seu ambiente, a natureza crescente ou decrescente do aspecto informacional desse processo de interação: não há por que recusar a idéia de um processo de interação que resulte na transmissão de uma informação negativa, a qual reduza a informação anteriormente existente no receptor da transmissão.

Então, se desde o ponto de vista do senso-comum é difícil aceitar a idéia de uma quantidade negativa de informação armazenada em algum lugar – posto

[9] Em outros termos, significados são atribuídos aos objetos físicos, e não extraídos deles. Se essa significação é puramente sensorio-motora ou se é representativa, se é não intencional ou se é intencional, isso depende da estrutura do agente e não especialmente do objeto físico (sobre essa idéia, ver [94]).

que, por exemplo, a própria expressão *carência de informação* significa apenas inexistência de informação, isto é, informação nula – não é difícil aceitar, desse mesmo ponto de vista, a idéia de uma *transmissão* positiva ou negativa de informação, isto é, uma transmissão que aumente ou reduza a quantidade de informação existente no lugar que recebe a transmissão [10].

Portanto, admitimos a noção de **informação relativa**, constituída em analogia à noção de *número relativo*, e falamos de *informação positiva*, de *informação nula* e de *informação negativa*. Em outros termos, falamos das operações de **adicionar** e **subtrair** informações, e da operação **identidade** de informações [11].

Dada a noção de informação relativa, a noção de *computação como regulação* se estabelece imediatamente: um processo computacional é **regulativo** se e somente se o efeito das interações que o constituem é o de manter a informação existente no objeto que ele constrói o mais próximo possível de um marco referencial, estabelecido pela máquina ou pelo ambiente.

Vê-se então que a noção convencional de computação, dirigida a alcançar o limite da construção que ela realiza, é apenas um caso particular de computação regulativa: o caso em que o marco referencial indica que a computação deve realizar o máximo de informação positiva, compatível com as condições estabelecidas pelo programa que regula aquela construção.

Assim, vê-se também em que um programa é um elemento regulador de uma computação, num sentido mais amplo que o de simplesmente ordenar operações: *o programa de uma computação é a estrutura que regula a informação contida no objeto construído por aquela computação.*

5.4.5 Diacronismo e sincronismo

Podemos, agora, distinguir os dois aspectos, diacrônico e sincrônico, de uma computação: o **aspecto diacrônico** diz respeito às variações de estrutura operatória e estrutura operacional da computação, enquanto que o **aspecto sincrônico** diz respeito às variações de estado de atividade.

Vê-se, então, que as computações algorítmicas só comportam aspectos sincrônicos, e sempre relativas ao crescimento da informação. São as computações não algorítmicas que apresentam aspectos diacrônicos e que, portanto, admitem uma noção interna de tempo – uma *história* – e que, por isso, estão abertas à noção de desenvolvimento.

Do ponto de vista estrutural, então, diacronismo e sincronismo se apresentam como relativos a domínios computacionais diferenciados: o diacronismo diz respeito a sequências de estruturas operatórias e operacionais, no domínio de tais estruturas, enquanto que o sincronismo diz respeito a sequências de estados

[10] Essa é, inclusive, a idéia que dá origem aos raciocínios e às lógicas não monotônicas [135], onde o positivo e o negativo são interpretados como afirmação e negação.

[11] Contudo, não pretendemos sair dos marcos da noção *qualitativa* de informação estipulada por Scott (ver [112], p.ex.), embora sejamos obrigados a reconhecer que talvez os domínios computacionais existentes dentro dos marcos da topologia T_0 não sejam suficientes para comportar essa noção de relatividade de informação.

de atividade daquelas estruturas, no domínio correspondente. Vê-se, no entanto, que esses domínios diferenciados são relacionados um ao outro.

5.4.6 Universalidade, estabilidade e competência computacional

Em [84], Newell estabelece a *universalidade* como característica geral definidora dos sistemas físicos de símbolos: a universalidade de uma máquina é a capacidade da máquina de realizar todos os comportamentos fisicamente possíveis, compatíveis com a natureza do seu funcionamento. Os sistemas físicos de símbolos são então os sistemas universais para os comportamentos simbólicos, e a máquina de Turing é o modelo teórico desse tipo de sistema.

Podemos dizer, portanto, que quanto mais uma máquina se aproxima da máquina universal, maior é sua **competência computacional**.

Em função das análises conceituais realizadas até agora – assim como das que se seguirão nos próximos capítulos – pensamos ser adequado reformular essa noção de competência computacional, substituindo a noção de universalidade pela noção de *estabilidade teleonômica*, definida na seção 3.4.4.

A noção de universalidade supõe que o comportamento da máquina seja pensado em termos de computação de uma função entre conjuntos de dados de entrada e de saída. Quer dizer, ela é relativa à noção de *máquina calculadora*, e não à noção de máquina computadora: todos os aspectos de interação e de regulação operacional existentes nas máquinas computadoras (bem como os aspectos de regulação funcional e de desenvolvimento que serão examinados nos caítulos que seguem) não são considerados por aquela noção, devido ao seu caráter estritamente comportamentalista.

A noção de estabilidade teleonômica é, no entanto, uma medida da competência computacional da máquina mais vantajosa que a noção de universalidade. Por um lado, ela serve os mesmos *propósitos externos* que Newell destinou à universalidade: avaliar o quanto a máquina é capaz de atender às solicitações de diversidade de comportamento que o meio lhe coloca.

Por outro lado, a noção de estabilidade teleonômica serve *propósitos internos* que escapam à universalidade: a estabilidade teleonômica é uma medida da necessidade de transformação interna – portanto, da necessidade de desenvolvimento – da máquina, posto que indica também as dificuldades que suas estruturas operatória e operacional apresentam para assimilar as ações e reações do ambiente com que a máquina interage. A estabilidade teleonômica serve também, portanto, como índice referencial para a regulação do desenvolvimento.

Capítulo 6

Organização de máquina

6.1 Observações iniciais

A noção usual de **sistema** é a de um *conjunto organizado* de objetos que, quando funcionantes, interagem produzindo um funcionamento global. Frequentemente se usa o termo *comportamento do sistema* para se referir, além do seu comportamento observável externamente, também ao seu funcionamento interno. Assim, é comum formular a idéia de sistema como um conjunto organizado de objetos que apresenta um comportamento global.

A **organização** de um sistema é, numa primeira aproximação, a totalidade das suas estruturas material, operacional e funcional. Na prática dos sistemas complexos, a organização de um sistema costuma ser dividida em *níveis hierárquicos* [133],[116].

Um **sistema dinâmico** é um sistema para o qual é possível identificar um espaço de estados e uma função de transformação de estados [79]. Uma máquina computadora é um sistema dinâmico, em diversos sentidos [32]. A *hierarquia dos níveis de organização de sistemas computadores* estabelecida em [8] é um modo conveniente de ordenar esses diversos sentidos [1]. Ela permitiu a A. Newell, por exemplo, distinguir nas máquinas computadoras o *nível simbólico* e *o nível do conhecimento* [84],[85], níveis cuja compreensão é essencial para a fundamentação da IA.

Contudo, essa noção de hierarquia de níveis de organização, amplamente difundida na ciência da computação, que fez sua entrada oficial na IA com aqueles artigos de Newell, e que parece fecunda para uma abordagem construtivista da inteligência de máquina, apresenta uma lacuna essencial: falta-lhe exatamente a explicitação de seu processo de construção, e com isso a prova de sua natureza concreta e construtivista.

Em consequência, o uso da noção de nível de organização se faz sempre sob

[1] Originalmente, e mesmo usualmente, a denominação utilizada para os níveis é a de *nível de sistema*. Aqui usamos o termo *nível de organização* para estabelecer um vínculo mais estreito desses níveis com as estruturas material, operacional e funcional dos sistemas.

o dilema clássico entre dar aos níveis da hierarquia, e portanto à própria organização, o estatuto de entidades reais, ou dar a eles o estatuto de entidades apenas conceituais e descritivas [2]. Assim, na prática das máquinas computadoras, os níveis da hierarquia são sempre pensados como construções reais, mas curiosamente costumam ser chamados de "níveis de abstração", e mesmo de "níveis de descrição".

O quadro geral da utilização da noção de "nível" na ciência da computação é, então, o da situação paradoxal de uma ciência de objetos artificiais, portanto construídos, cuja prática procede com base no realismo da hieraquia dos níveis de organização das máquinas, mas que não dá uma explicação da construção dos níveis dessa hierarquia, e que chama esses níveis de "níveis de abstração" [3].

Frente a essa situação, não deve ser surpresa que Newell se sinta obrigado a reiterar o realismo da hieraquia cuja existência ele oficializou [85]:

- *Computer systems levels are not simply levels of abstraction.* (p.97)
- *To sum up, computer system levels are a reflection of the nature of the physical world. They are not just a point of view that exists solely in the eye of the beholder.* (p.98)
- *To repeat the final remark of the prior section: Computer system levels really exist, as much as anything exists. They are not just a point of view.* (p.99)

Neste capítulo, visamos expor o modo com que se dá concretamente a construção dos níveis de organização das máquinas computadoras, compatibilizando o caráter de "abstração" desses níveis com seu caráter de realidade.

Para enfatizar o caráter real dos níveis de organização, evitamos chamá-los de níveis de descrição e abstração.

6.2 Crítica da noção de organização de máquina

6.2.1 A noção de hierarquia de níveis de organização

O termo *hierarquia de níveis de organização* não tem um uso claramente consolidado e podemos distinguir nele dois sentidos principais, um contendo uma referência interna à máquina computadora, outro contendo uma referência externa, sendo que o sentido interno pode ser apreendido de três modos diferentes: estrutural, operacional e funcional. Precisamos portanto elucidar esses diferentes sentidos.

[2] Não se pretende aqui abordar o problema geral das estruturas hierárquicas [133], mas apenas mostrar como os níveis de organização de máquinas computadoras, para além de seu aspecto auxiliar descritivo, constituem de direito realidades operacionais e funcionais nas máquinas computadoras, como ensina de fato a prática da computação.

[3] Newell, contudo, usa sempre o termo nível de sistema e mesmo o contrapõe ao termo nível de abstração. Por exemplo, em: *In fact, it* [the configuration level] *is more nearly a pure level of abstraction than a true system level* [85, p.97].

Para tanto, convém introduzir uma noção geral que serve para determinar os contextos das referências externas: a noção de **família de sistemas**, que formulamos frouxamente apenas como sendo *um conjunto de sistemas que tem natureza assemelhada*.

Por um lado, então, um nível de organização pode ser pensado como constituindo uma parte essencial da organização do sistema a que diz respeito. Nesse sentido, ele é uma realidade *no* sistema, e como nível ele é um nível *do* sistema.

Diversos níveis diferentes podem ser identificados em um mesmo sistema e ordenados segundo algum "grau de organização", constituindo o que chamamos de a **hierarquia interna dos níveis de organização** daquele sistema.

Em consequência, dada uma família de sistemas, o conjunto de todos os níveis de organização que se podem encontrar nos sistemas dessa família constitui também uma hierarquia. Cada sistema da família realiza, na sua organização particular, uma parte dessa hierarquia geral da família.

Desse modo, então, um nível de organização pode ser pensado como tendo uma referência externa aos sistemas, pois também é uma *classe* (ou, um *tipo*) que compara cada sistema, quanto à sua estrutura e funcionamento, com outros dentro da mesma família. É assim que um sistema pode ser colocado na classe dos circuitos eletrônicos, ou na classe dos circuitos digitais, ou na classe dos computadores de von Neumann, etc., conforme ele tenha a organização de um circuito eletrônico, de um circuito digital ou de um computador de von Neumann, etc., respectivamente.

Nesse sentido, o sistema enquanto membro da família é um sistema *do* nível de organização.

Esse tipo de classificação constitui o que chamamamos de a **hierarquia externa dos níveis de organização** da família considerada. Cada sistema da família se enquadra em um nível determinado dessa hierarquia externa, que se constitui em *uma classificação* dos sistemas da família, determinada em função de seus níveis de organização [4].

No entanto, o que ocorre com as duas hierarquias de níveis de sistema, interna e externa, é que elas se correspondem termo a termo, e cada sistema da família considerada é classificado na hierarquia externa exatamente no nível que corresponde ao nível mais alto que pode ser encontrado na sua hierarquia interna.

Quer dizer, a hierarquia externa de uma família de sistemas se constitui com base na hierarquia interna. Por isso, consideramos aqui que o sentido principal do termo hierarquia de níveis de organização é o de hierarquia interna.

Por outro lado, o próprio termo **hierarquia** necessita clarificação, pois se é certo que falar em hierarquia é falar em estrutura ordenada de níveis, não se diz – só com o termo *hierarquia* – qual é exatamente o tipo lógico da relação que ordena os níveis e que consitui assim tal hierarquia.

[4] Em outras palavras, o termo *nível de organização* é usado tanto de modo intensional, referindo a hierarquia interna, quanto de modo extensional, referindo a hierarquia externa. Pensamos que é nesse segundo sentido que deve ser entendida a frase: *Each computer system level is a* specialization *of the class of systems capable of being described at the next lower level* [85, p.97].

Do ponto de vista lógico [96], é possível pensar em dois grandes tipos de relações entre os níveis de uma hierarquia, as **relações de encaixamento** e as **relações de seriação** [5].

No primeiro caso, trata-se da relação entre níveis que se encaixam uns dentro de outros, como no caso bem conhecido das estruturas de módulos e submódulos. No segundo caso, trata-se da relação entre níveis que se ordenam segundo o grau ou intensidade de uma certa propriedade, como no caso também conhecido das hierarquias de postos de mando em organizações, que se ordenam pela sua força institucional [6].

Cabe portanto a pergunta sobre se a hierarquia interna dos níveis de organização das máquinas computadoras é uma hierarquia de encaixamentos ou uma hierarquia de seriações, e em qualquer caso é preciso dizer quais são exatamente os elementos que constituem os níveis dessa hierarquia e qual o critério de ordenação.

6.2.2 A natureza da estrutura material das máquinas computadoras

Como a noção de organização de máquina computadora refere as noções de estrutura material, operacional e funcional da máquina, temos então o problema de caracterizar em cada máquina o que é um *componente*, o que é uma *ação* e o que é uma *função* realizada por um componente, elementos que determinam, respectivamente, as estruturas material, operacional e funcional dela (ver capítulo 3).

A prática da computação tem insistido sobre a independência da natureza dos processos computacionais em relação à natureza da estrutura material (hidráulica, eletrônica, etc.) que os realiza. É claro que o princípio dessa insistência é o ponto de vista artificialista, interpretativo, que analisamos no capítulo 3. Porém, algo dessa noção podemos reter, que é exatamente o fato de esses *processos* serem *computacionais*, e não hidráulicos, eletrônicos, etc. Em consequência, vemos que a natureza da estrutura material de uma máquina computadora – e com isso a natureza dos seus componentes – não pode ser alcançada *per se*, mas somente a partir das estruturas operacional e funcional dessa máquina.

Ora, no capítulo 3 verificamos que tanto a estrutura operacional quanto a estrutura funcional de uma máquina computadora se determinam justamente a partir da estrutura material da máquina, porque toda ação executada e toda função desempenhada na máquina são sempre uma ação e uma função *de* um ou mais componentes da máquina. Donde se conclui que estamos diante de uma circularidade conceitual: o aspecto material é determinado pelos aspectos

[5] As primeiras tem a forma de uma estrutura de ordem parcial, as segundas de uma estrutura de ordem total.

[6] Vê-se que, nesse caso, a mera indicação da forma parcial ou total da relação de ordem não é suficiente para caracterizar o uso pretendido do termo hierarquia de níveis, e que os termos *encaixamento* e *seriação* contém mais do que a indicação dos detalhes da estrutura de ordem, pois indicam também algo sobre a natureza dos objetos ordenados e sobre o critério de ordenação.

operacional e funcional, e vice-versa

Resolvemos essa circularidade pela introdução de uma noção que desempenha um papel essencial na reflexão que estamos conduzindo aqui. É a noção abstrata de *componente computacional* [7].

A noção de componente computacional é introduzida aqui ao estilo das diversas noções de "ponto abstrato dotado de propriedades físicas" que se utilizam, desde Galileu, para estudar os vários aspectos físicos da realidade: *ponto material* no estudo da mecânica, *carga puntual* em eletricidade ou magnetismo, etc. Portanto, a noção de componente computacional é introduzida com o intuito de embasar a abordagem racional ao estudo da dinâmica das máquinas computadoras, em que as máquinas são vistas como *sistemas de componentes computacionais*, estudo necessário à elaboração da noção de *desenvolvimento de máquina*, que introduzimos no capítulo 9.

Assim definimos um **componente computacional** como um elemento abstrato capaz de *realizar* ou *sofrer* ações computacionais (para a noção de *ação computacional*, ver sec. 5.4.3) [8].

A circularidade conceitual encontrada antes se dissolve então porque a existência de componentes computacionais é postulada em relação a qualquer máquina computadora: por princípio, toda máquina computadora é um sistema de componentes computacionais. Assim, toda máquina computadora tem uma **estrutura material** num sentido abstrato bem preciso, o de *estrutura de componentes computacionais*, que não tem uma natureza física definida a priori, mas que pode ser utilizada para determinar a estrutura operacional (e consequentemente a estrutura funcional) da máquina dada, com base na idéia de que toda ação computacional ocorrida na máquina é produzida por um componente computacional ou conjunto de componentes computacionais.

Dessa forma, o termo *máquina computadora* passa a referir uma noção que tem como extensão os artefatos concretos tal como eles foram caracterizados no capítulo 3, e que tem como intensão um construto teórico ao estilo das noções abstratas usuais de *autômato* e de *máquina sequencial*. Só que, ao contrário dessas noções, que visam captar um particular modo de funcionamento e de uso das máquinas reais (tal como indicado na seção 3.2.2), a noção de máquina computadora, pensada como um sistema de componentes computacionais, visa captar a variedade total de organizações e funcionamentos possíveis das máquinas computadoras reais.

Com isso, se torna possível pensar a questão da natureza da organização das máquinas computadoras, e da hierarquia dos seus possíveis níveis de organização, a partir das estruturas operacionais e funcionais, pois é certo por princípio que para cada máquina será encontrada uma estrutura de compo-

[7] Ao contrário da noção de *autômato*, que refere uma *totalidade* autônoma, a noção de componente computacional adotada aqui pretende referir *partes* constituintes de uma totalidade autônoma, que podem ou não ser autônomas elas mesmas.

[8] Originalmente pensamos no termo *mecanema*, do grego *μηχανή* (*máquina*) e *νέμω* (*partir, distribuir*), para designar o conceito. Cremos que esse termo, abstraído das significações negativas que lhe foram historicamente superpostas (ver Introdução a [28]) é bastante conveniente para referir os componentes de máquinas computadoras. Terminamos optando, porém, pelo termo mais coloquial de *componente computacional*.

nentes computacionais para ela: a estrutura de componentes computacionais que *suporta adequadamente* (no sentido definido na seção 3.4.4) suas estruturas operacional e funcional.

Em consequência, a *estrutura operacional* de uma máquina não pode mais ser entendida como um dado relativo e casual, apenas descritivo da sua constituição, resultante de uma decomposição arbitrária do seu funcionamento (p.ex., obtida em função de algum objetivo de uso que se tenha ao fazer a análise dela). Ao contrário, se vê nessas circunstâncias, que a estrutura operacional de uma máquina tem um caráter real e necessário nessa máquina, pois é efetivamente intrínseca ao conjunto de seus componentes computacionais.

Do mesmo modo, toda *construção operacional* (no sentido a ser dado na seção 6.4.3) que puder ser determinada a partir da estrutura operacional de uma máquina também tem, devido ao modo de proceder desse tipo de construção, caráter real e necessário nessa máquina.

Por outra parte, é claro, a *estrutura material* da máquina, quando ela se incarna em um artefato real particular, fica determinada não diretamente por sua estrutura operacional ou funcional, mas pela sua estrutura de componentes computacionais, cada componente computacional podendo ser realizado fisicamente somente por elementos que respeitem as suas características operacionais.

Assim, tendo em vista que a estrutura operacional é o ponto de partida para a determinação da estrutura material com que a máquina se realiza, todo aspecto dos componentes materiais de uma máquina real que se puder determinar a partir da estrutura operacional e das construções operacionais correspondentes, terá também caráter real e necessário na máquina real dada, o que faz com que seja essa a maneira preferencial de determinar os aspectos necessários da estrutura material de uma máquina [9].

Compare-se, por outro lado, esse ponto de vista com o ponto de vista expresso, por exemplo, por E. Dijkstra em relação às estruturas de sofware de uma máquina computadora: *the structure is* **our** *invention and* **not** *an inherent property of the equipment: with respect to the structure mentioned the equipment itself is absolutely neutral* [42] (grifos no original).

Sobre isso, cremos que a idéia de "neutralidade" de uma máquina computadora em relação aos programas que executa é a idéia de que ela exibe uma *indiferença operacional* (no sentido com que se fala, p.ex., de equilíbrio mecânico *indiferente*) pelo estado de atividade em que se encontra.

Disso se seguiria que nós poderíamos colocar essa máquina computadora, a qualquer momento, em qualquer dos seus possíveis estados de atividade, sem resistência da parte dela [10]. De modo algum se seguiria, porém, que nós "inventamos" esses estados, a não ser no sentido trivial de que são estados de um artefato (ver capítulo 3).

É por esse caminho, portanto, que conseguiremos estabelecer a realidade e a necessidade dos níveis de organização de uma máquina computadora, explicitar o

[9] Ver em [99, cap.IV] a afirmação genérica da necessidade da análise funcional orientar a análise estrutural, e em [127, cap.5] a discussão sobre os critérios da divisão anatômica.

[10] Uma conclusão que ainda não se contestou, mas que também não se verificou, e que se baseia apenas no sentido corriqueiro da noção de programação.

caráter construtivo da hierarquia desses níveis, e mesmo exibir o seu mecanismo de construção.

Antes, porém, precisamos resumir e tecer alguns comentários sobre os aspectos principais da noção de nível de organização de máquinas computadoras, tal como ela foi apresentada por Newell em [85, sec.3.1], pois é ela que serve de ponto de partida para nossa elaboração.

6.2.3 A hierarquia dos níveis de organização de Newell

Newell não escapa da ambiguidade embutida no modo usual de empregar o termo hierarquia de níveis de organização. Na caracterização apresentada em [85, sec.3.1], um **nível de sistema computador** (*computer system level*) é tanto uma classe de sistemas computadores que tem em comum um conjunto de aspectos, quanto o próprio conjunto desses aspectos. Ele aponta os seguintes aspectos como caracterizadores de um nível de sistema computador:

- **componentes**: os elementos que detém os funcionamentos primitivos do sistema;
- **meio**: os elementos que são transformados pelas operações do sistema;
- **leis de conexão**: as propriedades das conexões entre componentes (Newell chama de *leis de composição*);
- **leis de funcionamento** as propriedades que determinam o funcionamento de um conjunto interconectado de componentes em função do funcionamento de cada um e do modo como estão interconectados (Newell chama de *leis de comportamento*);
- **sistemas** os conjuntos de componentes interconectados que constituem os elementos característicos do nível.

Na caracterização clássica apresentada em [8], são quatro os níveis de sistema principais dos sistemas computadores digitais, cada nível sendo denominado conforme o tipo de sistema que ele caracteriza intensionalmente e contém extensionalmente: nível dos *dispositivos*, nível dos *circuitos*, nível *lógico* e nível dos *programas*. O nível lógico contém dois sub-níveis, o nível dos *circuitos lógicos* e o nível de *transferência entre registradores*. Por outro lado, o nível dos programas passou a ser denominado *nível dos sistemas de símbolos*, em [84]. É esse o nome que usaremos aqui.

A seguinte lista apresenta o que nos parecem ser as características principais desses níveis de sistema (entre parênteses estão as siglas que usamos para representá-los):

1. **Nível dos dispositivos físicos básicos** (`D`)
   Componentes: *materiais físicos constituintes dos dispositivos físicos básicos*
   Meio: *elementos capazes de fluir através dos dispositivos físicos básicos*
   Leis de conexão: *propriedades de contato dos materiais componentes*

Leis de funcionamento: *propriedades de fluxo dos materiais componentes*
Sistemas: *dispositivos físicos básicos*

2. **Nível dos circuitos** (`C`)
   Componentes: *dispositivos físicos básicos*
   Meio: *fluxos operados pelos dispositivos físicos básicos*
   Leis de conexão: *ligação de fluxos entre os dispositivos físicos básicos*
   Leis de funcionamento: *propriedades digitais de manipulação de fluxos dos dispositivos físicos básicos*
   Sistemas: *portas lógicas*

3a. **Nível dos circuitos lógicos** (`CL`)
   Componentes: *portas lógicas*
   Meio: *bits (sinais lógicos digitais)*
   Leis de conexão: *ligação lógica das portas lógicas*
   Leis de funcionamento: *propriedades lógicas das portas lógicas*
   Sistemas: *circuitos lógicos* (inclusive registradores e operadores lógico-aritméticos)

3b. **Nível de transferência entre registradores** (`RTL`)
   Componentes: *registradores, operadores lógico-aritméticos*
   Meio: *vetores de bits*
   Leis de conexão: *caminhos de transferência*
   Leis de funcionamento: *carregamento de registradores, operações lógico-aritméticas*
   Sistemas: *sistemas digitais* (inclusive memórias e operadores de expressões simbólicas)

4. **Nível dos sistemas de símbolos** (`SS`)
   Componentes: *memórias, operadores de expressões simbólicas*
   Meio: *expressões simbólicas*
   Leis de conexão: *designação, associação*
   Leis de funcionamento: *interpretação algorítmica*
   Sistemas: *computadores*

6.2.4 A questão do nível de configuração

Além desses quatro níveis, Newell identificou um *nível de configuração*, chamado também de *nível* `PMS` (processor-memory-switch).

O nível de configuração não se enquadra exatamente na hierarquia dos níveis de sistema apresentada acima. Nos diagramas que representam essa hierarquia, ele sempre está colocado lateralmente, fora da ordenação dos demais níveis. Newell diz que ele "é mais um puro nível de abstração que um veradeiro nível de sistema" [85, p.97], pensando em atribuir a ele um caráter puramente descritivo, pois tanto o nível de transferência entre registradores como o nível de sistemas de símbolos admitem descrições em termos de estruturas `PMS`.

Nossa interpretação do fato de a colocação do nível de configuração na hierarquia dos níveis de sistema não ter sido estabilizada diverge da interpretação de

Newell. Pensamos que a noção de nível de configuração é tão concreta quanto a dos níveis de organização, e que o não enquadramento adequado do nível PMS na hierarquia de níveis de organização proposta por ele se deve a que este simplesmente não é o seu lugar.

Como indicado na seção 6.2.1 e definido na seção 6.4.2, toda hierarquia de níveis de organização elaborada conforme a proposição de Newell está relacionada termo a termo com uma hierarquia de construções operacionais e, portanto, é uma seriação em função de um "grau de construção" operacional.

A noção de *configuração*, no entanto, não é relativa à noção de construção operacional. Ela é relativa a uma noção de *construção funcional*, no sentido que foi dado ao termo estrutura funcional na seção 3.4.4.

Portanto, a noção de nível de configuração é relacionada à noção de construção operacional que orienta a seriação dos níveis de organização – na medida em que as noções funcionais estão relacionadas com as noções operacionais –, mas não se confunde com ela e, como tal, o nível PMS não pode encontrar lugar na hierarquia dos níveis de organização de sistemas computadores tal como essa hierarquia é entendida usualmente e foi formulada por Newell.

Isso, por si só, nos dá a indicação de que Newell não esgotou a noção de nível de organização, e que se ela comporta, como ele apontou, um aspecto de *construção operacional* e um aspecto de *construção estrutural material*, parece que ela comporta também um aspecto de *construção funcional*, indicado pela "intrusão" do nível PMS no nível de transferência entre registradores e no nível de programas. Em outros termos, o "nível" PMS não é um nível a mais no sistema, mas *um aspecto* a mais de cada nível, mas capaz de ser colocado em uma hieraquia específica [11].

Devemos portanto separar esses três aspectos da noção de nível de organização (estrutural, operacional e funcional) para poder captá-la adequadamente, depois, ao detalharmos o modo como eles se coordenam não só na noção de nível, mas na própria noção de organização.

Porém, como veremos a seguir, só poderemos alcançar de modo completo o sentido dessa coordenação compreendendo a hierarquia dos níveis de organização – e, portanto, a própria organização da máquina – como uma construção. É o que antecipamos no capítulo 3, e que desenvolvemos no resto desta reflexão, até atingirmos a noção de inteligência de máquina no capítulo 9.

6.3 Definição de organização de máquina

Partimos da análise funcional elaborada por Piaget em [85], e que procuramos sintetizar em [37]: *a organização de uma totalidade é a ação dessa totalidade sobre cada uma de suas partes.* Como tal, a organização é o que faz a totalidade ter a forma que tem.

[11] O fato de os níveis dos dispositivos, dos circuitos e dos circuitos lógicos não apresentarem aspectos de configuração PMS não significa ausência de estruturação funcional nesses níveis. Só significa que nesses níveis a estrutura funcional não não se dá em termos de *memórias*, *processadores* e *estruturas de chaveamento*.

Ora, o que faz uma máquina ter a forma que tem é sua construção, isto é, o processo pelo qual ela e cada uma de suas partes resultam ser como são. Assim, usamos o termo organização como sinônimo de construção, e o termo nível de organização como sinônimo de estágio de construção.

Desse modo, a hierarquia de níveis de organização de uma máquina computadora é a própria trajetória da construção dessa máquina, e a transição entre dois níveis da hierarquia é uma transição entre duas etapas da construção.

Notamos, por fim, que uma construção é um desenvolvimento, no sentido da seção 3.4.7.

Então definimos:

> A **organização de uma máquina** é uma sequência de transformações de níveis de organização, entre um nível inicial e um nível final, que preserva os invariantes funcionais encontrados no nível inicial, com cada nível criando novos componentes e novos funcionamentos especializados na máquina e também, possivelmente, novos invariantes funcionais para os níveis posteriores.

Essa definição apresenta as seguintes características:

1. a noção de *nível de organização* ainda precisa ser definida;
2. a organização sendo uma sequência de transformações de níveis, ela se dá através de um conjunto de *operações de transformação de níveis*;
3. a organização, sendo ela própria organizada, não se dá ao acaso das execuções de suas operações, mas sim conforme um *programa de organização*;
4. o programa de organização, ao ordenar as operações de transformação de níveis, ordena as várias etapas da construção;
5. o programa de organização, inicialmente executado exogenamente, a partir de alguma etapa avançada pode vir a ser executado pela própria máquina;
6. o programa de organização pode ser fixo ou pode variar com as condições do ambiente ou com a própria história da organização da máquina.

As situações em que o programa de organização é endógeno e/ou variável são consideradas no capítulo 8. Na próxima seção analisamos as demais caracterísiticas da definição.

6.4 Comentários à definição de organização de máquina

6.4.1 Indicações sobre a natureza dos níveis de organização

A caracterização da hierarquia dos níveis de sistemas que Newell apresenta não supre a carência de construtividade da noção de nível indicada no início do capítulo.

De modo indireto, porém, Newell faz importantes observações a respeito da natureza da construção dos níveis de sistema [85]:

- *That a system has a description at a given level does not necessarily imply it has a description at higher levels. There is no way to abstract from an arbitrary electronic circuit to obtain a logic-level system. There is no way to abstract from an arbitrary register-transfer system to obtain a symbol-level system. This contrasts with many types of abstraction which can be uniformily applied, and thus have a certain optional character (as in abstracting away from the visual appearance of objects to the their masses).* (p.97)

- *Each level is defined in two ways. First, it can be defined autonomously, without reference to any other level. To an amazing degree, programmers need not know logic circuits, logic designers need not know electrical circuits, managers can operate at the configuration level with no knowledge of programming, and so forth. Second, each level can be reduced to the level bellow. Each aspect of a level – medium, components, laws of composition and behavior – can be defined in terms of systems at the next level bellow.* (p.95)

- *Each computer system level is a* specialization *of the class of systems capable of being described at the next lower level.* (p.97)

- *It is not possible to invent arbitrarily additional computer system levels that nestle between existing levels. Potential levels do not become technologies* [12]*, just by being thought up. Nature has a say in whether a technology can exist.* (p.97)

Extraímos dessas observações algumas indicações essenciais para a compreensão da natureza das hierarquias de níveis de organização, em particular de sistemas computadores, mas também – pensamos – de sistemas artificiais em geral. Elaborando essas indicações em consonância com que foi apresentado nos capítulos anteriores e do que antecipamos sobre a construção operacional, obtemos as seguintes idéias sobre as hierarquias de níveis de organização:

1. os níveis tem *um caráter de necessidade* (daí sua existência real e não apenas nominal ou subjetiva);

2. os níveis tem autonomia em relação aos seus níveis imediatamente inferiores, mas também podem ser reduzidos a eles, só que para isso *nem a autonomia pode ser total, nem a redução pode ser completa*;

3. cada nível dá origem a certas estruturações (sistemas de componentes) – *estruturações que são possíveis pelas características desse nível*, mas que certamente *não são necessárias nesse nível* – e que originam o nível imediatamente superior;

[12] Isto é, meios para implementar níveis da hierarquia, portanto níveis da hierarquia eles próprios.

4. cada nível surge de certas estruturações (sistemas de componentes) do nível imediatamente inferior – estruturações que são portanto *necessárias para o nível que surge* e que, por isso, *determinam as possibilidades* de estruturação próprias desse nível.

6.4.2 Indicações sobre a natureza do programa de organização

As indicações recém dadas, por essenciais que sejam, são no entanto apenas características da hierarquia que resultam do seu modo de construção, mas não o programa de construção mesmo.

Para alcançar esse programa, temos de explicitar alguns elementos da construção que ainda não foram explicitados, quais sejam, três modos especiais de organização de componentes que vão além de uma interconexão simples, pois são aquilo mesmo que faz um nível dar origem ao seu nível imediatamente superior.

Caracterizamos esses três *modos de organização* como **operações de interconexão**, dotadas de um caráter polimórfico na medida em que se aplicam aos diversos níveis de sistema da hieraquia, que transformam conjuntos de componentes de um dado nível em componentes do nível imediatamente superior. Damos às duas primeiras operações os nomes de *conexão cruzada de componentes* (ou *cruzamento*, simplesmente) e *conexão circular de componentes* (ou *laço*, simplesmente). À terceira operação chamamos de *conexão agregadora* (ou *agregação de componentes*, simplesmente).

Em cada uma das transições de níveis da hierarquia identificada por Newell reconhecemos tanto a atuação conjunta das duas primeiras operações, cruzamento e laço, quanto a atuação isolada da agregação de componentes.

Para compreender o funcionamento dessas operações, porém, devemos introduzir uma *distinção funcional* na noção de componente que, Newell não introduziu, nem os que o seguiram. Dizemos que um determinado componente de um dado nível de sistema é um **componente operador** se o seu funcionamento é autônomo, isto é, tem seu desenrolar temporal determinado por princípios internos. Dizemos que o componente é um **componente operando** no caso contrário [13].

A **operação de cruzamento** é a operação de interconexão de componentes por realimentação, aplicada a componentes operadores, que faz com que as transições de estados de atividade dos componentes que ela interliga se tornem coincidentes do ponto de vista do nível de sistema que a operação está construindo, de tal modo que nesse nível imediatamente superior as transições de estados de atividade do conjunto dos componentes interconectados só podem

[13] Preferimos falar nesta seção de componentes operadores/operandos, e não de componentes autônomos/heterônomos, reservando o uso do termo autônomo para a caracterização de um particular tipo de componente, no nível do conhecimento (ver seção 6.4.6 e capítulo 9). Outra terminologia adequada seria a de componentes ativos/reativos, mas não em conexão com o termo *sistemas reativos* [74] que apenas denota sistemas com computações infinitas, e que seriam melhor denotados pelo termo *sistemas interativos* (ver capítulo 3).

ocorrer como resultantes de ações, externas a eles, que afetem a todos simultaneamente.

A **operação de laço** é a operação de interconexão de componentes por realimentação, aplicada a componentes operadores, que faz com que o estado de atividade de um componente induza uma mudança do estado de atividade em outro componente conectado a ele, ao longo de uma sequência circular de componentes a qual, pela sua forma, está então permanentemente com pelo menos um dos componentes passando por uma transição de estado de atividade, e de tal modo que no nível de sistema que a operação está construindo, essas transições de estado sejam autônomas.

A **operação de agregação de componentes** se aplica tanto a componentes operadores quanto a componentes operandos, "paralelizando" ou "vetorizando" seus estados de atividade, isto é, dando ao conjunto dos seus estados de atividade a estrutura de processos paralelos [14].

A operação de cruzamento faz com que um conjunto de componentes *operadores* de um dado nível de sistema aquira no nível imediatamente superior o caráter de um componente *operando*, cujo espaço de estados de atividade é a *soma* dos espaços de atividade dos componentes cruzados.

A operação de laço faz com que um conjunto de componentes *operadores* de um determinado nível do sistema conserve o caráter de componente *operador* no nível de sistema imediatamente superior, adquirindo um espaço de estados de atividade que é a *soma* dos espaços de atividade dos componentes enlaçados.

A operação de agregação de componentes faz com que um conjunto de componentes, *operadores ou operandos*, de um dado nível de sistema conserve seu caráter respectivo de *operador ou operando* no nível imediatamente superior, mas adquirindo um espaço de estados de atividade que é o *produto* dos estados de atividade dos componentes agregados.

Num certo sentido, portanto, a agregação de componentes apenas replica de modo combinatorial os aspectos funcionais (operador, operando) da atividade dos componentes de sistema sobre os quais ela é aplicada, enquanto que o cruzamento e o laço é que realizam efetivamente a construção de novidades operacionais e funcionais no conjunto das atividades dos componentes do sistema, mas as três são mutuamente necessárias. Assim, pode-se dizer que o cruzamento *constrói a dimensão espacial* do novo nível, o laço *constrói a dimensão temporal* desse nível, e a agregação uniforme *multiplica as dimensões* de um nível dado.

As três operações de interconexão de componentes, portanto, tem por efeito produzir mudanças na estrutura e funcionalidade do sistema, através de **construções materiais** e **operacionais** dos componentes do nível a que se aplicam.

Elas determinam então, juntamente com operações básicas de interconexão de componentes – dadas pelas leis de interconexão dos diversos níveis, e que participam em cada passo da construção como **interconexões auxiliares** –,

[14] Em [37], distinguimos entre *paralelismo*, onde múltiplos processos são combinados como componentes de um processo global, mas cada um conservando sua identidade, e *concorrência*, onde múltiplos eventos são combinados num único processo global em que não se individualizam necessariamente processos componentes.

não só certas formas básicas de estruturação material e operacional dos componentes construídos (soma/produto de estados de atividade), como acabamos de ver, mas também aspectos da estrutura funcional global do sistema (caráter funcional operador/operando, p.ex.), aspectos que são tratados no capítulo 8.

Com essas indicações gerais, podemos caracterizar o programa de construção da hierarquia de níveis de organização das máquinas computadoras.

6.4.3 Caracterização do programa de organização

A condição de *preservação dos invariantes funcionais* é chave: ela determina as transformações de níveis de sistema que são admissíveis em cada passo de uma construção. Quanto a isso, assumimos o modelo de processo construtivo proposto por Piaget [99]: todo *passo de construção* começa com a *reconstrução* do nível anterior no novo nível que se vai construir (reconstrução realizada nos termos do novo nível), seguida da *criação dos novos elementos* do novo nível (criação realizada sobre os elementos do nível anterior, mas com a forma que eles adquiriram no novo nível, depois da reconstrução).

No caso da hierarquia de níveis de sistema das máquinas computadoras, identificamos então os seguintes elementos no seu *programa de organização*:

- o **invariante funcional inicial das máquinas computadoras** é a existência dos dois caracteres funcionais dos componentes, *operadores* ou *operandos*;
- o **ciclo de construção**, que constitui o *motor do desenvolvimento* da hieraquia, se faz pela repetição de três passos:
  - a **reconstrução do nível anterior** no novo nível se dá pela *inclusão*, no domínio das operações de interconexão do novo nível, dos elementos do nível anterior, o que os transforma em elementos manipuláveis no novo nível;
  - a **expansão combinatória** de cada nível se dá através das operações de *agregação de componentes* aplicadas a componentes operadores ou operandos do nível anterior, recém reconstruído;
  - a **construção de novidades** em cada nível se dá através das operações de cruzamento e laço, aplicadas aos componentes operadores do nível recém expandido pelas agregações.

Todos os níveis clássicos da hierarquia, a partir de `CL` – quais sejam, `CL`, `RTL` e `SS` –, são sistematicamente reconstruídos desse modo, como se mostra a seguir. Também, a construção conduz com naturalidade a níveis que não estão oficialmente integrados na hierarquia clássica, quais sejam, o *nível dos sistemas de símbolos multiprocessados, com regulação funcional e estrutural*, e o *nível dos sistemas com adaptação funcional*, que corresponde a um nível de sistemas abertos [58] (ver capítulo 8).

Por outro lado, essa construção sistemática parece ir além do nível dos sistemas de símbolos e alcança o *nível do conhecimento*, talvez possibilitando com

isso esclarecer, pela sua construtividade, inúmeros aspectos cruciais daquele nível que foram deixados em aberto por Newell ao propô-lo em [85] (ver capítulo 9).

6.4.4 Caracterização geral dos níveis de organização

A construção da hierarquia de níveis de organização nos níveis que são anteriores ao nível dos circuitos lógicos não cai no campo de estudo computacional, já que não há nesses níveis inferiores a noção de autonomia operacional, pois não há fechos operacionais.

Os primeiros fechos operacionais aparecem no nível dos circuitos. Este, portanto, é um nível de passagem em direção aos níveis computacionais: não é ainda um nível computacional porque falta aos seus processos a construtibilidade operacional que caracteriza os processos de computação (capítulo 5) e que só se manisfesta no nível dos circuitos lógicos. Chamamos então aos níveis colocados abaixo do nível de circuitos lógicos de **níveis sub-computacionais**.

Há duas abordagens possíveis a esses níveis sub-computacionais: ou uma abordagem fisicalista estrita (isto é, baseada na dinâmica da matéria e da energia), que procure construir a noção de sinal digital a partir da noção de fluxo existente no nível dos dispositivos [15], ou uma abordagem de *dinâmica da informação* ao estilo de C. Petri que também visa estabelecer a ponte entre o contínuo e o discreto, e que mostra a construção do nível lógico a partir de um nível abstrato de *estruturas de concorrência* [92, p.11] [16].

Note-se, por outro lado, que o caráter sub-computacional de uma estrutura não está ligada à escala dos fenômenos físicos que ocorrem nela, mas sim à sua forma operacional. Por exemplo, em [9] se descrevem estruturas computacionais em escala atômica e molecular, e Petri vincula sua teoria da concorrência diretamente à teoria relativística do espaço/tempo [93].

Aos níveis de organização de máquina cujos componentes apresentam a característica de *autonomia operacional* e cujos meios apresentam a característica de *construtibilidade operacional* chamamos de **níveis computacionais de organização**.

Dividimos os níveis computacionais de organização em dois grandes tipos, em concordância com a análise feita no capítulo 4:

níveis computacionais algorítmicos são os níveis de organização que incluem a hierarquia clássica, cobrindo desde o nível dos circuitos lógicos até o nível dos sistemas de símbolos [84], avançando um pouco além deles.

níveis computacionais não-algorítmicos são os níveis de organização colocados a partir do nível dos sistemas de símbolos dotados de regulação material. Esses são os níveis que originam e constituem o *nível do conhecimento* [85] (ver capítulo 9).

[15] Por exemplo, estudando a transformação das correntes elétricas processadas pelos sistemas não- lineares que são os dispositivos eletrônicos, no caso dos circuitos eletrônicos.

[16] Parece-nos que esta segunda abordagem é mais promissora em termos de coordenação com o estudo da inteligência de máquina, pois ela concerne diretamente (e na verdade *fundamenta*) as teorias da concorrência que são nucleares para esse estudo.

6.4.5 Os níveis de organização algorítmicos

O programa de organização dos níveis computacionais algorítmicos, que apresentamos a seguir, tem por base as operações de agregação, cruzamento e laço, e satisfaz a exigência de conservação do *invariante funcional inicial dos níveis algorítmicos* (a distinção entre operadores e operandos).

A hierarquia de níveis de organização que ele constrói é bastante mais detalhada que a hierarquia tradicional [8]. Os *níveis* tradicionais passam a ser divididos em *sub-níveis* (chamados `/1`, `/2` ou `/3`) e os sub-níveis, em *etapas* (chamadas `(a)`, `(b)` ou `(c)`).

A tabela 6.1 sintetiza os principais níveis da hierarquia assim construída, mostrando tanto os níveis tradicionais quanto os sub-níveis construídos pelo programa, e dá uma caracterização geral de cada sub-nível. As etapas que constituem os sub-níveis não são mostradas na tabela, mas serão detalhadas a seguir.

A sequência das etapas constitui a construção da hierarquia (na tabela, a sequência dos níveis se dá de cima para baixo). Cada sub-nível se completa em três etapas, a menos do nível `CL` que só contém um sub-nível inicial e o nível completo, e que se completa então em duas etapas.

Um sub-nível é dito *completo* quando é fechado para as operações de interconexão por realimentação (cruzamento e laço) e por expansão (agregação). Cada nível se completa quando se completa seu sub-nível mais alto: `RTL/3` para `RTL`, `SS/3` para `SS` (mas `SS/3` já não é algorítmico), enquanto `CL` não tem sub-níveis.

A construção de cada sub-nível completo a partir do sub-nível completo imediatamente anterior se faz pela sucessão de suas etapas, em conformidade com o ciclo de construção da hierarquia, definido acima.

A uniformidade, persistência e consistência do processo de criação de sub-níveis com recurso às três operações referidas anteriormente (a exceção é a completação do nível `CL`, que só requer as realimentações cruzadas e de laço) explica o caráter de necessidade dos sub-níveis, pois exclui a possibilidade de sub-níveis intermediários completos entre os sub-níveis identificados: todo sub-nível intermediário será uma etapa da construção do sub-nível imediatamente seguinte, mas não um nível completo ele próprio. Por extensão, isso explica também o caráter de necessidade dos níveis clássicos, no sentido sugerido por Newell (ver seção 6.4.1).

Claro, considerar que os `RTL/1`, `RTL/2` e `RTL/3` são "sub-níveis" e não "níveis" em si próprios é certamente uma decisão arbitrária que visa apenas manter a terminologia clássica. O que se diz aqui, de outra forma, é que a *existência* deles na construção – vistos como sequências de etapas – não é arbitrária ou casual, mas necessária, dado o programa de organização estabelecido [17].

Concretamente temos então, em relação às máquinas computadoras, as seguintes **etapas da construção** dos níveis de organização de máquina, que são

[17] Nesse sentido, é possível opor os níveis clássicos aos níveis construtivos que damos aqui, e considerar que os níveis clássicos (`CL`, `RTL`, `SS`) são na verdade *classes*, ou *tipos*, de níveis construtivos. É por isso que dizemos, quando `x` é um sub-nível do nível `y`, que `x` *é um nível* `y`.

Níveis	Sub-níveis	Características
CL	CL	Nível dos circuitos lógicos
RTL	RTL/1	Nível RTL com ciclos de controle fixos e armazenamento de dados em registradores
	RTL/2	Nível RTL com ciclo de controle fixo e armazenamento de dados em memórias
	RTL/3	Nível RTL com ciclos de controle variáveis em função de dados armazenados em memórias
SS	SS/1	Nível simbólico uniprocessado
	SS/2	Nível simbólico multiprocessado

Tabela 6.1: Os níveis de organização algorítmicos

comentadas a seguir e cujo detalhamento está no apêndice A [18]:

- Construção do nível `CL` (Fig. A.1)
  O nível `CL` é o nível dos circuitos lógicos. Seus componentes são as portas lógicas.

  1) `CL(a)`: `CL` inicial
     O nível `CL` inicial é aquele cujo funcionamento é dado pelas *expressões booleanas*. Comporta circuitos combinacionais não realimentados.
  2) `CL(a)` ⇒ `CL(b)`: Realimentações em `CL`
     As novidades em `CL` são introduzidas por realimentações entre portas-lógicas. As realimentações introduzem o aspecto de *estado* nos circuitos.
  3) `CL(b)`: `CL` completo
     Os componentes resultantes dessas realimentações tem funcionamento que é ou bi-estável (e são chamados de *flip-flops*) ou oscilatório (e são chamados de *osciladores*).

- Construção do nível `RTL/1` (Fig. A.2)
  O nível `RTL/1` é o nível de transferência entre registradores que comporta circuitos de *lógica aleatória* onde não ocorrem memórias, mas apenas registradores isolados.

  1) `RTL/1(a)`: `CL` reconstruído em `RTL`
     Um flip-flop já é em si um registrador de 1 bit, e um oscilador já é um relógio de uma fase. Portanto, a última etapa do nível `CL` já está contida no nível `RTL/1`
  2) `RTL/1(a)` ⇒ `RTL/1(b)`: Expansão de `RTL/1`
     A expansão de `RTL/1` consiste na formação de componentes de múltiplos bits (registradores, operadores lógicos, seletores, etc.) e na introdução de relógios de múltiplas fases.
  3) `RTL/1(b)`: `RTL/1` expandido
     O nível `RTL/1(b)` é o primeiro nível que tem a aparência típica de nível `RTL`.
  4) `RTL/1(b)` ⇒ `RTL/1(c)`: Realimentações em `RTL/1`
     As realimentações em `RTL/1` produzem o caráter cíclico bem conhecido do funcionamento ao nível de transferências entre registradores. A multiplicidade das fases permite organizar processamentos aritméticos e estabelecer uma coordenação entre o funcionamento interno e sinais externos.
  5) `RTL/1(c)`: `RTL/1` completo
     O nível `RTL/1` é o primeiro nível completo de transferência entre registradores. Corresponde tipicamente aos circuitos `RTL` simples caracterizados como de "lógica aleatória".

[18] A seta ⇒ representa transição entre níveis ou sub-níveis.

- Construção do nível `RTL/2` (Fig. A.3)
  O nível `RTL/2` é o nível de organização em que são introduzidas as memórias internas de dados e as interrupções. Isso requer a complexificação dos ciclos de operações, para comportar os diferentes ciclos de acesso.

  1) `RTL/2(a)`: `RTL/1` reconstruído em `RTL/2`
     O nível `RTL/1` é um nível cujas memórias de dados são ainda registradores isolados. Mas um registrador isolado já é uma memória de 1 posição. Então o nível `RTL/1(c)` já está contido em `RTL/2`.
  2) `RTL/2(a)` ⇒ `RTL/2(b)`: Expansão de `RTL/2`
     A expansão de `RTL/2` consiste na vetorização dos registradores e dos relógios, produzindo as memórias propriamente ditas e ciclos de controle com múltiplas fases, cada fase de ciclo contendo múltiplas fases de relógio.
  3) `RTL/2(b)`: `RTL/2` expandido
     As múltiplas fases dos ciclos permitem organizar o endereçamento às memórias.
  4) `RTL/2(b)` ⇒ `RTL/2(c)`: Realimentações em `RTL/2`
     As realimentações em `RTL/2` permitem a variação das fases dos ciclos. Isso possibilita condicionar o sequenciamentos das operações seja a resultados de operações sobre dados contidos nas memórias internas, seja a sinais de interrupção, o que possibilita o uso de dispositivos externos e o estabelecimento de conexões com outras máquinas.
  5) `RTL/2(c)`: `RTL/2` completo
     O segundo nível `RTL` completo corresponde ao do sistema digital complexo que opera sob controle de um *programa fixo* (com todos os tipos de laços e operações condicionais), mas *implícito* na estrutura do circuito.

- Construção do nível `RTL/3` (Fig. A.4)
  O nível `RTL/3` é o nível em que surge a noção de programa armazenado, executado por uma unidade de execução de instruções. Corresponde ao nível interno de organização da arquitetura de von Neumann.

  1) `RTL/3(a)`: `RTL/2` reconstruído em `RTL/3`
     O nível `RTL/2` é um nível que controla o sequenciamento das operações ainda através de resultados de operações sobre *dados* armazenados nas memórias. Mas esses dados já podem ser vistos como instruções armazenadas em uma memória interna, portanto `RTL/2(c)` já está contido em `RTL/3`.
  2) `RTL/3(a)` ⇒ `RTL/3(b)`: Expansão de `RTL/3`
     A expansão de `RTL/3` consiste na vetorização dos dados armazenados, produzindo subrotinas e estruturas de dados lineares. A indexação desses vetores requer a dualidade dados/endereços, para que os endereços de dados e instruções contidos em vetores possam ser calculados.

3) `RTL/3(b)`: `RTL/3` expandido
O endereçamento indexado é, portanto, desde o ponto de vista da organização de máquina, o primeiro modo de endereçamento que não é direto.

4) `RTL/3(b)` ⇒ `RTL/3(c)`: Realimentações em `RTL/3`
As realimentações em `RTL/3` permitem tratar dados como instruções, e vice-versa, e a chamada de procedimentos. Essa condição é a que permite que programas sejam modificados na memória e então ativados, pois precisam ser tratados como dados durante sua modificação e como programa durante a ativação. Isso possibilita também a autotransformação dos programas.

5) `RTL/3(c)`: `RTL/3` completo
O terceiro nível `RTL` completo corresponde ao nível interno de organização da arquitetura de von Neumann.

- Construção do nível `SS/1` (Fig. A.5)
O nível `SS/1` é o primeiro nível dos sistemas de símbolos. Corresponde ao nível externo da arquitetura de von Neumann. Os sistemas desse nível são uniprocessados. São os primeiros sistemas dotados de *regulação de atividade* (ver capítulo 5).

1) `SS/1(a)`: `RTL/3` reconstruído em `SS/1`
O primeiro passo na construção do nível de sistemas de símbolos é justamente estabelecer os símbolos como entidades. `RTL/3` é um nível em que instruções, dados e endereços são ainda códigos binários. Mas um código binário já pode ser visto como um símbolo, de modo que `RTL/3(c)` já está contido em `SS/1`.

2) `SS/1(a)` ⇒ `SS/1(b)`: Expansão de `SS/1`
A expansão de `SS/1` consiste na vetorização conjunta de programas, dados e endereços em um único tipo de estruturas simbólicas. A inclusão de endereços em estruturas de dados requer endereçamento indireto. O resultado é a produção de estruturas encadeadas de programas e dados.

3) `SS/1(b)`: `SS/1` expandido
O modo de endereçamento indireto é então, do ponto de vista da organização, o segundo modo não direto de endereçamento.

4) `SS/1(b)` ⇒ `SS/1(c)`: Realimentações em `SS/1`
As realimentações em `SS/1` possibilitam que um programa simbólico chame qualquer programa (inclusive a si próprio) independentemente de sua colocação na estrutura de memória e permitem que ele interprete livremente qualquer expressão simbólica (inclusive a própria expressão simbólica que o consitui como programa simbólico), o que implica a autonomia operacional da máquina para a variação de estado de atividade.

5) `SS/1(c)`: `SS/1` completo
   O primeiro nível `SS` completo corresponde ao nível de sistemas de símbolos de Newell [85] e ao nível de linguagem de máquina da arquitetura de von Neumann. É o nível em que aparece a regulação operacional de atividade. É o primeiro nível cujos sistemas se aproximam do funcionamento universal no sentido de Turing [19].

6.4.6 Observações gerais sobre o programa da organização

- o invariante funcional inicial da organização é a *existência de componentes operadores e operandos*; esse é o **invariante funcional característico** dos níveis computacionais algorítmicos da organização de máquina;

- a partir do nível de transferência entre registradores, há um *novo invariante funcional* a ser preservado, que é a *existência de ciclos de operações*; correspondentemente, ao se completar o nível de transferência de registradores, os circuitos adquirem a característica de *unidades complexas operacionalmente autônomas*, dotadas de uma interface bem definida com o ambiente externo;

- a partir do nível dos sistemas simbólicos se estabelece um *novo invariante funcional*, a *variação autônoma dos estados de atividade* dos sistemas de símbolos, que chamamos de *regulação operacional de atividade*; ela é devida à dualidade operador/operando que dá a possibilidade de *auto-aplicação e auto-transformação* dos programas simbólicos; ao se completar o primeiro nível de sistemas simbólicos, os programas de sistema (tipicamente, os sistemas operacionais) adquirem essa característica de sistemas operacionalmente autônomos;

- a dualidade operador/operando se faz acompanhar (é causada por) uma *transformação no mecanismo de construção*, que passa a conter um único operador, o operador de *aplicação*, atuando sobre expressões simbólicas que podem assumir os papéis de operadores e operandos, e as operações de cruzamento, laço e expansão se transformam em duas operações, a *recursão* e o *pareamento* (ou seus equivalentes);

- a partir do segundo nível simbólico `SS/2` (descrito no capítulo 7), se estabelece um *novo invariante funcional*, que é a regulação da variação dos estado de atividade das máquinas computadoras através de suas *interações* com outras máquinas computadoras; com isso, as estruturas das funções sistêmicas se tornam responsivas às demandas dos sistemas que as utilizam, e os fechos de funções sistêmicas adquirem a *capacidade de adaptação operacional* ao ambiente, atingindo o terceiro nível de sistema simbólico;

[19] Apenas se aproximam, porque nenhum sistema finito pode ser universal, nem mesmo no sentido restrito de ter um programa que interprete apenas os programas que ele é capaz de executar.

- esse é, portanto, um nível de passagem para o nível das máquinas computadoras dotadas de fecho teleonômico (isto é, equilíbrio estrutural, operacional e funcional); chamamos de *agentes autônomos* a essas máquinas; o processo de construção sofre *uma nova transformação*, pela qual componentes de sistemas simbólicos se transformam em *um único tipo de componente computacional*, que chamamos de **estruturas cognitivas**; esse próximo tipo de nível é o *nível do conhecimento* (ver capítulo 9).

Capítulo 7

Regulação de máquina

7.1 Observações iniciais

Organização e regulação são duas noções que se aplicam com propriedade aos seres vivos, para destacar os aspectos de estruturação e funcionamento que lhes são essenciais. A regulação condiciona as formas que a organização deve assumir, assim como a organização determina as formas com que a regulação pode se dar.

A partir de Darwin, a ciência aprendeu a entender a organização e a regulação não como situações estáticas, "estados" em que se encontram a estrutura e o funcionamento de cada ser vivo, mas sim como *desenvolvimentos*, como sucessões de estados ou *níveis* de organização e regulação [1].

No caso dos artefatos – pelo menos no caso dos artefatos não vivos, que são os únicos que conhecemos – a tendência do ponto de vista usual é pré-evolucionista: entende-se que a organização do artefato é dada de uma vez para sempre na sua construção ou no seu projeto, e permanece fixa, não afetada pelo seu funcionamento. Da mesma forma, a regulação é pensada como sendo estabelecida na criação do artefato, fixando-se independentemente do seu funcionamento posterior.

Como no caso do pensamento anterior a Darwin, pensa-se na criação do artefato como o ato que determina as formas da organização e da regulação. O projetista, no ato de criação, estabelece as estruturas e as regras de regulação, numa harmonia que deriva de seu conhecimento prévio e completo do modo como o objeto projetado deverá existir no ambiente.

A organização projetada dá os instrumentos requeridos pela regulação do funcionamento do artefato frente às perturbações causadas pelo ambiente, e a regulação mantém o funcionamento do artefato no ambiente exatamente na medida em que a organização lhe possibilita isso.

Por outro lado, de cada artefato se pensa poder perguntar – e ficar sabendo com certeza definitiva – *como funciona? o que faz?* Nenhum artefato deixa de ser como é nem deixa de fazer o que faz – ou mesmo muda o modo de fazê-lo –

[1] A *organização como desenvolvimento* é a idéia elaborada no capítulo 6.

por estar fazendo ou tê-lo feito [2]. Assim, do ponto de vista usual, a associação entre a organização do artefato e o seu funcionamento é bem clara e definida desde sua criação.

Os computadores, porém, parecem problemáticos quando analisados nesses termos. A pergunta *o que faz um computador?* não pode ser respondida de modo simples: do ponto de vista usual, um computador faz tudo aquilo que conseguirmos que faça. Portanto, nos termos da análise dos capítulos 3 e 4, tudo aquilo para o qual ele dispõe de um estado de atividade indicado por um programa. Um computador "executa programas".

Ora, executar programas não é *fazer alguma coisa* no mesmo sentido claro e definido em que, p.ex., registrar imagens fotográficas *é* fazer alguma coisa: compra-se uma máquina fotográfica sabendo-se tudo o que ela faz – fotografias. Compra-se um computador sabendo-se apenas que ele é capaz de executar programas, mas, em princípio, não se sabe quais são esse programas: alguns "caberão na memória", outros não; alguns produzirão "erros em tempo de execução", outros não; alguns serão "eficientes", outros não; mas nem uns, nem outros, podem ser conhecidos antecipadamente a partir apenas da descrição do computador: precisam ser explicitados na sua constituição de estrutura simbólica para então poderem ser analisados.

Chamamos de **regulação operacional** à atividade de selecionar funcionamentos (estados de atividade) dentro da estrutura operacional global de máquina. Podemos dizer então que a questão da regulação operacional do funcionamento das máquinas computadoras, coloca-se em termos diferentes daqueles em que se coloca a questão da regulação para os demais artefatos. A regulação operacional não tem as mesmas características que a conhecida regulação de funcionamentos estudada usualmente na teoria do controle [43]. Neste tipo de regulação, trata-se de *ajustar*, às iniciativas do ambiente, um único funcionamento bem determinado e dotado de uma estrutura operacional local bem definida, enquanto que na regulação operacional trata-se de *escolher* entre funcionamentos com estruturas operacionais locais possivelmente bem diferentes.

A questão da regulação operacional do funcionamento de máquinas computadoras coloca-se assim em termos de uma linguagem de ordem mais alta: não apenas essas máquinas são capazes de diversos funcionamentos diretos no ambiente, dados pelos seus diferentes estados de atividade, funcionamentos que podemos chamar de **funcionamentos de 1ª ordem** – formando um *repertório de funcionamentos* correspondente ao conjunto dos possíveis estados de atividade – como também as máquinas computadoras são dotadas de um funcionamento especial sobre esse repertório de 1ª ordem, a *variação do funcionamento* de 1ª ordem – ou *transição de estado de atividade* – que pode ser executada em função de ações e condições do ambiente, e que chamamos de **funcionamento de 2ª ordem**.

Por outro lado, a autonomia operacional das máquinas implica que a determinação das regulações operacionais de funcionamentos não é exógena, mas sim endógena, ainda que relativa a condições externas, e que, portanto, a regulação

[2] Exceto pelo desgaste material (ver seção 4.4.1).

operacional da máquina computadora, como todo funcionamento de que ela é capaz, é um funcionamento autônomo.

Então, é essa existência de dois níveis relativos de funcionamentos autônomos, o nível dos *funcionamentos autônomos selecionados* e o nível dos *funcionamentos autônomos selecionadores*, que torna específico o problema da regulação operacional de funcionamentos em máquinas computadoras, comparativamente aos demais artefatos.

Chamamos de **regulação operacional de $1^{\underline{a}}$ ordem** à atividade de variar funcionamentos de $1^{\underline{a}}$ ordem em função do ambiente, e notamos que a regulação operacional de $1^{\underline{a}}$ ordem é um funcionamento de $2^{\underline{a}}$ ordem. Vemos então que podemos encontrar máquinas em que ocorrem sucessivamente funcionamentos de ordem $n+1$ que são regulações operacionais de ordem n.

A regulação operacional, enquanto funcionamento que realiza transições de estados de atividade, tem ela própria um *programa de regulação*. Chamamos de **reguladores operacionais** aos programas que organizam as ações dos funcionamentos de regulação operacional. Dizemos também que um regulador que é capaz de realizar regulações de ordem n é um *regulador operacional de ordem* n [3].

Chamamos de **sistema primário de regulação operacional** de uma máquina computadora ao conjunto de todos os reguladores operacionais, de todas as ordens, que se pode encontrar nela. O sistema primário de regulação contém, portanto, o conjunto de todos os sistemas operacionais e interpretadores que a máquina é capaz de por em execução. Dizemos que os reguladores operacionais pertencentes ao sistema primário de regulação são **reguladores operacionais primários**.

Se vê, assim, porque o problema da organização e da regulação das máquinas computadoras se coloca numa ordem mais alta que o desse problema em relação a outros artefatos: a questão da organização e da regulação das máquinas computadoras diz respeito principalmente a um *funcionamento geral regulador* – que é uma regulação operacional de funcionamentos de diversas ordens –, e não a um *particular funcionamento* de $1^{\underline{a}}$ ordem, como acontece com as máquinas em geral.

Um problema que se poderia considerar então, eventualmente, é o de determinar a maior ordem de funcionamento de que uma máquina é capaz e, correlativamente a esse problema, o problema de determinar se uma dada máquina computadora contém um **regulador operacional universal**, isto é, um regulador operacional capaz de por em funcionamento todos os demais reguladores operacionais, de todas as ordens, que a máquina é capaz de executar.

Evidentemente, esse problema está relacionado com a conectividade do conjunto de estados de atividade da máquina para a relação de transformação de

[3] Os sistemas operacionais são o exemplo corrente de programas cuja função é a regulação operacional de $1^{\underline{a}}$ ordem da atividade da máquina. São reguladores de $1^{\underline{a}}$ ordem. Nas chamadas *máquinas virturais* [123], que possibilitam a execução paralela de vários sistemas operacionais, o *programa de controle da máquina virtual* é um exemplo de regulador de $2^{\underline{a}}$ ordem. Note-se, também, que todo interpretador de programas tem uma função de regulação operacional em relação aos programas que interpreta.

estados de atividade: se esse conjunto é desconexo, não há um tal regulador universal. Por outro lado, só um regulador operacional de ordem infinita é capaz de regular a si próprio como funcionamento de ordem imediatamente inferior. Mas nesse caso, a máquina em que ele opera não pode ser uma máquina finita.

Por outro lado, poder-se-ia colocar que um objetivo analítico básico em arquitetura de computadores seria o de caracterizar o sistema de regulação operacional primário que cada máquina contém.

7.2 Crítica da noção de regulação de máquina

A existência da regulação operacional de funcionamentos leva o ponto de vista comum a considerar que ela se acompanha de – ou mesmo se deve a – uma regulação estrutural da organização da máquina computadora e que, assim como o funcionamento continuado faz variar os estados de atividade, assim também ele faz variar a organização da máquina.

A ilusão que possibilita essa visão é a reificação das estruturas de símbolos, isto é, a idéia de que as estruturas de símbolos manipuladas por uma máquina estão "contidas", ou "armazenadas", na máquina, mas tem existência independente dela.

Fala-se, por exemplo, que antes de ter um programa "carregado" na sua memória, a máquina não tem funcionamento definido; que os programas "criam" estruturas de dados e arquivos durante o seu funcionamento; que há procedimentos de "instalação" de sistemas na máquina, etc.

Como vimos no capítulo 3, porém, esse é um ponto de vista interpretativo, que se deixa influenciar pelo realismo dos níveis de organização identificados no capítulo 6 e que, talvez, deva a eles a sua aparência de realidade [4].

Na verdade, o conjunto de estruturas de símbolos que podemos identificar em uma máquina dada é um aspecto dessa máquina e é fixo e determinado pela organização dela: há um só modo de indicar ações na máquina (um só "conjunto de instruções de máquina"), uma só estrutura material fixa e, consequentemente, estruturas operacional e funcional também únicas e fixas, as quais podem ser vistas pela óptica das estruturas físicas de símbolos, mas sem que essas existam independentemente daquelas [5].

Com base nessa imanência das estruturas de símbolos e dos processos que as modificam, a noção de regulação de máquina pode avançar no sentido de uma distinção importante: distinguimos a *regulação operacional* e a *regulação estrutural*. A regulação operacional foi definida anteriormente. A **regulação estrutural** é aquela em que se seleciona, dentro de um conjunto dado de possíveis estruturas operacionais globais, a estrutura operacional global capaz de permi-

[4] Toda a terminologia corrente da engenharia de software é uma terminologia voltada para hipostasiar essa fictícia regulação estrutural da organização, que é de fato uma regulação operacional.

[5] A introdução da noção de *microprogramação* não altera o quadro, apenas permite a *migração vertical de funções* [18] – ou, se se quiser, faz com que a "máquina" seja a estrutura microprogramada – mas não cria propriamente variabilidade na estrutura operacional.

tir a regulação operacional requerida para dar conta das perturbações causadas pelo ambiente que que a máquina se encontra.

A regulação estrutural só se dá pela modificação da estrutura material da máquina, e podemos dizer que toda máquina computadora é capaz de regulação operacional, mas é incapaz de regulação estrutural. A razão para tanto é simples: a máquina computadora é dotada de um fecho operacional, que lhe dá autonomia operacional, mas ela não é dotada de um fecho estrutural.

Quer dizer, sua estrutura operacional global não está submetida a uma estrutura operacional mais ampla que seja capaz de moldá-la através de regulações estruturais – a não ser a ação do projetista, certamente capaz de produzir a regulação estrutural da máquina, mas aí não mais como regulação autônoma.

Cabe então a pergunta sobre se essa é uma limitação inerente aos artefatos, se esse é o limite da autonomia de funcionamento dos objetos artificiais, ou se é possível que os artefatos alcancem a regulação estrutural autônoma, e em que grau. Para decidir a questão, porém, é necessário estabelecer antes, com precisão suficiente, o significado do próprio termo regulação.

7.3 Definição de regulação de máquina

Regular significa submeter a regras. Regras são restrições de possibilidades, inclusive a negação da possibilidade do oposto, portanto, a necessidade. O que é passível de submissão a regras são as ações. Portanto, definimos:

> A **regulação de máquina** é a submissão das ações da máquina à organização de um programa.

Dada uma máquina, podemos considerar sua regulação sob diversos ângulos [6]:

1. *a forma*: circular, compensatória, operatória;
2. *o conteúdo*: operacional, estrutural;
3. *a origem*: inata, adquirida.

7.4 Comentários à definição de regulação de máquina

7.4.1 As formas da regulação

As três formas de regulação (cíclica, compensatória e operatória), que definiremos a seguir, constituem três *níveis* de desenvolvimento da regulação, caracterizadas pela relação que estabelecem entre perturbações e compensações.

A classificação das ações que podem ocorrer em uma máquina em termos de *perturbações* e *compensações* é relativa aos efeitos dessas ações sobre a atividade

[6] O desenvolvimento que segue é uma elaboração baseada em diversas análises da noção de regulação feitas por Piaget (em [99] e [100], p.ex.).

que a regulação está regulando. Ela tem como referência um estado ou conjunto de estados dessa atividade que são considerados *normais* [7].

As perturbações podem provir do ambiente da máquina, mas também podem se originar na própria máquina. É conveniente dizer que as perturbações são *ações positivas*, por produzirem efeitos, e que as compensações são *ações negativas*, por tenderem a anular as perturbações.

Como o caso mais básico, podemos identificar pelo menos um aspecto dos processos computacionais que se justifica como constituinte da noção de normalidade desses processos: a persistência dos processos ao longo do tempo, usualmente chamada **vivacidade** (*liveness*) [105]. Um segundo aspecto que também poderia ser considerado é a velocidade de execução do processo.

Em geral, uma **perturbação** é qualquer forma de *obstáculo* à continuação do funcionamento da máquina e uma **compensação** é qualquer mudança de estado de atividade capaz de permitir a continuidade do funcionamento.

Assim, uma perturbação, no primeiro caso considerado acima, é uma parte da região da estrutura operacional englobando a atividade da máquina e a atividade do ambiente em que a máquina se insere, que leva à quebra daquela persistência do processo. No caso da velocidade, uma perturbação é uma parte da região da estrutura operacional na qual as ações se desenrolam numa escala de tempo incompatível com a persistência do processo considerado.

Em ambos os casos, uma perturbação pode ser vista como uma ação que determina um aspecto particular numa dada região da estrutura operacional global da máquina, enquanto que uma compensação é uma mudança no estado de atividade da máquina que realiza o processo, ou do ambiente em que ela funciona, deslocando o funcionamento da parte da região que atua como perturbação, de modo que o processo seja retomado, ou tenha sua velocidade adequada.

A **regulação cíclica** é aquela que impõe ao conjunto de operações da máquina uma estrutura operacional circular, no sentido amplo de conter ciclos variados de repetição, com posssíveis aninhamentos de ciclos.

A característica essencial da regulação cíclica é a fixidez da organização das ações: o conjunto de operações se estrutura como se fosse um grafo, que pode ser percorrido variadamente e multiplamente em seus caminhos, mas cujo conjunto de nodos e arcos não pode ser alterado, nem em extensão nem em interconexão.

Em consequência, a regulação cíclica é uma regulação apenas no nome: ela não admite perturbações na execução das ações que organiza, pois não prevê compensações para elas. Todas alternativas de percursos no grafo correspondem a modos variados, mas sempre normais – não compensatórios – de realização das ações reguladas.

A forma cíclica de regulação é, por exemplo, a determinada pela noção clássica de algoritmo e, em geral, por toda estrutura operacional fixa. É, portanto, a regulação operacional existente nas máquinas computadoras ao nível dos sistemas de simbolos uniprocessados.

[7] Note-se, no entanto, que há, provavelmente, tantas noções de normalidade de atividade de máquina quantas são as variedades de atividades que se pode encontrar em máquinas.

A **regulação compensatória** é aquela que tem por princípio a admissão de ações perturbadoras e a efetiva realização de ações compensadoras das perturbações que elas produzem.

A regulação compensatória apresenta as seguintes características, que a distinguem da regulação operatória, definida a seguir: a regulação tem caráter temporal, isto é, a compensação se dá ao longo do tempo, e posteriormente à perturbação; a perturbação não é considerada como constituinte do sistema de ações que a regulação compensatória está regulando [8]; a compensação é, em geral, parcial, porque ocorre simultaneamente com a realização de outras ações, o que impede a captação das perturbações em isolamento dos efeitos produzidos por essas outras ações. A regulação compensatória é a regulação estrutural por excelência.

A **regulação operatória** é o limite a que tende o desenvolvimento da regulação estrutural, é o seu nível mais elaborado. Na regulação operatória, o conjunto das perturbações possíveis é incorporado ao conjunto das ações a serem reguladas, como ações constituintes desse conjunto, e as ações reguladas – pelo seu fechamento operatório – se tornam atemporais.

Perturbações e compensações apresentam, então, apenas as características de **ações diretas** e **inversas** dentro do conjunto de ações considerado, sem poderem mais ser classificadas como "positivas" ou "negativas": todas as ações possíveis são parte do conjunto de ações, sua classificação como diretas ou inversas sendo relativa à situação em que são consideradas, e não uma classificação absoluta.

Como as ações reguladas são atemporais, a própria regulação operatória é atemporal, e as compensações se dão *a priori*: toda ação é *a priori* compensada pela sua inversa. Por isso, a compensação pode ser completa, pois em cada situação, a ação inversa se aplica exatamente sobre os próprios elementos e resultados produzidos pela ação direta.

A regulação operatória é a regulação das operações lógico-matemáticas encontradas no pensamento sistemático efetivo [96]. Portanto, a regulação operatória só ocorre em máquinas capazes de funcionamento representativo, realizado através de uma função semiótica intencional [9].

Podemos, então, definir o **problema da regulação de máquina**:

> Dado que as máquinas computadoras são dotadas de regulação operacional, na forma de regulação cíclica, o **problema da regulação** em máquinas computadoras consiste em determinar o quanto elas podem avançar no sentido da regulação estrutural e, portanto, o quanto podem se aproximar da regulação operatória.

[8] Pelo contrário, a regulação existe exatamente para compensar por ações endógenas os efeitos dessas perturbações exógenas.

[9] Ver seção 9.3.4 sobre a possibilidade de as máquinas alcançarem o funcionamento operatório representativo. Ver também a próxima seção.

7.4.2 Os conteúdos da regulação

Num ser vivo, todo funcionamento é uma construção, e toda construção é um funcionamento [99],[100]. Esse é o sentido do termo *autopoiese* [76] (ver seção 3.4.6).

Num artefato, contudo, o ponto de vista usual pensa a construção como um processo anterior ao funcionamento, e o funcionamento levando a um desgaste, não a uma construção.

Nos artefatos, a *regulação da construção* é dada pelo projeto do artefato, as ações reguladas são as *ações de construção* e as regras que ele impõe são as *regras de construção*. O projeto, por sua vez, como tarefa do projetista, está submetido a uma regulação específica por *regras de projeto*, nem sempre só de caráter técnico, mas também econômico, social, cultural, etc.

Imaginar um artefato que se construa ao funcionar é imaginar um artefato cujo funcionamento realize ações de construção, reguladas por um projeto de que está dotado internamente e que é então um programa que organiza as ações de construção.

É possível pensar nas máquinas, portanto, como passíveis de uma construção simultânea com seu funcionamento, e mesmo realizada por esse funcionamento. Nesse caso, a regulação da construção da máquina se torna parte de, ou se confunde com, a regulação de sua atividade.

Não necessariamente, porém, a máquina se torna autopoiética apenas por produzir sua própria construção. Por definição, autopoiese é o funcionamento que *é* a construção de si mesmo e não, como é possível nas máquinas, um funcionamento que *produz* uma construção como resultado [10]. Não é por poder construir-se que os artefatos podem se tornar seres vivos.

O estudo dos *autômatos que constroem* (e mesmo que *se reproduzem*) iniciou com von Neumann mesmo, e foi continuado por outros (ver, p.ex., [5], capítulo 10: *Machines that compute and construct*). Até onde se percebe, porém, esses trabalhos se desenvolveram tendo em conta apenas o caso em que a regulação da construção é operacional, isto é, é dada por um algoritmo que regula o estado de construção da máquina que está sendo construída, mas que mantém fixa a estrutura da máquina construtora.

Também, nas técnicas de programação de inteligência artificial, especialmente nos trabalhos relativos à meta-programação [52], se vê aparecer a regulação operacional, porque se lida nesse nível com programas que são reguladores operacionais. É assim que surge, por exemplo, a questão da "assimilação de conhecimentos" (ver [67], p.ex.), ainda não resolvida adequadamente [11].

Em todos esses casos, como também no caso dos sistemas operacionais, referido anteriormente, tratam-se de regulações circulares, incapazes de compensar perturbações

[10] Maturana & Varela denominam essas duas situações de *autopoiese* e *heteropoiese* (ou *alopoiese*), respectivamente.

[11] É na direção dessa regulação operacional que desenvolvemos algum estudo preliminar sobre o problema da organização da interação simbólica de agentes computacionais com seu ambiente [39], [38], e sobre a pragmática formal de R. Martin [129].

No entanto, é possível pensar em máquinas auto-construídas, com regulações de construção por compensação de perturbações – portanto dotadas de regulação estrutural.

A questão permanece em aberto, porém, a respeito da possibilidade de máquinas dotadas de funcionamentos com regulação operatória. Não se trata de que os funcionamentos operatórios sejam atemporais e, portanto, não realizáveis em máquinas cujo funcionamento é temporal. Ao contrário, os funcionamentos operatórios são funcionamentos reais, realizados por sujeitos reais, no espaço e no tempo – as estruturas operatórias é que são atemporais, não os funcionamentos que elas regulam.

Trata-se, antes, do modo pelo qual a regulação operatória se impõem aos funcionamentos operatórios, em comparação com o modo com que a regulação compensatória se impõe aos funcionamentos compensatórios: esta se impõem pela causalidade material e operacional (ver seção 3.4.4), ao passo que aquela se impõe à consciência do sujeito que realiza tais funcionamentos e, pela determinação consciente do sujeito em concordância com aquela regulação, atua sobre os funcionamentos, restringindo suas possibilidades e determinando suas necessidades (ver também seção 9.3.4).

7.4.3 As origens da regulação

A máquina tendo estrutura material fixa, todos os seus funcionamentos possíveis são pré-formados, inclusive os funcionamentos reguladores que, pela fixidez da estrutura, só podem ser reguladores operacionais. Os reguladores primários, portanto, são inatos.

A aquisição de funcionamentos só se faz pela aquisição de novos componentes materiais, ou pela modificação da estruturação do conjunto desses componentes, isto é, pela modificação das conexões materiais entre eles. Essa aquisição pode, portanto, ser tanto exógena quanto endógena.

Se a máquina é capaz de ações que afetem sua estrutura material, e não só o estado operacional dessa estrutura, então ela é capaz de produção endógena de novas ações, entre elas ações de regulação – não só regulação operacional como também regulação estrutural. Nesse caso, dizemos que a regulação é adquirida.

O problema da regulação de máquina, que é o problema de determinar o quanto as máquinas podem avançar no sentido da regulação estrutural, é também, portanto, o problema de quanto as máquinas podem avançar no sentido da aquisição de regulações.

Por outro lado, convém enfatizar o sentido preciso do termo *aquisição*: o adquirido é adquirido em relação ao que existia anteriormente, não em relação ao exterior. Isto é, a aquisição é uma construção – certamente com elementos importados do exterior, mas organizados internamente – e não uma importação do adquirido como tal.

Portanto, as regulações ou são inatas, ou são construídas, e o problema da regulação é o problema do quanto as máquinas podem avançar na construção de suas próprias regulações.

Nesse sentido, o nível de organização SS/2 é uma etapa essencial no caminho da construção das regulações: ele cria a noção de função realizada por uma máquina em relação a outra(s), abrindo com isso a perspectiva das estruturas funcionais e fechos funcionais, construídos no nível SS/3, onde a noção de adaptação, e portanto de regulação estrutural, estabelece suas raízes, ao formarem-se estruturas funcionais encarregadas da transformação e da aquisição estrutural.

7.4.4 A construção do nível SS/2

- Construção do nível SS/2 (Fig. A.6)
  O nível SS/2 é o nível dos sistemas de múltiplos sistemas de símbolos autônomos e individualizados, que chamamos *máquinas computadoras*, capazes de interação e coordenação operacional dos seus funcionamentos conjuntos.

  1) SS/2(a): SS/1 reconstruído em SS/2
     SS/1 é um nível em que as máquinas computadoras ainda funcionam isoladamente, pois não podem coordenar-se mutuamente por falta de identidade. Apesar disso, interagem com o ambiente imediato através de canais de comunicação de E/S, e são autônomas em relação a ele. Por isso, SS/1(c) já está incluído em SS/2.
  2) SS/2(a) ⇒ SS/2(b): Expansão de SS/2
     A expansão de SS/2 consiste na vetorização das máquinas computadoras e dos canais de comunicação simbólica. A vetorização requer que expressões simbólicas sejam usadas como identificadores dos elementos vetorizados.
  3) SS/2(b): SS/2 expandido
     A expansão de SS/2 permite a nomeação das máquinas computadoras, o que as individualiza. Isso, mais a nomeação dos canais de comunicação, permite a coordenação dos funcionamentos de máquinas que não tem conexões imediatas entre si, através da atividade cooperativa de outras máquinas. As máquinas passam a desempenhar *papéis*, umas para as outras. O nível SS/2(b) é o nível dos sistemas comumente chamados *sistemas distribuídos com funções distribuídas* [122], que em IA costumam ser chamados variadamente de *sistemas de IA distribuída ou descentralizada* [41], *sociedades de agentes* [114], [115] e também *sistemas abertos* [58].
  4) SS/2(b) ⇒ SS/2(c): Realimentações em SS/2
     As realimentações por cruzamento, em SS/2, possibilitam que conjuntos de máquinas computadoras, agindo coletivamente, coordenem seus funcionamentos frente aos funcionamentos de outros conjuntos de máquinas computadoras. Com isso se constitui o que na análise funcional dos sistemas organizados [37] se chama de **funções sistêmicas**, isto é, conjuntos de componentes que realizam coordenadamente uma função para outros componentes e para o sistema

como um todo. As realimentações por laços possibilitam o surgimento de **ciclos de atividade** no conjunto das máquinas, o que marca o surgimento dos **ciclos funcionais** naqueles conjuntos. Se o conjunto de operações de que os programas dispõem inclui operações de variação da estrutura material, a capacidade de regulação funcional se abre à capacidade de *regulação da estrutura material* das máquinas e dos conjuntos de máquinas interconectadas, o que vem a ocorrer no nível `SS/3`.

5) `SS/2(c)`: `SS/2` completo
O nível `SS/2(c)` corresponde ao nível dos sistemas com capacidade de *regulação operacional sistêmica*, incluindo os sistemas com *regulação da distribuição da carga de trabalho* entre as máquinas computadoras que os compõem, e os chamados *sistemas tolerantes a falhas* [4].

Capítulo 8

Adaptação de máquina

8.1 Observações iniciais

Assim como a organização e a regulação, a adaptação é outra noção que se aplica com propriedade aos seres vivos. Da mesma forma que aquelas duas, ela também é melhor compreendida como um desenvolvimento.

Piaget [95] caracteriza a *adaptação* como o equilíbrio entre duas funções gerais do organismo, a *assimilação* e a *acomodação*. A **assimilação** é o processo pelo qual o organismo incorpora estruturas do ambiente a suas próprias estruturas. A **acomodação** é o processo pelo qual as estruturas do organismo são modificadas para possibilitar a assimilação das estruturas apresentadas pelo meio, quando alguma resistência é encontrada.

A **adaptação**, então, é o estado de desenvolvimento do organismo em que toda estrutura que ele encontra no meio pode ser assimilada, e toda acomodação necessária pode ser realizada.

8.2 Crítica da noção de adaptação de máquina

Na visão utilitarista das máquinas, pela qual se determinam antecipadamente os usos pretendidos e se projetam as máquinas estritamente para atender a tais usos, o problema da adaptação de máquina se resolve *em tempo de projeto*: projetam-se as máquinas, os ambientes em que elas vão funcionar, as estruturas dos ambientes que elas poderão encontrar, e as acomodações que elas possivelmente terão de fazer. Desse modo, as máquinas surgem adaptadas graças a uma harmonia pré-estabelecida pelo projetista [1].

Na presente tese, procuramos estabelecer a visão realista da IA. Devemos, portanto, inverter o ponto de vista e perguntar, dada uma máquina com uma arquitetura determinada: *que estruturas pode assimilar? que acomodações é*

[1] Boa parte do livro de Winograd & Flores [136] e do trabalho relativo a sistemas abertos em IA – [58], p.ex. – consiste em apontar as limitações dessa abordagem (ver tambeém seção 2.2.2).

capaz de realizar? Somente então, conhecidas as capacidades adaptativas da máquina, é que se pode perguntar seriamente: *como ela pode ser usada? o que ela pode fazer, num ambiente dado?*

8.3 Definição de adaptação de máquina

Para tanto, é preciso elaborar uma noção de adaptação (portanto, também de assimilação e acomodação) que não envolva noção de uso, ou melhor, que seja neutra em relação aos usos pretendidos da máquina, isto é, em relação às significações que se pode projetar sobre seu comportamento e funcionamento, mas que de fato não atuam sobre eles.

Pensamos que as atuais teorias da concorrência, ou teorias de processos [80], [60], [105], etc., tem um papel chave na constituição da noção intrínseca de adaptação. Em [36] e [37] iniciamos um quadro conceitual onde tal elaboração parece ser possível.

Então, com base naquela elaboração inicial e na conceituação de Piaget [95] definimos:

> A **assimilação** de uma parte de um ambiente por uma máquina funcionando nesse ambiente é a construção de uma estrutura funcional englobando a máquina e essa parte do ambiente.
>
> A **acomodação** de uma máquina a uma nova parte encontrada no ambiente em que ela está funcionando é a transformação da estrutura funcional da máquina de modo a possibilitar a assimilação dessa nova parte do ambiente.
>
> A **adaptação** de uma máquina ao ambiente em que ela está funcionando é a construção, por assimilação e acomodação correlativas, de uma estrutura funcional englobando a máquina e a parte do ambiente com a qual ela interage, de um modo que possibilita a continuidade do funcionamento da máquina no ambiente cada vez que ela encontra uma nova parte do ambiente.

Essas definições apresentam as seguintes características, que serão comentadas na próxima seção:

1. a assimilação é funcional no sentido da seção 3.4.4;
2. a acomodação funcional pode ser de origem estrutural e/ou operacional;
3. a adaptação é uma regulação, operando por compensaçõesestruturais e/ou operacionais;
4. é preciso caracterizar os dois mecanismos da assimilação e acomodação: integração e diferenciação;
5. organização, regulação e adaptação são aspectos diferentes da mesma totalidade e estão, assim, relacionados.

8.4 Comentários à definição de adaptação de máquina

8.4.1 Assimilação funcional

Piaget [95] distingue assimilação material e assimilação funcional: na *assimilação material*, o organismo troca *substâncias materiais* com o meio e incorpora às suas *estruturas materiais* as substâncias que recebe. Na *assimilação funcional*, o organismo troca *ações* com o meio – isto é, *interage* com ele – e incorpora às suas *estruturas de ações* (que Piaget chama de *esquemas*, ou *conceitos*) as estruturas de ações dos objetos sobre os quais ele age.

Na seção 3.4.4 identificamos estruturas material, operacional e funcional em uma máquina que funciona em um ambiente. Aqui, propomos que a noção de piagetiana de *assimilação funcional* seja identificada com o estabelecimento de ligações funcionais entre a máquina e os componentes do ambiente – isto é, a *construção de uma estrutura funcional* englobando a máquina e componentes do ambiente – o que faz simultaneamente dar a esses componentes funções em relação à máquina, e dar à máquina funções em relação ao ambiente.

Uma estrutura funcional de máquina se torna, então, o que Piaget denominou de *esquema de assimilação* e o problema da acomodação, que é o problema da variação dos esquemas de assimilação, se torna o problema da variação funcional da máquina.

8.4.2 Acomodação funcional

Como vimos na seção 3.4.4, a variação funcional é função das variações operacional e estrutural, ocorrendo dentro dos limites destas. Se a variação operacional se dá sem variação estrutural – isto é, apenas em termos de escolha de regiões de funcionamento – a variação funcional também se dará em termos de escolha de estruturas funcionais dentro do espaço de estruturas funcionais disponíveis na máquina, e não de real transformação de estruturas funcionais.

Então, para que a acomodação funcional se realize como transformação de estruturas funcionais, e não apenas como deslocamento entre regiões de uma estrutura funcional existente, é preciso que a máquina modifique de modo essencial sua estrutura material, e isso só pode ser conseguido através de trocas materiais com o ambiente.

Por isso, no domínio das máquinas – pelo menos no sentido dos artefatos operacionalmente autônomos, mas desprovidos da capacidade de produção de significados conscientes, o que pressupõe a intervenção da intencionalidade na organização do comportamento externo e mesmo do funcionamento global (ver seção 9.3.4) – a acomodação funcional, no sentido estrito da variação do conjunto de estruturas funcionais possíveis, pressupõe a variação estrutural e só pode se dar em máquinas que tenham a capacidade de transformar seus componentes e as conexões entre eles.

8.4.3 Adaptação e regulação

A acomodação consiste na transformação das estruturas funcionais por acomodação estrutural e/ou operacional, resultando na viabilização da assimilação de componentes do ambiente que anteriormente não eram assimilaveis. A situação de impossibilidade de assimilação de um componente pode ser vista, portanto, como uma situação de perturbação, e a acomodação que possibilita a assimilação, como uma compensação.

O jogo adaptativo da assimilação e da acomodação pode ser visto, portanto, como um processo de regulação da máquina: a assimilação produz as perturbações, pelo encontro de componentes externos que não são assimiláveis, enquanto que a acomodação produz as compensações, modificando as estruturas funcionais da máquina e permitindo a assimilação daqueles componentes.

É preciso explicitar então os mecanismos da assimilação e da acomodação.

8.4.4 Integração e diferenciação

Piaget [99] estabelece as operações de *integração* e *diferenciação* de esquemas como as operações que levam à regulação de esquemas. Aqui definimos essas operações no contexto computacional.

Por **integração** de duas estruturas funcionais A, B entendemos a reunião de A,B em uma única estrutura funcional C tal que $A \sqcup B \sqsubseteq C$ (isto é, uma integração é uma construção que engloba as estruturas anteriores). Note-se que como $A \sqsubseteq C$ e $B \sqsubseteq C$, a funcionalidade anterior não está perdida na nova estrutura. Por outro lado, C pode apresentar novas funcionalidades resultantes da criação de novos papéis operacionais e de ligações entre papéis de A e B que antes não estavam ligados, e de possíveis novos vínculos causais entre ligações operacionais de A e B, que antes eram causalmente independentes.

Por **diferenciação** de uma estrutura funcional A entendemos a variação funcional de A que produz como resultado uma estrutura funcional A' tal que $A \sqsubseteq A'$ (isto é, uma diferenciação de uma estrutura funcional é uma expansão dessa estrutura). Pela diferenciação de uma estrutura funcional, novas estruturas funcionais são geradas, com papéis e ligações operacionais antes não existentes. Note-se, também, que pelo fato de $A \sqsubseteq A'$, a estrutura funcional anterior A não deixa de existir. Assim como a integração, a diferenciação amplia a capacidade funcional da máquina cuja estrutura funcional ela modifica, e não implica perda de funcionalidade.

Dessa forma, se A_1 é estrutura funcional da máquina M_D que está numa situação de *perturbação* frente à estrutura funcional A_2 do ambiente, de modo que o funcionamento conjunto $A_1 \,||\, A_2$ não é efetivo, a máquina pode *compensar* essa perturbação ou diferenciando A_1 em A_1' de modo que o funcionamento $A_1' \,||\, A_2$ seja efetivo, ou integrando A_1 com B_1 em C_1, de modo que o funcionamento conjunto $C_1 \,||\, A_2$ seja efetivo. No primeiro caso há uma *compensação por diferenciação* de uma estrutura funcional, no segundo caso há uma *compensação por assimilação mútua* de duas estruturas funcionais. Isso mostra que a diferenciação e a integração de estruturas funcionais são formas de acomodação, e

que toda acomodação é uma construção de estruturas.

Por outro lado, sempre podemos pensar a assimilação de um componente de um ambiente por uma máquina funcionando naquele ambiente como uma integração da estrutura funcional da máquina com a estrutura funcional do componente, de modo que mesmo a assimilação externa de componentes pode ser vista como uma integração de estruturas funcionais. Então, integração e diferenciação se tornam as operações básicas sobre estruturas funcionais.

8.4.5 Adaptação, organização, regulação e desenvolvimento

A adaptação de máquina como processo de manutenção da efetividade das estruturas funcionais implica necessariamente a regulação operacional, mas não necessariamente a regulação estrutural, e as possibilidades da adaptação autônoma são determinadas pelas possibilidades de regulação que a organização da máquina admite.

A organização como processo de construção de estruturas materiais, operacionais e funcionais, mantidas funcionalmente efetivas através das correspondentes regulações, tem então na adaptação o seu mecanismo específico de regulação: a organização se regula em função das determinações da adaptação que, em cada momento compensa as perturbações funcionais que impedem a efetividade da organização, e toda compensação funcional é, como visto, uma re-organização (estrutural, ou apenas operacional).

Dessa forma, vemos que não só todo funcionamento é uma construção e toda construção um funcionamento, tal como indicado no capítulo 5, mas também que toda regulação é um desenvolvimento e todo desenvolvimento uma regulação, com a organização e a adaptação sendo respectivamente o aspecto interno e o aspecto externo desse desenvolvimento (ver [99] para essa relação).

8.4.6 Adaptação e computação não algorítmica

A regulação funcional de uma máquina pode se dar por deslocamento no espaço de estruturas funcionais que sua estrutura operacional pode suportar (o que requer deslocamento no próprio espaço operacional, deslocamento que chamamos de *regulação operacional*), ou pode se dar por regulação estrutural, com variação da estrutura material da máquina e consequente variação operacional e funcional.

No caso da regulação funcional por regulação estrutural, o funcionamento da máquina não é mais algorítmico, tal como discutido no capítulo 5. O nível simbólico `SS/3`, descrito a seguir, é o primeiro nível de organização em que a regulação estrutural é possível. É, portanto, o primeiro nível de organização que não é algorítmico e o primeiro nível em que se torna possível a regulação funcional por regulação estrutural e, consequentemente, o nível que embasa a *adaptação* de máquina.

Restrita ao nível da regulação operacional, a máquina é incapaz de promover sua própria organização: seu programa de organização é necessariamente exógeno. Com capacidade de regulação estrutural, a máquina – como se viu –

se torna capaz de um programa endógeno de organização e, portanto, capaz de desenvolvimento auto-regulado e de adaptação autônoma.

Assim, a adaptação e o desenvolvimento sendo diretamente relacionados, o nível SS/3 é também o nível que embasa o *desenvolvimento de máquina*.

8.4.7 A construção do nível SS/3

Se a sequência dos níveis de organização até o nível SS/2 é realista, no sentido de ser uma reconstrução lógica de estruturas existentes, a construção do nível SS/3 apresentada a seguir é especulativa, no sentido de que pretende avançar essa construção para além do que é visível no presente.

Os sistemas do nível SS/3 são pensados como sistemas cuja melhor descrição é a dada pelo termo *organismo artificial*: são estruturas cujos componentes se organizam em funções sistêmicas realizadas por subsistemas, e cujo conjunto de subsistemas suporta funções gerais, em especial a *organização*, a *adaptação* e a *regulação*, que se realizam então endogenamente.

Seguindo [99], chamamos de **invariantes funcionais gerais** a essas funções gerais. Verificamos que elas são os invariantes funcionais criados nesse nível da organização, e que devem ser preservados pelos níveis que possam ser construídos a partir dele.

Para caracterizar os sistemas constituintes desse nível, adotamos o termo **agente autônomo**, no qual se supõe o significado pleno do termo autonomia operacional, pois aqui ela tende a se estender a todos os invariantes funcionais gerais, já que aqui todos eles estão operacionalizados endogenamente nos sistemas.

- Construção do nível SS/3

  1. SS/3(a): SS/2 incluído em SS/3

     Uma função sistêmica (conjunto de máquinas computadoras capaz de realizar um papel funcional para outras máquinas computadoras e/ou para seu ambiente) não é capaz de realizar os invariantes funcionais gerais, mas na medida em que é capaz de regulação material do conjunto de máquinas que a compõe, já é um agente autônomo (ainda que em sentido parcial), e já está incluído no nível SS/3.

  2. SS/3(a) ⇒ SS/3(b): expansão de SS/3

     A expansão de SS/3 resulta na vetorização e consequente nomeação das funções sistêmicas existentes no conjunto de máquinas consideradas, o que permite a troca de **ações funcionais** entre as funções sistêmicas, isto é, a **interação funcional**.

  3. SS/3(b): nível SS/3 expandido

     O nível SS/3(b) é o nível em que as funções sistêmicas estão nomeadas e, portanto, podem operar umas sobre as outras.

  4. SS/3(b) ⇒ SS/3(c): realimentações em SS/3

Os cruzamentos e laços de funções sistêmicas, estabelecendo vínculos entre funções sistêmicas diferentes, estabelecem **fechos funcionais** na estrutura funcional do conjunto de máquinas considerado e possibilitam, com isso, a constituição das funções gerais de *organização*, *adaptação* e *regulação*, que aparecem como os **invariantes funcionais gerais** que devem ser preservados a partir do nível `SS/3`. Secundariamente, os cruzamentos permitem estabelecer mecanismos para a **função de conservação**, a qual se constitui então em uma quarta função geral, que desempenha papel central na regulação do desenvolvimento.

5. `SS/3(c)`: nível `SS/3` completo

   O nível `SS/3` completo é o nível de organização em que se identificam *fechos funcionais* capazes de operar como mecanismos endógenos de organização, adaptação, regulação e conservação, permitindo à máquina desenvolver-se e adaptar-se autonomamente às ações e reações realizadas pelo ambiente, com liberdade de assimilação e acomodação material, operacional e funcional, nos marcos de sua estrutura material e dos recursos materiais, operacionais e funcionais do ambiente que ela é capaz de incorporar às suas estruturas. É o nível dos **agentes autônomos**.

Capítulo 9

Inteligência de máquina

9.1 Observações iniciais

> *I propose to consider the question, 'Can machines think?' This should begin with definitions of the meaning of the terms 'machine' and 'think'. [...] Instead of attempting such a definition, I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.*

Com essas palavras, A. Turing inicia o artigo "Computing Machinery and Intelligence" [125], frequentemente considerado o trabalho inaugural da inteligencia artificial, em que o autor discute a possibilidade de as máquinas virem a pensar quando desenvolvidas suficientemente em sua capacidade de processamento de informações.

Chama a atenção, na passagem citada, a recusa explícita de Turing em dar uma definição direta da noção de inteligência (e mesmo da noção de máquina!). Essa atitude é, na verdade, uma tradição da inteligência artificial na sua vertente artificialista e especialmente na vertente da simulação da cognição humana. Ela consiste, essencialmente, em negar que psicólogos, filósofos, epistemólogos, biólogos, etc., possam ter dado contribuições decisivas à resolução do problema do que é a inteligência e em afirmar que só a abordagem do processamento de informações vai produzir essas contribuições.

Portanto, segue-se que nenhuma definição de inteligência deve ser dada antes que algum sistema de processamento de informações suficientemente sofisticado tenha sido construído de modo a permitir que se lhe atribua inteligência. Só aí se saberá o que é inteligência: aquilo que, na estrutura e funcionamento desse sistema, faz com que ele seja inteligente [1].

O artigo de Turing [125] contribui para essa atitude dando um instrumento de decisão sobre quando o sistema construído alcançou o nível necessário: é o

[1] Essa estratégia de trabalho também se aplica a outros temas estudados nas ciências cognitivas. A. Sloman, por exemplo, aborda o problema do afeto e das emoções dessa forma [117].

teste de Turing, pelo qual se diz que uma máquina *pensa* quando não é possível distinguí-la de um ser humano ao interagir-se simbolicamente com ambos.

Contraste-se, agora, essa atitude tradicional em inteligência artificial com a atitude de Piaget expressa em várias de suas obras, mas especialmente em [95], na seção "Definição da inteligência": nenhum trabalho científico pode avançar metodicamente sem que seu objeto de estudo esteja definido, *o que se impõe, sem dúvida, para delimitar o domínio de que ele se ocupa* [95, p.19].

Assim, se o trabalho artificialista em inteligência artificial não se baseia em uma definição adequada de inteligência humana então em verdade seu domínio de interesse não fica delimitado, e quase tudo na conduta humana passa a ser tema da IA, quando ela procurar artificializar essa conduta. Definir inteligência humana, portanto, é um passo preliminar necessário ao ponto de vista artificialista em IA, do contrário não se saberá exatamente o que artificializar. E isso, mesmo que não haja acordo preliminar entre os pesquisadores a respeito de como delimitar aquela noção.

Do mesmo modo, definir as noções de máquina, inteligência de máquina, e outros conceitos relacionados, tal como feito nos capítulos precedentes, é passo preliminar necessário ao ponto de vista realista em IA, do contrário não se saberá o que observar nas máquinas.

9.2 Definição de inteligência de máquina

Partimos de uma leitura, talvez bastante pessoal, da noção de inteligência expressa por Piaget em várias obras, especialmente [95] e [99] e que tentamos resumir em diversas passagens [23], [27], especialmente na proposta de tese [30]. Nessa condições definimos:

> A **inteligência de máquina** é o termo final do desenvolvimento da estrutura de regulação das interações funcionais da máquina com o ambiente em que está funcionando.

Esta definição apresenta as seguintes características, que serão comentadas na próxima seção:

1. a definição é funcional, deixando em aberto a caracterização estrutural da inteligência;
2. a noção de interação funcional precisa ser definida;
3. a definição enraíza a noção de inteligência de máquina no nível dos sistemas de símbolos mas aponta para o nível do conhecimento;
4. a definição leva em conta que a noção de inteligência envolve operações com valores.

9.3 Comentários à definição de inteligência de máquina

9.3.1 Função e estrutura da inteligência de máquina

O problema preliminar do ponto de vista realista em inteligência artificial, que é o da definição da noção de inteligência de máquina, apresenta dois aspectos principais: por um lado, encontrar um conteúdo concreto objetivo para essa noção; por outro lado, justificar o uso do termo *inteligência* para designar esse conteúdo.

A noção de inteligência de Piaget possibilita resolver esses dois aspectos do problema preliminar da visão realista exatamente dentro dos marcos dessa visão, isto é, sem fazer apelo à artificialização de condutas.

Por um lado, Piaget enraíza sua noção de inteligência na organização e na dinâmica do organismo do indivíduo, fazendo da inteligência uma *adaptação biológica*, isto é, um prolongamento das estruturas e funções existentes no organismo (o que estende as raízes da inteligência até os processos evolutivos). Por outro lado, ele distingue na inteligência a *função* que ela exerce no organismo e a *estrutura* de que está constituída para realizar essa função.

A idéia do enraizamento da inteligência na organização e na dinâmica do organismo é exatamente o que permite resolver o primeiro aspecto do problema preliminar do ponto de vista realista em IA: o conteúdo concreto e objetivo da noção de inteligência de máquina deve ser buscado na organização e na dinâmica da máquina, e não nas projeções semânticas que se possam fazer sobre ela, ou em usos possíveis que se possam fazer dela.

A idéia de que a inteligência é uma estrutura que cumpre uma função é exatamente o que permite resolver o segundo aspecto do problema preliminar do ponto de vista realista em IA: na medida em que a inteligência tem uma definição funcional, toda estrutura de qualquer sistema que realize uma função correspondente a essa nesse sistema pode também, em princípio, ser chamada de *inteligência* desse sistema. Quer dizer, é o aspecto funcional da inteligência, e não seu aspecto estrutural, que justifica falar simultaneamente em inteligência humana, animal, vegetal ou de máquina, independentemente das (certamente muito grandes) diferenças de estrutura que essas inteligências possam apresentar.

Resta resolver, ainda, em relação à inteligência de máquina, o problema que Piaget apontou e resolveu com clareza em relação à inteligência humana: se a inteligência é uma estrutura cuja função é regular as interações funcionais do indivíduo com o meio, nem toda estrutura que cumpre essa função é uma estrutura de inteligência. Caso contrário, os reflexos inatos e os instintos seriam estruturas de inteligência, e a noção de inteligência perderia sua especificidade (ver [95], seção "Definição de inteligência"). A inteligência é a estrutura que regula as interações funcionais que são ao mesmo tempo *adquiridas* e *intencionais*.

Porém, como a inteligência se enraíza nas estruturas cognitivas inatas, não é possível opô-la completamente a elas, e a solução consiste em encontrar um

termo, no curso do desenvolvimento da inteligência, que possa ser considerado como intermediário entre o inato e o adquirido, e entre o não intencional e o intencional, e estipular que as estruturas da inteligência são aquelas que se desenvolvem *a partir* desse termo intermediário.

Piaget identificou os três primeiros estágios do desenvolvimento sensoriomotor [94] como esse termo de passagem: o primeiro estágio é o das condutas inatas; o segundo estágio é o das condutas que, embora já adquiridas, ainda não são intencionais; e o terceiro estágio é aquele em que se iniciam as condutas adquiridas e intencionais, portanto aquele em que a estrutura de regulação das condutas justificadamente pode ser chamada de *inteligência*.

Ora, esse problema de determinar o nível de desenvolvimento estrutural a partir do qual se justifica o uso do termo inteligência em um sentido estrutural, e não apenas funcional, esse problema não se coloca ainda para a inteligência de máquina, pela simples razão de que ainda não é uma atitude realista atribuir-se intencionalidade às máquinas.

Nessa situação, portanto, e desde o ponto de vista de uma análise que toma o trabalho de Piaget como quadro conceitual de referência, o uso do termo *inteligência de máquina* não pode ser justificado enquanto o desenvolvimento técnico não possibilitar a atribuição de intencionalidade às máquinas. Pode, quando muito fazer-se uso do termo *inteligência de máquina* no sentido funcional – o que já é um uso claro e justificado, e que já aponta para uma perspectiva de desenvolvimento futuro – mas não no sentido estrutural.

Quer dizer, no estágio atual do desenvolvimento da ciência da computação, o ponto de vista realista em inteligência artificial deve contentar-se com uma situação intermediária, em que a possibilidade de máquinas capazes de condutas adquiridas já pode ser vislumbrada (ver o nível de organização `SS/3`, seção 8), mas a intencionalidade que permite especificar as estruturas propriamente de inteligência nem sequer dá indicações de ser possível.

Em outros termos, a noção de inteligência de máquina só se mostra, no presente, ao nível do que podemos chamar de **pré-inteligência**, e o trabalho em inteligência de máquina tem necessariamente de se restringir, por enquanto, ao problema da aquisição não intencional de condutas.

9.3.2 A noção de interação funcional

Como vimos, Piaget [95, cap.1] distingue entre as *trocas materiais* e as *trocas funcionais* que um organismo realiza com seu ambiente: as trocas materiais são claramente definidas como aquelas em que o organismo e o ambiente trocam *substâncias materiais*. As trocas funcionais, porém, não recebem definição explícita precisa.

A intenção de Piaget, porém, é clara em diversas passagens: as trocas funcionais envolvem ações, e entrelaçamento de ações, por parte do organismo e do ambiente, e são portanto de cunho psicológico (com o organismo constituindo-se como sujeito, e o ambiente como objeto) e não mais de cunho biológico, embora ainda capazes de função biológica (na medida em que o comportamento tem função biológica).

O termo específico que Piaget utiliza para caracterizar a participação do sujeito nas trocas funcionais que ele realiza com o objeto é o termo *conduta*. Conduta é mais que *comportamento*, não só porque diz respeito a processos internos, que a visão comportamentalista esquece, como também porque diz respeito à totalidade da ação do indivíduo e não só à sequência de atos que ele realiza numa situação determinada.

Vê-se então que o termo *funcional*, em Piaget, pode ser explicado por uma combinação de dois sentidos: de um lado, ele significa *operacional*, e de outro, *funcional*, no sentido que demos a esses dois termos na seção 3.4.4. Isto é, uma troca funcional é uma *interação operacional*, no sentido do entrelaçamento de ações (eventos) devidas ao sistema e ao seu ambiente, mas também é uma *interação funcional*, no sentido do estabelecimento recíproco de funções, o sistema realizando funções para o ambiente e o ambiente (i.é, seus componentes) realizando funções para o sistema.

Assim, definimos **interação funcional** (de uma máquina com seu ambiente) como a interação em que se dá o estabelecimento recíproco de funções entre a máquina e o ambiente.

Verificamos, então, que a noção de *inteligência de máquina* refere exatamente à estrutura (operacional e funcional) da máquina encarregada de realizar a regulação da interação funcional que ela mantém com o ambiente.

9.3.3 O nível do conhecimento segundo A. Newell

Em [85], A. Newell apresenta a noção de *nível do conhecimento*, pela qual se avança na hierarquia de níveis de organização estabelecida em [8] e reformulada em [84] com a caracterização do nível dos sistemas de símbolos.

O nível do conhecimento, diz Newell, difere dos que o precedem na hierarquia pelo fato de que ele não dispõe de leis de conexão (ver seção 6.2.3).

Os componentes do nível do conhecimento são *objetivos* e *ações* (e *corpos* que realizam ações). Os sistemas do nível do conhecimento são os *agentes*. O meio é o *conhecimento*. A lei de funcionamento é o *princípio da racionalidade*: ações são selecionadas para atingir objetivos, conforme indicado pelo conhecimento disponível.

O nível do conhecimento, porém, não contém leis de conexão:

> *the specification at the knowledge level is provided entirely by the content of the knowledge and the goals, not by any structural way they are connected together.* [85, p.99]

Portanto, diz Newell, o nível do conhecimento não contém estrutura:

> *the gross anatomical description of a knowledge-level system is simply (and only) that the agent has as parts bodies of knowledge, goals and actions. They are all connected together in that they enter into the determination of what actions to take. This structure carries essentially no information.* [85, p.99]

9.3.4 A questão do conhecimento em sistemas artificiais

É interessante notar que Piaget, embora dando uma definição clara e precisa de *inteligência*, tal como discutido ao longo do presente trabalho, não se preocupou em dar uma definição de *conhecimento*. Ao contrário, em [97] (seção: Objeto e métodos da epistemologia genética), ele argumenta que a noção de conhecimento é, em si, excessivamente ampla para poder ser definida – pois contém uma referência inevitável à totalidade da experiência humana.

Então, Piaget propõe que a pergunta *o que é o conhecimento?* seja substituída pela pergunta *como crescem os conhecimentos?*, na certeza de que a perspectiva genética possibilite a objetivação dos procedimentos de análise epistemológica e, portanto, seu controle.

No estudo do crescimento dos conhecimentos, ao longo de toda sua obra, Piaget se concentra no estudo do desenvolvimento da *forma da ação*, nas estruturas operacionais e operatórias que moldam a ação do indivíduo. Isto está de acordo com sua formulação, em [95] (capítulo: Inteligência e Adaptação Biológica), em que o *aspecto cognitivo* das condutas é identificado com seu *aspecto estrutural*, e o *aspecto afetivo* delas, com seu *aspecto energético*.

Assim, definimos que o **conhecimento** de uma máquina é o conjunto das estruturas operacionais e operatórias que organizam seu funcionamento.

Ao contrário de Newell, portanto, que não reconhece estrutura no nível de conhecimento, reconhecemos que o nível de conhecimento é um nível essencialmente estrutural, e definimos que os *componentes*, os *sistemas*, e o *meio* do nível do conhecimento são todos constituídos de estruturas operacionais e operatórias, umas atuando sobre as outras, segundo *leis de interconexão* e *leis de funcionamento* essencialmente operatórias, isto é, lógico-algébricas.

Não só na sua forma, no entanto, mas também no seu no aspecto energético da ação, isto é, no aspecto dos valores envolvidos na ação, há estrutura: o próprio domínio dos valores admite estruturação [98]. Portanto, também em relação à questão da orientação geral da ação, a questão da estrutura está presente no nível do conhecimento.

Por outro lado, Piaget estabelece pelo menos três distinções no domínio dos conhecimentos de um indivíduo, todas já referidas anteriormente: a distinção entre o *inato* e o *adquirido*; a distinção entre o *sensorio-motor* e o *representativo*; e a distinção entre o *uso* (ou, o *exercício*) dos conhecimentos e sua *construção*. Também, Piaget distingue nas ações capazes mobilizar um conhecimento, as *ações intencionais* e as *ações não intencionais*.

O que essas diversas distinções – ou, como preferimos chamá- las, **dimensões do conhecimento** – nos colocam, é o problema de saber onde, no espaço dos conhecimentos construíveis, está o lugar que os sistemas artificiais podem alcançar.

Com a análise do nível `SS/3`, concluímos que esse espaço é o dos conhecimentos inatos, avançando em direção aos conhecimentos adquiridos por construção endógena, mas sem perspectivas de alcançar – pelo menos no estágio atual da técnica – o campo dos conhecimentos relativos a ações intencionais.

Ora, os conhecimentos representativos, sendo baseados no uso intencional da

linguagem [2] e da função semiótica que a sustenta, fica claro que o conhecimento representativo também não pode ser alcançado pelas máquinas no estágio atual da técnica [3].

Em resumo, as máquinas atuais parecem amarradas ao nível sensorio-motor do conhecimento, e especificamente às fases que antecedem o nível sensorio-motor intencional. Quer dizer, toda aquisição de que possam ser capazes se dá por uma constatação *a posteriori* do interesse que ela pode apresentar, e não por uma antecipação, criativa ou por experimentação exploratória, do novo conhecimento, posto que isso supõe a intencionalidade.

Cremos que o reconhecimento desses fatos coloca a perspectiva realista em inteligência artificial dentro de um caminho menos ambicioso e aventuresco, porém certamente mais controlável em sua objetividade, que o caminho normalmente seguido pela artificialização, tal como ela tem sido feita até agora. Em particular, é um caminho que aproxima de modo sistemático a formalização e a experimentação empírica, colocando a inteligência de máquina como tema da arquitetura de máquina e, por isso mesmo, como elemento capaz de polarizar e catalisar a sistematização dedutiva dessa área que, tradicionalmente, procede apenas por analogia e acumulação histórica de inovações bem sucedidas [19].

9.3.5 A questão dos valores

Cremos que, como os significados, os valores se originam nas ações (ver [2] para uma análise nesse sentido). Como os significados, os valores dizem respeito a como algo se insere numa ação – ou melhor, num esquema de ação – ou num sistema de conceitos. Como a atribuição de significados aos objetos, a atribuição de valores aos objetos é sempre relativa às ações que se aplicam sobre esse objetos, e assim como os principais significados com que lidamos se referem a ações (a linguagem), os principais valores com que lidamos se referem também a ações (a conduta social).

Assim como os significados, que vão se tornando cada vez mais abstratos à medida em que os objetos a que se referem são manipulados por intermédio de ações de ordem reguladora cada vez mais alta – a ponto de o significado

[2] Muitos argumentariam sobre o caráter redundante dessa afirmativa: toda linguagem é intencional *por definição*.

[3] Pelo menos enquanto não se vislumbrar o modo de construção da intencionalidade a partir da dinâmica do sistema, como se concebe a construção da inteligência a partir dessa dinâmica (e da própria intencionalidade). A formulação mais direta desse problema em relação à inteligência artificial foi a feita por J. Searle, no artigo já citado [113], na forma de uma pergunta sobre a possibilidade da *intencionalidade* nas máquinas. A resposta que ele mesmo deu (*Could a machine think? My own view is that* **only** *a machine could think, and indeed only very special kinds of machines, namely brains and machines that had the* **same causal powers** *as brains.*) não exclui a possibilidade do caráter artificial dessas máquinas intencionais, mas exige que elas tenham uma natureza biológica. Na medida em que a consciência, como a inteligência [99], for um fenômeno biológico, a resposta de Searle estará correta. Aqui, evidentemente, procuramos estabelecer que a inteligência não é um fenômeno exclusivamente biológico, mas não nos manifestamos sobre a questão da consciência. Portanto, deixamos em aberto a possibilidade de os artefatos não vivos se desenvolverem até o nível do funcionamento operatório.

aparentemente desaparecer, no caso das ações puramente lógicas, de caráter totalmente formal – também no caso dos valores parece que quanto mais reguladoras são as ações que envolvem tais valores, mais esses valores se tornam abstratos – valores sociais gerais que orientam a atividade do sujeito – e, como os significados, só se tornam acessíveis ao pensamento reflexivo.

Por outro lado, porém, assim como os objetos concretos encontram significação operacional na organização das ações sensorio- motoras concretas (como habilitadores ou desabilitadores dessas ações, p.ex.), os valores também encontram lugar nos esquemas de ações concretas, na forma de elementos energéticos iniciadores e mantenedores da continuação da ação, especialmente diante de obstáculos que exigem variação na estrutura operacional em utilização.

Também, assim como a atribuição de significados, a atribuição de valores pode ser feita com base em estruturas adquiridas ou inatas.

Por outro lado, se os signficados podem ser explicados em termos das estruturas funcionais dos esquemas em que se inserem os objetos a que dizem respeito (p.ex., na forma dos papéis que esses objetos desempenham naquelas estruturas), os valores só podem ser explicados pelas estruturas relacionais que o indivíduo consegue estabelecer entre essas estruturas funcionais, na forma de comparações e correspondências que ele realiza. Por isso, a noção de valor supõe a noção de conservação, pois todo valor refere uma experiência anterior, diante da qual o objeto atual é valorado.

Vê-se então que a noção de valor não pode encontrar lugar na organização de máquina antes do nível `SS/3`, e que portanto a adaptação que é possível nesse nível está totalmente ligada à experiência presente, só podendo proceder, por isso, atravé de tateios, de tentativa e erro. Só com a conservação estabelecida e integrada na organização da máquina é que as **relações de valor** podem se tornar operacionais e participar da adaptação.

Assim, posto que a intencionalidade é condição de possibilidade dos valores racionalizados, a *questão dos valores* que se coloca para a inteligência de máquina no estado atual da técnica, é exatamente paralela à questão dos significados: saber o quanto a máquina pode avançar na construção de valores que não sejam inatos, mas que ainda não necessitem, para seu uso e elaboração, da participação da intencionalidade.

A medida da possibilidade das construções cognitivas endógenas de uma máquina será exatamente a medida das suas construções axiológicas endógenas, e portanto seu grau de autonomia cognitiva será exatamente seu grau de autonomia axiológica [4].

[4] O apêndice B traz as elaborações posteriores acrescentadas ao esboço formal preliminar formulado em [27]. Em especial, mostra-se o espaço que os valores podem ocupar na organização da inteligência de máquina.

Capítulo 10

Conclusão

Esta tese estabeleceu um ponto de vista pelo qual se pode atribuir um conteúdo objetivo à noção de inteligência de máquina. O princípio da autonomia operacional das máquinas rege todas as ações de que elas são capazes, incluindo a construção de sua organização, a elaboração de estruturas funcionais, a regulação de sua atividade e a adaptação de suas estruturas operacionais e funcionais às correspondentes estruturas do ambiente em que a máquina funciona.

O princípio da autonomia operacional também atribui um caráter histórico ao funcionamento das máquinas, o qual – em conjunto com a função de conservação – permite a construção de valores e, com eles, o surgimento dos processos de desenvolvimento.

A inteligência de máquina, como termo final do desenvolvimento da estrutura de regulação das interações funcionais, aparece então como estrutura real, possível de ser estudada formal e empiricamente, e suas características, assim identificadas, podem embasar a procura por uma melhor resposta à questão *o que os computadores podem fazer?*

Porém, o quanto as máquinas avançarão concretamente na direção desse desenvolvimento que se pode conceber racionalmente, não depende da vontade de quem as constrói, mas da natureza que lhes é própria.

Apêndice A

Construção dos níveis de organização algorítmicos

A.1 Construção do nível CL

1) `CL(a)`:
(`CL` inicial)

⊧O nível `CL` inicial é aquele cujo funcionamento é dado pelas "expressões booleanas".⊧

operadores: portas-lógicas
operandos: caminhos de sinais de 1 bit
meio: bits

2) `CL(a)` ⇒ `CL(b)`:
(Realimentações em `CL`)
⊧As novidades em `CL` são introduzidas por realimentações entre portas-lógicas.⊧

cruzamentos: flip-flops
laços: osciladores
interconexões auxiliares: seletores de 1 bit

3) `CL(b)`:
(`CL` completo)
↯Os componentes resultantes dessas realimentações tem funcionamento bi-estável (flip-flops) ou oscilatório.↯

operadores: osciladores
portas-lógicas
operandos: flip-flops
seletores de 1 bit
caminhos de sinais de 1 bit
meio: bits

A.2 Construção do nível RTL/1

1) RTL/1(a):
(CL reconstruído em RTL)
⊧Um flip-flop já é em si um registrador de 1 bit, e um oscilador já é um relógio de uma fase.⊧

operadores: relógios de 1 fase
operadores lógicos de 1 bit
operandos: registradores de 1 bit
seletores de 1 bit
barramentos de 1 bit
meio: vetores de 1 bit

2) RTL/1(a) ⇒ RTL/1(b):
(Expansão de RTL/1)
⊧A expansão do nível RTL/1 consiste na formação de componentes de múltiplos bits e na introdução de relógios de múltiplas fases.⊧

agregações:
vetorização de:
registradores
seletores
operadores lógicos
barramentos
interconexões auxiliares:
codificadores/decodificadores
divisores de frequência de relógios

3) `RTL/1(b)`:
(`RTL/1` expandido)
⊧A expansão de `RTL/1` é o que produz os "registradores" propriamente ditos.⊧

operadores: relógios de múltiplas fases
codificadores/decodificadores
operadores lógicos de múltiplos bits
operandos: registradores de múltiplos bits
seletores de múltiplos bits
barramentos de múltiplos bits
meio: vetores de múltiplos bits

4) `RTL/1(b)` ⇒ `RTL/1(c)`:
(Realimentações em `RTL/1`)
⊧As realimentações em `RTL/1` produzem o caráter cíclico típico do funcionamento das "transferências entre registradores". A multiplicidade das fases permite organizar processamentos aritméticos e coordenar funcionamento interno e sinais externos.⊧

cruzamentos: separação entre blocos e sinais de dados
e blocos e sinais de controle
laços: sequenciamento cíclico de operações
(ciclos fixos, múltiplas fases de relógio)
interconexões auxiliares: sinais de entrada e de saída

5) `RTL/1(c)`:
(`RTL/1` completo)
⊧O nível `RTL/1` é o primeiro nível completo de transferência entre registradores. Corresponde tipicamente aos circuitos `RTL` simples caracterizados como de "lógica aleatória".⊧

operadores: sequenciadores de operações
(ciclos fixos com múltiplas fases de relógio)
operadores lógicos
codificadores/decodificadores
operadores aritméticos
sinais de E/S
operandos: registradores
seletores
barramentos
meio: vetores de múltiplos bits

A.3 Construção do nível RTL/2

1) RTL/2(a):
(RTL/1 reconstruído em RTL/2)
♮O segundo passo no desenvolvimento dos circuitos RTL é a introdução das memórias de dados. RTL/1 é um nível cujas memórias de dados são ainda registradores isolados.♮

operadores: unidades de controle
(ciclos fixos com múltiplas fases de relógio)
unidades lógico-aritméticas
sinais de E/S
operandos: memórias de dados de 1 posição
seletores
barramentos
meio: dados binários codificados

2) RTL/2(a) ⇒ RTL/2(b):
(Expansão de RTL/2)
♮A expansão de RTL/2 consiste na vetorização dos registradores e dos relógios, produzindo as memórias propriamente ditas e as fases dos ciclos de controle, cada cada fase de ciclo com múltiplas fases de relógio.♮

agregações:
vetorização de memórias de 1 posição
sequenciamento cíclico de operações
(ciclos fixos com múltiplas fases de ciclo)
interconexões auxiliares:
endereçamento de memórias

3) `RTL/2(b)`:
(`RTL/2` expandido)
⊧As múltiplas fases dos ciclos permitem organizar o endereçamento a memórias.⊧

operadores: unidades de controle
(ciclos fixos com múltiplas fases de ciclos)
unidades lógico-aritméticas
operandos: memórias de dados de múltiplas posições
barramentos
sinais de E/S
meio: endereços binários codificados
dados binários codificados

4) `RTL/2(b)` ⇒ `RTL/2(c)`:
(Realimentações em `RTL/2`)
⊧As realimentações em `RTL/2` permitem a variação das fases dos ciclos. Isso possibilita condicionar o sequenciamentos das operações seja a dados das memórias internas, seja a sinais de interrupção o que possibilita as memórias externas e conexões com outras máquinas.⊧

cruzamentos:
condicionamento das fases de ciclos a dados da
memória interna e a sinais de interrupção
laços:
sequenciamento cíclico de operações
(ciclos variáveis com múltiplas fases de ciclos)
interconexões auxiliares:
interfaces de E/S

5) `RTL/2(c)`:
(`RTL/2` completo)
⊧O segundo nível `RTL` completo corresponde ao do sistema digital complexo que opera sob controle de um programa fixo, *implícito* na estrutura do circuito.⊧

operadores: unidades de controle
(ciclos variáveis com múltiplas fases de ciclos)
unidades lógico-aritméticas
interfaces de E/S
operandos: memórias internas de dados
memórias externas de dados
barramentos
meio: endereços e dados binários codificados

A.4 Construção do nível RTL/3

1) RTL/3(a):
(RTL/2 reconstruído em RTL/3)
⊧O terceiro passo na construção do nível RTL é a introdução da noção de programa armazenado. RTL/2 é um nível que controla o sequenciamento das operações ainda através de *dados* armazenados nas memórias.⊧

operadores: controladores de instruções
operandos: operadores lógico-aritméticos
memórias internas de instruções
memórias internas de dados
memórias externas de dados
dispositivos de E/S
meio: instruções, dados e endereços

2) RTL/3(a) ⇒ RTL/3(b):
(Expansão de RTL/3)
⊧A expansão de RTL/3 consiste na vetorização separada das instruções e dos dados armazenados, produzindo subrotinas e estruturas de dados lineares. A indexação desses vetores requer a dualidade dados/endereços.⊧

agregação:
vetorização de instruções e dados
interconxões auxiliares:
dualidade dados/enderecos

3) `RTL/3(b)`:
(`RTL/3` expandido)
⊧A dualidade dados/endereços permite a manipulação dos índices de vetores de dados e instruções.⊧

operadores: controladores de instruções
operandos: operadores lógico-ariméticos
memórias de vetores de instruções
memórias de vetores de dados
dispositivos de E/S
meio: vetores de intruções
vetores de dados/endereços

4) `RTL/3(b)` ⇒ `RTL/3(c)`
(Realimentações em `RTL/3`)
⊧As realimentações em `RTL/3` permitem tratar dados como instruções, e vice-versa.⊧

cruzamentos:
leitura/escrita em memórias de programas
laço:
execução de vetores de dados/endereços como vetores de instruções
interconexões auxiliares:
ambiguidade instruções/dados/endereços

5) `RTL/3(c)`:
(`RTL/3` completo)
⊧O terceiro nível `RTL` completo corresponde ao nível interno de organização da arquitetura de von Neumann.⊧

operadores:
controladores de vetores de instruções
operandos:
operadores lógico-aritméticos
memórias internas e externas de vetores de
instruções e de dados
dispositivos de E/S
meio:
vetores de instruções/dados/endereços

A.5 Construção do nível SS/1

1) SS/1(a):
(RTL/3 reconstruído em SS/1)
¦O primeiro passo na construção do nível de sistemas de símbolos é justamente estabelecer os símbolos como entidades. RTL/3 é um nível em que instruções, dados e endereços são ainda códigos binários.¦

operadores:
unidades de execução de programas simbólicos
operandos:
memórias de programas e dados simbólicos
canais simbólicos de E/S
meio:
instruções, dados e endereços simbólicos

2) SS/1(a) ⇒ SS/1(b):
(Expansão de SS/1)
¦A expansão de SS/1 consiste na vetorização conjunta de programas, dados e endereços em um único tipo de estruturas simbólicas. A inclusão de endereços em estruturas de dados requer endereçamento indireto.¦

agregação:
vetores de instr./dados/ender. simbólicos
interconexões auxiliares:
símbolo como índice de partes de
vetores de instr./dados/ender. simbólicos

3) SS/1(b):
(SS/1 expandido)
¦A expansão de SS/1 permite a manipulação de estruturas encadeadas de símbolos.¦

operadores: unidades de execução de vetores
de programas simbólicos
operandos: memórias de expressões simbólicos
canais simbólicos de E/S *meio*: expressões simbólicas

4) `SS/1(b)` ⇒ `SS/1(c)`:
(Realimentações em `SS/1`)
⊧As realimentações em `SS/1` possibilitam que um programa simbólico chame qualquer programa (inclusive a si próprio) durante seu funcionamento, permite que ele interprete qualquer expressão simbólica (inclusive a própria expressão simbólica que o.consitui como programa) e possibilita a auto-transformação de programas.⊧

recursões:
recursão mútua de programas simbólicos
reflexões:
auto-interpretação e auto-transformação de programas simbólicos
interconexões auxiliares:
memórias de registros de ativação de programas

5) `SS/1(c)`:
(`SS/1` completo)
⊧O primeiro nível `SS` completo corresponde ao nível de sistemas físicos de símbolos de Newell [85] e ao nível da arquitetura de von Neumann. É o nível em que surgem os primeiros sistemas simbólicos funcionamente autônomos.⊧

operadores: sistemas simbólicos autônomos
operandos: memórias de expressões simbólicas
canais simbólicos de E/S *meio*: expressões simbólicas

A.6 Construção do nível SS/2

1) SS/2(a):
(SS/1 reconstruído em SS/2)
⊧O segundo passo na construção do nível de sistemas de símbolos é reconhecer os sistemas do nível SS/1 como máquinas computadoras individualizadas que interagem com quaisquer outras máquinas computadoras individualizadas. SS/1 é um nível em que os sistemas de símbolos ainda funcionam individualmente, interagindo apenas com seus canais de E/S para o ambiente imediato, mas que por serem autônomos já estão no nível SS/2.⊧

operadores:
sistemas físicos de símbolos
operandos:
canais de comunicação externa de
expressões simbólicas
meio:
expressões simbólicas

2) SS/2(a) ⇒ SS/2(b):
(Expansão de SS/2)
⊧A expansão de SS/2 consiste na vetorização de máquinas computadoras e de canais de comunicação simbólica.⊧

agregação:
vetores de máquinas computadoras e
de canais de comunicação simbólica
interconexões auxiliares:
símbolos como índices de partes de
vetores de sistemas físicos simbólicos e
de canais de comunicação simbólica

3) `SS/2(b)`:
(`SS/2` expandido)
⊧A expansão de `SS/2` permite a nomeação das máquinas computadoras e dos canais de comunicação, o que individualiza os sistemas de símbolos, e permite a coordenação das máquinas sem conexão imediata.⊧

operadores: máquinas computadoras individualizadas
operandos: memórias de expressões simbólicos
meio: expressões simbólicas

4) `SS/2(b)` ⇒ `SS/2(c)`:
(Realimentações em `SS/2`)
⊧Os cruzamentos em `SS/2` possibilitam que conjuntos de máquinas computadoras coordenem e regulem suas atividades, constitutindo funções sistêmicas. Os laços permitem fechos de funções sistêmicas. A regulação das atividades requer avaliação de histórias de funcionamentos.⊧

cruzamentos: funções sistêmicas
laços: fechos de funções sistêmicas
interconexões auxiliares: registros históricos de funcionamentos

5) `SS/2(c)`:
(`SS/2` completo)
⊧O nível `SS` completo corresponde ao nível de sistemas de máquinas computadoras dotadas de capacidade de acomodação efetivamente autônoma frente ao ambiente. Nesse nível, operadores e operandos se combinam em um único tipo de componente computacional, os sistemas auto-organizados.⊧

operadores: sistemas autônomos de
funções sistêmicas
operandos: sistemas autônomos de
funções sistêmicas
meio: estados de atividade de funções sistêmicas

Apêndice B

Um quadro formal para o estudo da inteligência de máquina

B.1 Introdução

Em [27] foi apresentado um esboço de quadro formal para o estudo da inteligência de máquina. Este apêndice contém a re-elaboração daquele quadro, em função da evolução posterior do trabalho. Apresenta especialmente os seguintes aspectos:

i) a presença da tripartição *estrutura/comportamento/função*, realizada em [36] e [37];

ii) a consequente elucidação da noção de teleonomia;

iii) a definição de um espaço para a noção de valor (seção 3.4.4), que toma o lugar da noção de *interesse*, no quadro formal, como a noção primitiva encarregada de captar os aspectos energéticos (motivacionais, etc.) do funcionamento;

iv) um refinamento da noção de ambiente.

B.2 Definições

As definições a seguir estão apresentadas numa ordem de refinamento sucessivo.

Definição 1 (Máquina)
Uma máquina é uma estrutura $\mathcal{M} = (M, \mathcal{E})$ onde

i) M é um mecanismo;

ii) $\mathcal{E}$ é um ambiente contendo a máquina $\mathcal{M}$.

Definição 2 (Ambiente)
Um ambiente é uma estrutura $\mathcal{E} = (P, C, F_{\mathcal{E}})$ onde

i) P é um conjunto de máquinas;

ii) C é a estrutura de coordenação das ações das máquinas de $\mathcal{E}$;

iii) $F_{\mathcal{E}}$ é a estrutura funcional de $\mathcal{E}$.

Definição 3 (Mecanismo)
Um mecanismo é uma estrutura $M = (DS, DF)$ onde

i) DS é uma estrutura de desenvolvimento;

ii) DF é um conjunto de fatores de desenvolvimento.

Definição 4 (Estrutura de desenvolvimento)
A estrutura de desenvolvimento de uma máquina $\mathcal{M} = (M, \mathcal{E})$ com $M = (DS, DF)$ e $\mathcal{E} = (P, C, F_{\mathcal{E}})$ é a estrutura $DS = (DS, \sqsubseteq, T)$ onde

i) $DS = S \times B \times F_S \times V_S$ é o conjunto dos possíveis estágios de desenvolvimento de $\mathcal{M}$;

ii) S é o conjunto dos possíveis estágios estruturais de $\mathcal{M}$, parcialmente ordenado por $\sqsubseteq_S$;

iii) B é o conjunto dos possíveis estágios comportamentais de $\mathcal{M}$, parcialmente ordenado por $\sqsubseteq_B$;

iv) $F_S \subseteq F_{\mathcal{E}}$ é o conjunto dos possíveis estágios funcionais sincrônicos de $\mathcal{M}$, parcialmente ordenado por $\sqsubseteq_{F_S}$;

v) V_S é o conjunto dos estágios de valores sincrônicos de $\mathcal{M}$, parcialmente ordenado por $\sqsubseteq_{V_S}$;

vi) $\sqsubseteq \subseteq DS \times DS$ é a ordem parcial dos possíveis estágios de desenvolvimento, definida componente a componente;

vii) T é a estrutura teleonômica de $\mathcal{M}$.

Definição 5 (Estrutura teleonômica)
A estrutura teleonômica T de uma máquina $\mathcal{M} = (M, \mathcal{E})$ com mecanismo $M = (DS, DF)$ é uma estrutura relacional em DS contendo as seguintes relações:

i) $S \overset{\mathcal{E}}{>\!\!-} B$, a relação de **habilitação comportamental** dos $s \in S$ em $\mathcal{E}$;

ii) $S \overset{\mathcal{E}}{\prec\!\!-} B$, a relação de **requisito estrutural** dos $b \in B$ em $\mathcal{E}$;

iii) $S \overset{\mathcal{E}}{\diamond\!\!-}$, a relação de **adequação dinâmica** entre $s \in S$ e $b \in B$ em $\mathcal{E}$;

iv) $(S \overset{\mathcal{E}}{\diamond\!\!-} B) \overset{\mathcal{E}}{\rightarrow} F_S$, a relação de **habilitação funcional** de $(s, b) \in S \times B$ em $\mathcal{E}$;

v) $(S \overset{\mathcal{E}}{\diamond\!\!-} B) \overset{\mathcal{E}}{-\!\!<} F_S$, a relação de **requisito dinâmico** dos $f \in F_S$ em $\mathcal{E}$;

vi) $(S \overset{\mathcal{E}}{\diamond\!\!-} B) \overset{\mathcal{E}}{-\!\!\diamond} F_S$, a relação de **equilíbrio teleonômico** entre $(s, b) \in S \times B$ e $f \in F_S$ em $\mathcal{E}$.

Definição 6 (Fatores de desenvolvimento)
O conjunto dos fatores de desenvolvimento da máquina $\mathcal{M} = (M, \mathcal{E})$ com $M = (DS, DF)$ é o conjunto $DF = O \cup V_D \cup F_D$ onde

i) O é o conjunto dos possíveis estágios de operações de desenvolvimento, parcialmente ordenado por $\sqsubseteq_O$;

ii) V_D é o conjunto dos possíveis estágios de valores diacrônicos de $\mathcal{M}$, parcialmente ordenado por $\sqsubset_{V_D}$;

iii) F_D é o conjunto dos possíveis estágios funcionais diacrônicos de $\mathcal{M}$, parcialmente ordenado por $\sqsubseteq_D$.

Definição 7
A **regra da motivação continuada** para o desenvolvimento é:
para todo $(s, b, f_S, v_S) \in DS$ de $\mathcal{M} = (M, \mathcal{E})$, com $M = (DS, DF)$, vale

$$\left[(s \overset{\mathcal{E}}{\diamond\!\!-} b) \overset{\mathcal{E}}{-\!\!\diamond} f_S\right] \in v_S.$$

Definição 8
A **regra do equilíbrio teleonômico** é:
para todo $d, d' \in DS$ de $\mathcal{M} = (M, \mathcal{E})$, com $M = (DS, DF)$ e $d = (s, b, f_S, v_S)$, vale

$$\left[d \sqsubseteq d' \wedge (s \overset{\mathcal{E}}{\diamond\!\!-} b) \overset{\mathcal{E}}{-\!\!\diamond} f_S\right] \Rightarrow d = d'.$$

Referências Bibliográficas

[1] C. Ades. A teleologia revisitada. *Cadernos de História e Filosofia da Ciência*, 2:31–42, 1981.

[2] R. T. Allen. *The Structure of Value*. Avebury, Aldershot, 1993.

[3] S.-I. Amari. Mathematical foundations of neurocomputing. In C. Lau, (ed.), *Neural Networks - Theoretical Foundations and Analysis*. IEEE Press, New York, 1991.

[4] T. Anderson & P. A. Lee. *Fault Tolerance – Principles and Practice*. Prentice-Hall, Englewood Cliffs, 1981.

[5] M. Arbib. *Theories of Abstract Automata*. Prentice-Hall, Englewood Cliffs, 1969.

[6] M. A. Arbib & E. G. Manes. *Arrows, Structures, and Functors – The Categorical Imperative*. Academic Press, New York, 1975.

[7] J. Backus. Can programming be liberated from the von Neumann style? a functional style and its algebra of programs. *Comm. of the ACM*, 21(8):613–641, 1978.

[8] C. G. Bell & A. Newell. *Computer Structures: Readings and Examples*. McGraw-Hill, New York, 1971.

[9] C. H. Bennett. The thermodynamics of computation – a review. *Intern. Journ. of Theor. Phys.*, 21(12):905–940, 1982.

[10] L. Bertallanffy. *General Systems Theory. Foundations, Development, Applications*. George Brazillier, New York, 1968.

[11] R. Bird. *Programs and Machines - an Introduction to the Theory of Computation*. Wiley, London, 1976.

[12] W. S. Brainerd & L. H. Landweber. *Theory of Computation*. Wiley, New York, 1974.

[13] W. Brauer, (ed.). *Net Theory and Applications*. Springer-Verlag, Berlin, 1980. (Lec. Notes Comp. Sci., 84).

[14] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1–3):139–159, 1991.

[15] A. W. Burks, H. H. Goldstine, & J. v. Neumann. Preliminary discussion of the logical design of an electronic computing instrument. Part I, vol.1, 1946. In A. H. Taube, (ed.), *John von Neumann - Collected Works*, p. 34–79. MacMillan, New York, 1963.

[16] R. Carnap. *Meaning and Necessity - a Study in Semantics and Modal Logic*. Univ. Chicago Press, Chicago, 1956.

[17] A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

[18] A. C. R. Costa. Microprogramação e projeto de sistemas operacionais: pré-projeto de um núcleo de sistemas operacional microprogramado. Dissertação de Mestrado, CPGCC/UFRGS, Porto Alegre, 1980.

[19] A. C. R. Costa. Modelagem do computador de von Neumann através de redes de canais e agências. In *Seminário Integrado de Software e Hardware, 8*, p. 1–13, Florianópolis, jul. 27-31, 1981. SBC.

[20] A. C. R. Costa. Modelos em rede para estruturas factuais percebidas na interação com sistemas dinâmicos. Porto Alegre, 1983. (RI n.10/83).

[21] A. C. R. Costa. A natureza do artificial, 1986. In: A. C. R. Costa *A Fundamentação da Inteligência Artificial.*

[22] A. C. R. Costa. A necessidade da inteligência de máquina - um enfoque sociológico, 1986. In: A. C. R. Costa *Seminário de Epistemologia da Inteligência Artificial, 1986: máquina e inteligência.*

[23] A. C. R. Costa, (ed.). *Seminário de Epistemologia da Inteligência Artificial, 1986: máquina e inteligência*. CPGCC/UFRGS, Porto Alegre, 1986. (RP 67/86).

[24] A. C. R. Costa. *Sobre os Fundamentos da Inteligência Artificial*. Curso JAI, Congresso da SBC, Recife, 1986. (Também em: Porto Alegre, CPGCC/UFRGS, 1986. RP 61/86).

[25] A. C. R. Costa. Bancos de conhecimentos evolutivos, 1988. (Proposta de Projeto de Pesquisa - não publicado).

[26] A. C. R. Costa. Efetividade abstrata e efetividade concreta. In: A. C. R. Costa *A Fundamentação da Inteligência Artificial.*, 1990.

[27] A. C. R. Costa. *Preliminary Thoughts on Agents and Their Development.* CPGCC/UFRGS, Porto Alegre, 1990. (RP 133/90).

[28] A. C. R. Costa, (ed.). *Seminário de Epistemologia da Inteligência Artificial, 1989: IA e Racionalidade*. CPGCC/UFRGS, Porto Alegre, 1990. (RP 122/90).

[29] A. C. R. Costa. The aporia of the transparent machine, 1991. (não publicado).

[30] A. C. R. Costa. *A Fundamentação da Inteligência Artificial*. CPGCC/-UFRGS, Porto Alegre, 1991. (Proposta de tese de doutoramento).

[31] A. C. R. Costa. O papel da análise de fundamentos no desenvolvimento da ciência, 1991. In: A. C. R. Costa *A Fundamentação da Inteligência Artificial*.

[32] A. C. R. Costa. Programmed dynamic systems, the dynamical systems approach to computation theory, and the control theory approach to artificial intelligence, 1991. (Mimeo - não publicado).

[33] A. C. R. Costa. A noção de inteligência de máquina. Notas de Aula. Porto Alegre, CPGCC/UFRGS, 1992.

[34] A. C. R. Costa. Teoria da computação interativa. Notas de Aula. Porto Alegre, CPGCC/UFRGS, 1992.

[35] A. C. R. Costa. Book review of: Drescher, G. *Made-up Minds: a Constructivist Approach to Artificial Intelligence*. MIT Press, 1991. Submetido ao *Journal of Logic and Computation*, 1993.

[36] A. C. R. Costa, J. M. V. Castilho, & D. M. Claudio. Functional roles and functional processes in societies of computing agents. In *SBIA'93 - Simpósio Brasileiro de Inteligência Artificial*, p. 267–276, SBC, Porto Alegre, 1993.

[37] A. C. R. Costa, J. M. V. Castilho, & D. M. Cláudio. Towards a constructive notion of functionality. II/UFRGS, não publicado, 1993.

[38] A. C. R. Costa & H. Coelho. On machine intelligence and the effectiveness of conversations among computing agents, 1991. (CPGCC/UFRGS, não publicado).

[39] A. C. R. Costa & P. G. Greenfield. Knowledge operating systems, 1991. (CPGCC/UFRGS, não publicado).

[40] M. Davis. *Computability and Unsolvability*. McGraw-Hill, New York, 1958.

[41] Y. Demazeau & J.-P. Müller, (eds.). *Decentralized Artificial Intelligence*. Elsevier, Amsterdam, 1990.

[42] E. W. Dijkstra. Hierarchical ordering of sequential processes. In C. A. R. Hoare & R. H. Perrot, (eds.), *Operating Systems Techniques*, p. 72–93. Academic Press, London, 1972.

[43] R. C. Dorf. *Modern Control Systems*. Addison-Wesley, Reading, 1974.

[44] G. Drescher. *Made-up Minds: a Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, 1991.

[45] H. Dreyfus. *What Computers Can't Do - a Critique of Artificial Reason*. Harper and Row, New York, 1972. (Trad. bras.: O que os computadores não podem fazer - uma crítica da razão artificial. A Casa do Livro Eldorado, Rio de Janeiro, 1975).

[46] C. C. Elgot & A. Robinson. Random-access stored-program machines, an approach to programming languages. *Journ. of the ACM*, 11(4):365–399, 1964.

[47] M. H. Escardó. Comunicação pessoal.

[48] M. H. Escardó. *Uma teoria da computação com não terminação significativa: aspectos externos dos computadores*. CPGCC/UFRGS, Porto Alegre, 1991. (Trabalho individual).

[49] M. H. Escardó. Números naturais parciais. Dissertação de Mestrado, CPGCC/UFRGS, Porto Alegre, 1993.

[50] M. H. Escardó. On lazy natural numbers with applications to computability theory and functional programming. *ACM SIGACT News*, February 1993.

[51] PACE Forum. From artificial intelligence to artificial life - background material. The EuroPACE Open Forum - Nov. 7, 1991.

[52] M. Genesereth & N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, 1987.

[53] H.J. Genrich & E. Stankiewicz-Wiechno. A dictionary of some basic notions of net theory. In W. Brauer, (ed.), *Net Theory and Applications*, p. 519–535. Springer-Verlag, Berlin, 1980.

[54] K. Gödel. On undecidable propositions of formal mathematical systems, 1934. In *Kurt Gödel - Obras Completas*, p. 147–182. Alianza, Madrid, 1981. (Trad. espanhola).

[55] H. H. Goldstine & J. v. Neumann. On the principles of large scale computing machines, 1946. In A. H. Taub, (ed.), *John von Neumann – collected works*, p. 1–32. MacMillan, 1963, New York, 1963.

[56] H.H. Goldstine & J. v. Neumann. Planning and coding of problems for an electronic computing instrument - Part I, vol.I, 1947. In A. H. Taub, (ed.), *John von Neumann – Collected Works*, p. 80–151. MacMillan, New York, 1963.

[57] C. A. Gunter. *Semantics of Programming Languages - structures and techniques*. MIT Press, Cambridge, 1992.

[58] C. Hewitt. The challenge of open systems. *Byte*, 10:223–242, 1985.

[59] D. Hilbert. On the infinite, 1925. In J. v. Heijenoort, (ed.), *From Frege to Goedel – a Source Book in Mathematical logic, 1879–1931*. Harvard Univ. Press, Cambridge, 1967.

[60] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[61] C. A. R. Hoare & R. H. Perrott, (eds.). *Operating Systems Techniques*. Academic Press, London, 1972.

[62] J. E. Hopcroft & J. D. Ullman. *Formal Languages and their relation to Automata*. Addison-Wesley, Reading, 1969.

[63] E. Hunt. Computer simulation: Artificial Iintelligence studies and their relevance to Psychology. *Annual Review of Psychology*, 19:135–168, 1968.

[64] Keller. Formal verification of parallel programs. *Comm. of the ACM*, 7:371–384, 1976.

[65] S. C. Kleene. *Introduction to Metamathematics*. D. van Nostrand, New York, 1952.

[66] D. Knuth. von Neumann's First Computer Program. *ACM Computing Surveys*, 2(4):247–260, 1970.

[67] R. Kowalski. *Logic for Problem Solving*. North-Holland, New York, 1979.

[68] T. S. Kuhn. *The Structure of Scientific Revolutions*. Univ. Chicago Press, Chicago, 1962. (Trad. brasileira: A estrutura das revoluções científicas. Perspectiva, São Paulo, 1982).

[69] J. Ladrière. *Les Limitation Internes des Formalismes: études sur la signification du théorème de Gödel et des théorèmes apparentés dans la théorie des fondements mathématiques*. Gauthiers-Villars, Paris, 1957.

[70] P. Latil. *Introduction à la pensée artificielle*. Gallimard, Paris, 1952. (Trad. brasileira: O pensamento artificial. São Paulo, IBRASA, 1968).

[71] C. Lau, (ed.). *Neural Networks - Theoretical Foundations and Analysis*. IEEE Press, New York, 1991.

[72] D. G. Luenberger. *Introduction to Dynamic Systems - Theory, models, and applications*. Wiley, New York, 1979.

[73] P. Maes, (ed.). *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*. MIT Press, Cambridge, 1991.

[74] Z. Manna & A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, New York, 1992.

[75] A. Marques. *Organismo e Sistema em Kant – ensaio sobre o sistema crítico kantiano*. Presença, Lisboa, 1987.

[76] H. Maturana & F. Varela. *Autopoiesis and Cognition - the realization of the living*. D. Reidel, Dordrecht, 1980.

[77] J. McCarthy. Programs with common sense. In M. Minsky, (ed.), *Semantic Information Processing*, p. 403–418. MIT Press, Cambridge, 1968.

[78] P. McCorduck. *Machines Who Think*. Freeman, New York, 1979.

[79] M. Mesarovic & Y. Takahara. *General Systems Theory: Mathematical Foundations*. Academic Press, 1975.

[80] R. Milner. *Communication and concurrency*. Prentice-Hall, Englewood Cliffs, 1989.

[81] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, 1967.

[82] M. Minsky, (ed.). *Semantic Information Processing*. MIT Press, Cambridge, 1968.

[83] J. Monod. *Le hazard et la necessité*. Seuil, 1970. (Trad. bras.: Monod, J. O Acaso e a Necessidade. Vozes, Petrópolis, 1971).

[84] A. Newell. Physical symbol systems. *Cognitive Science*, 4:135–183, 1980.

[85] A. Newell. The knowledge level. *Artificial Intelligence*, 18:87–127, 1982.

[86] A. Newell, J. C. Shaw, & H. A. Simon. Elements of a theory of human problem solving. *Psychological Review*, 65(3):151–166, 1958.

[87] A. Newell & H. Simon. *Human Problem-solving*. Prentice-Hall, Englewood Cliffs, 1972.

[88] A. Newell & H. Simon. Computer science as empirical inquiry: Symbols and search. *Comm. of the ACM*, 19(3):113–126, 1976.

[89] M. Nielsen, G. Plotkin, & G. Winskel. Petri nets, event structures and domains, part 1. *Theoretical Computer Science*, 13:85–108, 1981.

[90] N. Nilsson. *Problem-solving Methods in Artificial Intelligence*. McGraw-Hill, New York, 1971.

[91] N. Nilsson. *Principles of Artificial Intelligence*. Springer-Verlag, Berlin, 1982.

[92] C. A. Petri. Introduction to net theory. In W. Brauer, (ed.), *Net Theory and Applications*, p. 1–19. Berlin, Springer-Verlag, 1980. (Lec. Notes in Comp. Sci., 84).

[93] C. A. Petri. State-transition structures in physics and in computation. *Int. Journ. of Theor. Phys.*, 21(12):979–992, 1982.

[94] J. Piaget. *La Naissance de l'Intelligence chez l'Enfant*. Delachaux et Niestlé, Neuchâtel, 1936. (Trad. bras.: O Nascimento da Inteligência na Criança. Zahar, RJ, 1982).

[95] J. Piaget. *La Psychologie de l'Intelligence*. Colin, Paris, 1947. (Trad. bras.: A Psicologia da Inteligência. Zahar, RJ, 1977).

[96] J. Piaget. *Traité de Logique - Essai de Logique Operatoire*. Collin, Paris, 1949. (2éme ed., Dunod, Paris, 1971. Trad. bras.: Ensaio de lógica operatória. Globo/EDUSP, Porto Alegre, 1976.).

[97] J. Piaget. *Introduction à l'Épistémologie Génétique*. PUF, Paris, 1950. (vol. I: La pensée mathématique).

[98] J. Piaget. *Études Sociologiques*. Droz, Génève, 1965. (Trad. bras.: Estudos sociológicos. Forense, RJ, 1973).

[99] J. Piaget. *Biologie et Connaissance*. Gallimard, Paris, 1967. (Trad. brasileira: Piaget, J. Biologia e Conhecimento. Petrópolis, Vozes, 1973).

[100] J. Piaget. *L'équilibration des Structures Cognitives – Problème central du dévelopment*. P.U.F., Paris, 1975. (Trad. bras.: A equilibração das estruturas cognitivas – problema central do desenvolvimento. Zahar, RJ, 1976).

[101] J. Piaget. *Tratado de Lógica e Conhecimento Científico*, chapter Los métodos de la epistemologia. Paidos, Buenos Aires, 1979. (Orig. francês: Logique et Connaissance Scientifique, Gallimard, Paris,1967).

[102] G. Plotkin. Domains. Technical report, Comp. Sci. Dept., Edinburgh Univ., 1984. (Lecture notes).

[103] E. Post. Finite combinatory processes - formulation 1. *The Journal of Symbolic Logic*, 1(3):103–105, September 1936.

[104] R. J. Queiroz. *Proof Theory and Computer Programming - The Logical Foundations of Computation*. Tese de Doutoramento, Imperial College, Univ. London, London, 1990.

[105] W. Reisig. *Petri nets: an introduction*. Springer-Verlag, Berlin, 1985.

[106] E. Rich. *Artificial Intelligence*. McGraw-Hill, New York, 1983.

[107] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967.

[108] F. Rosenblatt. The perceptron: a probabilistic model for information storage and retrieval in the brain. *Psychological Review*, 65(3):386–407, 1958.

[109] A. Rosenblueth, N. Wiener, & J. Bigelow. Behaviour, purpose and teleology. *Philosophy of Science*, 10:18–24, 1943. (Trad. brasileira: Cadernos de História e Filosofia da Ciência, vol. 2, p. 51–89, 1981. Unicamp, SP).

[110] D. S. Scott. Outline of a mathematical theory of computation. Tech. Monogr. PRG-2, Oxford, Oxford Univ. Comp. Lab., 1970.

[111] D. S. Scott. The lattice of flow diagrams. In E. Engeler, (ed.), *Symposium on semantics of algorithmic languages*, p. 311–366. Berlin, Springer-Verlag, 1971, 1971. (Lec. Notes in Mathematics, 188).

[112] D. S. Scott. Lattice theory, data types and semantics. In R. Rustin, (ed.), *NYU Symposium on Formal Semantics*, p. 64–106, New York, 1972. Prentice-Hall.

[113] J. R. Searle. Minds, brains, and programs. *The Behavioral and Brain Sciences*, 3(3):417–424, 1980.

[114] C. Sernadas, H. Coelho, & G. Gaspar. Communicating knowledge systems – big talk among small systems (part I). *Applied Artificial Intelligence*, 1:233–260, 1987.

[115] C. Sernadas, H. Coelho, & G. Gaspar. Communicating knowledge systems – big talk among small systems (part II). *Applied Artificial Intelligence*, 1:315–335, 1987.

[116] H. Simon. *The Sciences of the Artificial*. MIT Press, 1969. (Trad. port.: As ciências do artificial. Armênio Amado, Coimbra, 1981).

[117] A. Sloman. Prolegomena to a theory of communication and affect. Technical report, School of Comp. Sci., Univ. Birmingham, 1991. (CSRP-91-5).

[118] A. Sloman. The emperor's real mind. Technical report, School of Comp. Sci., Univ. Birmingham, 1992. (CSRP-92-8).

[119] R. Sommerhalder & S. C. van Westrhenen. *The Theory of Computability - programs, machines, effectiveness and feasibility*. Addison-Wesley, Wokingham, 1988.

[120] E. Stein. *Racionalidade e existência – uma introdução à filosofia*. L&PM, Porto Alegre, 1988.

[121] J. E. Stoy. *Denotational Semantics: the Scott-Strachey Approach to Programming Language Theory*. MIT Press, Cambridge, 1977.

[122] A. Tanenbaun. *Modern Operating Systems*. Prentice-Hall, Englewood Cliffs, 1992.

[123] C. Thiel (ed.). Special Issue on VM/370. *IBM Systems Journal*, 18(1), 1979.

[124] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc.*, 42:230–265, 1936.

[125] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950. (Trad. brasileira: Computação, Maquinaria e Inteligência. In: Epstein, I. Cibernética e Comunicação. São Paulo, Cultrix, 1973. p. 49-82).

[126] R. Turner. *Constructive Foundations for Functional Languages*. McGraw-Hill, London, 1991.

[127] F. J. Varela. *Autonomie et Connaissance - essai sur le vivant*. Seuil, Paris, 1989.

[128] F. J. Varela. Organism - a meshwork of selfless selves. In *From Artificial Intelligence to Artificial Life - background material*, p. 12–57. PACE Forum, 1991.

[129] R. Vieira & A. C. R. Costa. The acceptance relation and the specification of communicating agents. In *ICICIS 93 - Int. Conf. on Intelligent and Cooperative Information Systems*. IEEE, 1993. (Erasmus Univ., Rotterdam, The Netherlands, May 11-14, 1993).

[130] J. von Neumann. *The Computer and the Brain*. Yale Univ. Press, New Haven, 1958. (2nd. ed., 1967).

[131] H. Wang. A variant to Turing's theory of computing machines. *Journ. of the ACM*, 4(1):63–92, 1957.

[132] K. Weihrauch. *Computability*. Springer-Verlag, Berlin, 1987.

[133] L. L. Whyte, A. G. Wilson, & D. Wilson, (eds.). *Hierarchical Structures*, Douglas Adv. Res. Lab., Huntington Beach, Cal., 1969. Elsevier. (Trad. esp.: Las estructuras jerárquicas. Alianza, Madrid, 1973).

[134] N. Wiener. *Cybernetics - or, control and communication in the animal and the machine*. MIT Press, Cambridge, 1948. (2nd ed.,1961).

[135] T. Winograd. Extended inference modes in reasoning by computer systems. *Artificial Intelligence*, 13:5–26, 1980. (Special issue on Non-monotonic Logic).

[136] T. Winograd & F. Flores. *Understanding Computers and Cognition - a new foundation for design.* Ablex, Norwood, New Jersey, 1987.

[137] P. H. Winston. *Artificial Intelligence.* Addison-Wesley, Reading, 1984.

[138] H. Zemanek. Some philosophical aspects of information processing. In *The Skyline of Information Processing*, p. 93–140. North-Holland, Amsterdam, 1972.