Alan Turing

Legados para a Computação e para a Humanidade

Dante Augusto Couto Barone,
André Noronha Furtado de Mendonça,
Museu da UFRGS
(organizadores)

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor - Carlos Alexandre Netto

Vice-reitor - Rui Vicente Oppermann

Pró-reitor de pesquisa - João Edgar Schmidt

Pró-reitora de extensão - Sandra de Deus

Pró-reitor de planejamento e administração - Ário Zimmermann

Superintendente de infraestrutura - Alberto Tamagna

Diretora do Museu da UFRGS - Claudia Porcellis Aristimunha

Alan Turing: legados para a computação e para a humanidade (2012 : Porto Alegre, RS)

Alan Turing: legados para a computação e para a humanidade / catálogo da exposição organizado por Dante Augusto Couto Barone, André Noronha Furtado de Mendonça e Museu da UFRGS. - Porto Alegre: UFRGS, 2013.

164 p.: 20 il., 134 fots (Série Catálogos das Exposições, 2).
ISBN 978-85-64701-01-4

Exposição realizada pelo Museu da UFRGS, de outubro de 2012 a março de 2013, com curadoria geral de Dante Augusto Couto Barone e subcuradoria de André Noronha Furtado de Mendonça.
Apresentação de Claudia Porcellis Aristimunha

1. Turing, Alan, 1912-1954. 2. Computação - UFRGS - Exposição. I. Barone, Dante Augusto Couto. II. Mendonça, André Noronha Furtado de. III. Aristimunha, Claudia Porcellis. IV. Museu da UFRGS.

CDU 681.32(063)

Catalogação-na-publicação: Biblioteca Central/UFRGS

Computabilidade & Máquina de Turing

A Máquina de Turing

É um dispositivo imaginário que formou a estrutura para fundamentar a ciência da computação moderna.

David Hilbert

Kurt Goedel

O início do século XX foi fértil em transformações sobre nossa visão de mundo. A física newtoniana era desafiada pela Relatividade de Einstein e pelo princípio da incerteza de Heisenberg. Na matemática isso não poderia ter sido diferente.

Até as primeiras décadas do século XX o mundo ainda acreditava que todo problema matemático poderia ter uma solução. Tal crença ganhou força na comunidade científica com um amplo projeto conhecido como Programa de Hilbert, que consistia em 23 problemas complexos elaborados por Hilbert, e que, segundo ele, todos teriam uma solução. Desde então, as maiores mentes matemáticas do mundo vem tentando solucionar tais problemas, sendo que, alguns destes ainda hoje não puderam ser resolvidos.

Em um congresso em Munster, em 1925, David Hilbert afirmou:

> "Se existe um problema, ache a solução! Você poderá encontrá-la apenas pensando, pois não há ignorabimus* em matemática".

** ignorabimus é uma expressão latina que significa "ignorar ou ignoremos". Foi empregada entre fins do século XIX e início do século XX, refletindo certa tendência acadêmica da época em aceitar a validade de teoremas ainda que não totalmente comprovados, submetendo-se às limitações do conhecimento humano, sem almejar a solução plena de determinado problema.*

Para Hilbert um matemático deveria:

1- Axiomatizar todo o corpo de conhecimento matemático (inclusive

com objetivo de provar que todo problema matemático fosse solúvel).
2- Provar, por meios estritamente finitários, que a axiomática pretendida fosse consistente.

Em 1931, outro matemático, Göedel, propôs (e comprovou isso) que nem todos os problemas teriam solução. Isso ficou conhecido como teorema da incompletude de Göedel. Esse teorema faria ruir a romântica era positivista na matemática.

Com isso, Gödel passou a representar um contraponto importante com relação a Hilbert, propondo que, de fato, existem problemas matemáticos insolúveis.

Em seu teorema da incompletude de 1931, Kurt Gödel timidamente anunciou que:

> "Pode-se, de fato, exibir sentenças verdadeiras mas que são indemonstráveis no sistema formal da matemática".

O único dos presentes que imediatamente compreendeu a revolução que se iniciava foi John von Neumann.

A partir desta sentença, os matemáticos acabavam de ser expulsos do paraíso...

Os teoremas da Incompletude de Gödel:

> 1º Teorema de Incompletude: Em todo sistema formal consistente S, com um mínimo de Aritmética, é possível formalizar uma sentença U tal que U possa ser interpretada intuitivamente como a afirmação de que ela própria é indemonstrável em S..
>
> 2º Teorema da Incompletude: A prova da consistência para sistemas formais (envolvendo um pouco de

Antares. Foto de Eder Iván, 2012.

John von Neumann

Aritmética, nas condições que Hilbert queria) não pode ser formalizada dentro do próprio sistema.

Qual seria a relação destes dois matemáticos com Alan Turing?

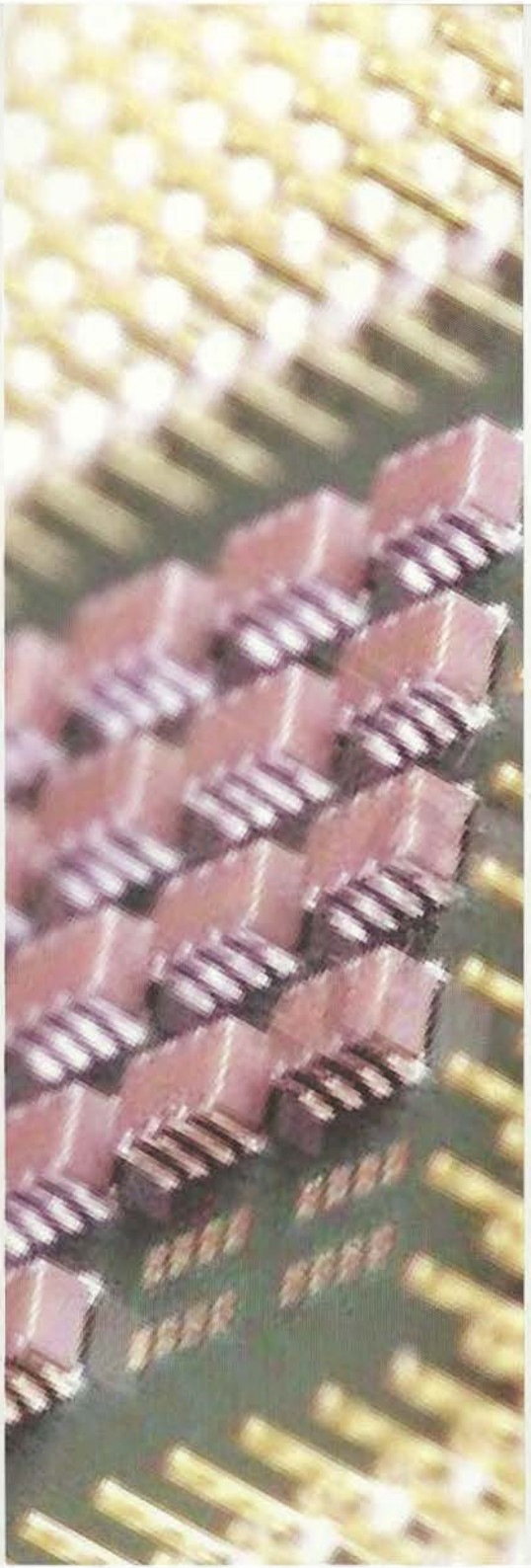
Em 1936, Alan Turing revelou ao mundo seu brilho, pela primeira vez, em um artigo chamado Números Computáveis. Neste artigo, Turing demonstrou que, ao contrário de Gödel, de fato todo problema poderia ter uma solução, desde que este problema pudesse ser descrito com números computáveis. Para demonstrar sua teoria, Alan Turing descreveu pela primeira vez na história, a lógica computacional como a conhecemos hoje. Com isso, ele inaugurou

a ciência da computação, ou seja, ele possibilita o surgimento do computador moderno.

Cada calculadora, cada celular, cada tablet, cada processador embarcado em fornos de micro-ondas, aparelhos de TV, relógios, na estação espacial internacional, ou onde quer que imaginemos que possa existir um computador, tudo isso tem origem nesse artigo publicado por Turing em 1936.

A invenção da Máquina de Turing possibilitou o desenvolvimento de inúmeras outras áreas como a computação afetiva, a inteligência artificial e a robótica, entre tantas outras. Todas estas, são áreas dentro da ciência da computação. Associar a invenção do computador com a criação de todas essas áreas do conhecimento poderia ser algo natural, mas o surpreendente nisso tudo, é que Turing não apenas abriu o caminho para elas ao inventar o computador moderno, como inaugurou, ele mesmo boa parte dessas áreas. Turing se aprofundou em neurociência e vislumbrou formas de como tornar os computadores mais inteligentes.

Vamos tentar entender um pouco mais sobre o contexto matemático que influenciou Turing a criar o computador. Para isso precisamos antes compreender dois conceitos importantes: a consistência e a completude.



Stock Photos ©

$\mathfrak{sim}$			$f'(\mathfrak{sim}_1, \mathfrak{sim}_1, z)$
$\mathfrak{sim}_1$			con $(\mathfrak{sim}_2,)$
$\mathfrak{sim}_2$	A		$\mathfrak{sim}_3$
	not A	R, Pu, R, R, R	$\mathfrak{sim}_2$
$\mathfrak{sim}_3$	not A	L, Py	$e(\mathfrak{mf}, z)$
	A	L, Py, R, R, R	$\mathfrak{sim}_3$

$\mathfrak{sim}$. The machine marks out the instructions. That part of the instructions which refers to operations to be carried out is marked with u, and the final m-configuration with y. The letters z are erased.

$\mathfrak{mf}$			$g(\mathfrak{mf}, :)$
$\mathfrak{mf}_1$	not A	R, R	$\mathfrak{mf}_1$
	A	L, L, L, L	$\mathfrak{mf}_2$
$\mathfrak{mf}_2$	C	R, Px, L, L, L	$\mathfrak{mf}_2$
	:		$\mathfrak{mf}_4$
	D	R, Px, L, L, L	$\mathfrak{mf}_3$
$\mathfrak{mf}_3$	not :	R, Pv, L, L, L	$\mathfrak{mf}_3$
	:		$\mathfrak{mf}_4$
$\mathfrak{mf}_4$			con $\left(t\left(1(\mathfrak{mf}_5)\right), \right)$
$\mathfrak{mf}_5$	Any	R, Pw, R	$\mathfrak{mf}_5$
	None	$P:$	$\mathfrak{sh}$

$\mathfrak{mf}$. The last complete configuration is marked out into four sections. The configuration is left unmarked. The symbol directly preceding it is marked with x. The remainder of the complete configuration is divided into two parts, of which the first is marked with v and the last with w. A colon is printed after the whole. $\rightarrow \mathfrak{sh}$.

$\mathfrak{sh}$			$f(\mathfrak{sh}_1, \mathfrak{inst}, u)$
$\mathfrak{sh}_1$		L, L, L	$\mathfrak{sh}_2$
$\mathfrak{sh}_2$	D	R, R, R, R	$\mathfrak{sh}_2$
	not D		$\mathfrak{inst}$
$\mathfrak{sh}_3$	C	R, R	$\mathfrak{sh}_4$
	not C		$\mathfrak{inst}$
$\mathfrak{sh}_4$	C	R, R	$\mathfrak{sh}_5$
	not C		$pe_2(\mathfrak{inst}, 0, :)$
$\mathfrak{sh}_5$	C		$\mathfrak{inst}$
	not C		$pe_2(\mathfrak{inst}, 1, :)$

$\mathfrak{sh}$. The instructions (marked u) are examined. If it is found that they involve "Print 0" or "Print 1", then 0: or 1: is printed at the end.

Parte do artigo publicado por Alan Turing em 1936 - Sobre Números Computáveis

Consistência: Uma teoria axiomática * é dita consistente quando nela não é derivada uma contradição, ou seja, não são derivados uma proposição e a sua negação. Por exemplo, a teoria das álgebras de Boole pode ser considerada consistente, pois possui um modelo finito.Quando as regras utilizadas correspondem a lógica clássica, se a teoria for inconsistente, a partir de uma contradição pode ser derivado qualquer enunciado, trivializando o sistema.

Completude: Uma teoria axiomática é dita completa se para cada proposição P da teoria (fórmula sem variáveis livres), ou bem pode ser deduzida P ou bem pode ser deduzida a negação de P. Como exemplos de teorias matemáticas completas podemos citar a teoria dos corpos algebricamente fechados de característica fixa e a teoria das álgebras de Boole sem átomos.

**Na matemática, um axioma é uma hipótese inicial da qual outros enunciados são logicamente derivados.*

Alèm de David Hilbert e Kurt Göedel, outro contemporâneo de Turing foi John von Neumann. Neumann foi um matemático húngaro de etnia judaica, naturalizado norte-americano. Durante muitos anos ele foi considerado o inventor do computador moderno. Na realidade, ele trabalhou junto com Alan Turing no desenvolvimento do computador. Entre suas principais contribuições:

- *Análise Funcicnal*
- *Teoria Ergódica*
- *Estatística*
- *Teoria dos Números*
- *Formalização da Computação*

Descobertas paralelas - como já mencionamos anteriormente, no campo da Física, encontrava-se em pleno andamento o desenvolvimento da teoria quântica e quatro anos antes de Göedel, (1927) Heisenberg divulgara seu "principio da incerteza", colocando um limite físico na experimentação microscópica direta quando demonstrou que ou se determinava a posição de uma partícula, ou se determinava o seu momento. Estes foram duros golpes nas hipóteses determinísticas da ciência de então.

O problema da parada

Trata-se em outras palavras decidir se um programa é um algoritmo, ou seja, um programa possui ou não um término. Se o número de dados é finito, o problema deve ser verificável para todos os dados. Caso contrário, é impossível provar que ele para, para qualquer dado do conjunto disponível.

O problema da parada é um belo exemplo do que um computador não pode realizar, pois de certo modo ele deve decidir sobre o seu próprio funcionamento. A mente humana pode realizar tal tarefa sem se autodestruir, o computador ainda não.

O problema da parada consiste inicialmente em colocar em uma fila infinita blocos de dados que chamaremos de M (M1, M2, M3,...). Todos os computadores que já foram ou que serão inventados funcionam

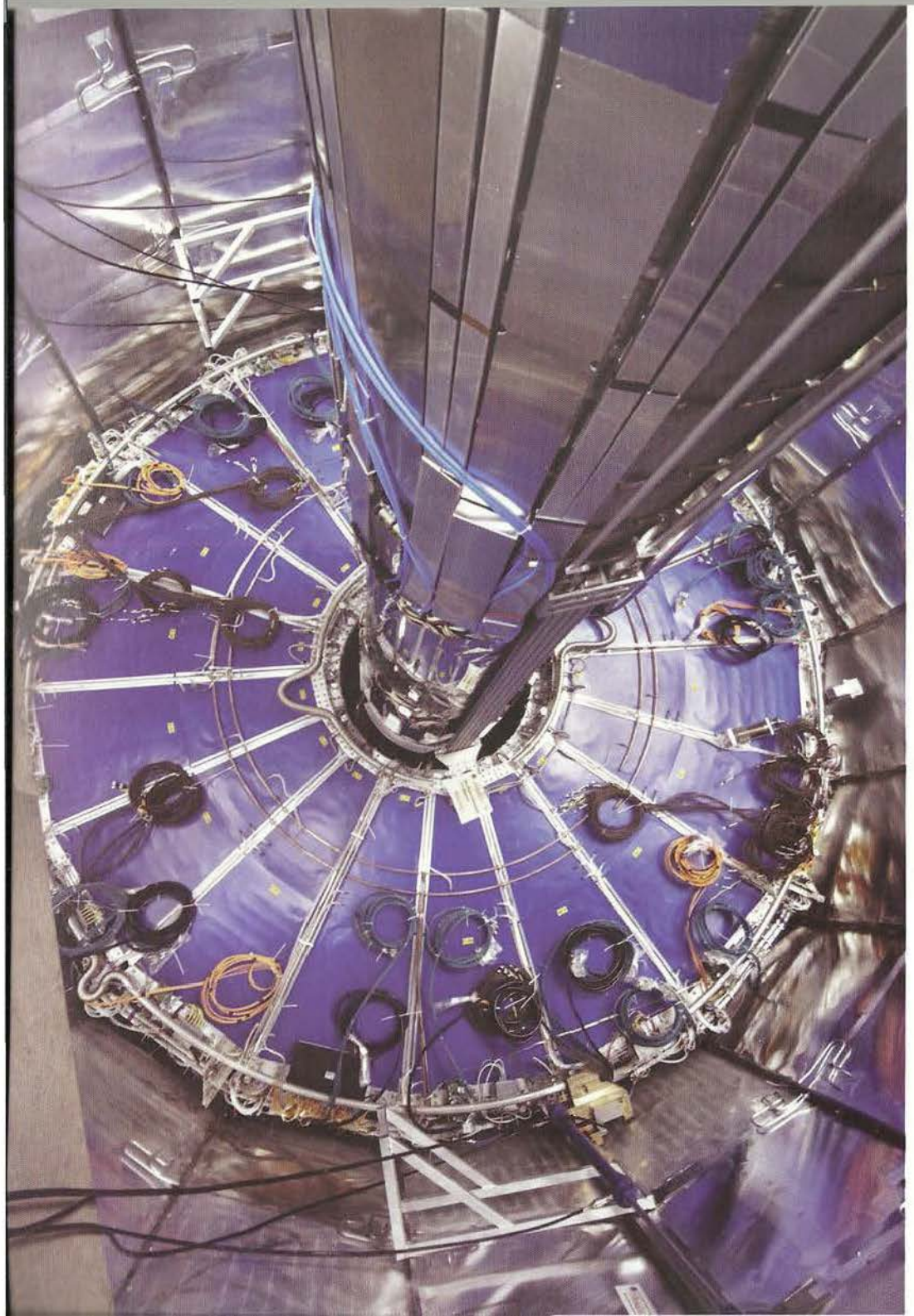


Foto - CERN (European Laboratory for Particle Physics)

assim.. Estamos supondo que todos estes computadores são máquinas de Turing (isto é usando o que se conhece como a tese de Church). Quando alimentado com dados, cada um destes computadores pode parar depois de processá-lo ou então entrar em um loop infinito, não parando nunca.

Gostaríamos de inventar um novo computador que respondesse a seguinte pergunta: Dada uma máquina Mi e um número n, a máquina Mi pára quando alimentada com o número n? Mais precisamente, existe um computador P que, analisando a máquina Mi e o dado n consegue responder:

P (Mi , n) será igual a 11 (dois uns seguidos), se a máquina Mi pa-

rar quando alimentada inicialmente com o dado n.

P (Mi , n) será igual a 1 (um único 1) se a máquina Mi não parar quando alimentada inicialmente com o dado n.

Podemos mostrar que tal computador P não pode existir. A demonstração deste fato será dada na forma intuitiva explorando-se a auto-referência.

Suponhamos, por absurdo, que o computador P existisse. Todo número n pode ser escrito como uma sequência consecutiva de n 1's.

É fácil construir uma máquina M que dobra o número de 1's: o primeiro destes n 1's será usado para localizarmos a máquina pelo seu número na lista de todas as máquinas que fizemos acima e o segundo para servir de dado de entrada para essa mesma máquina.

Podemos acoplar três máquinas: M, P e uma terceira máquina que pára quando alimentado por um único 1 e nunca pára quando alimentada por dois números 1's consecutivos. Esta nova máquina com os três acoplamentos tem o seguinte aspecto:

Vamos chamar esta nova máquina, formada pelo acoplamento das três anteriores de Mk. Esta é uma nova máquina com uma propriedade muito estranha: ela pára se e somente se a máquina Mn não pára quando alimentada com o dado n (isto é, com n 1's consecutivos). Mas, como n é um número natural qualquer, podemos fazer este "vampiro" digital se alimentar do próprio sangue, tomando-se n = k. O que acontece então? Se Mk pára então ela mesma não pára e se ela não pára, então ela pára. Essa é a contradição. Assim, a máquina P não pode existir e o problema da parada é insolúvel.

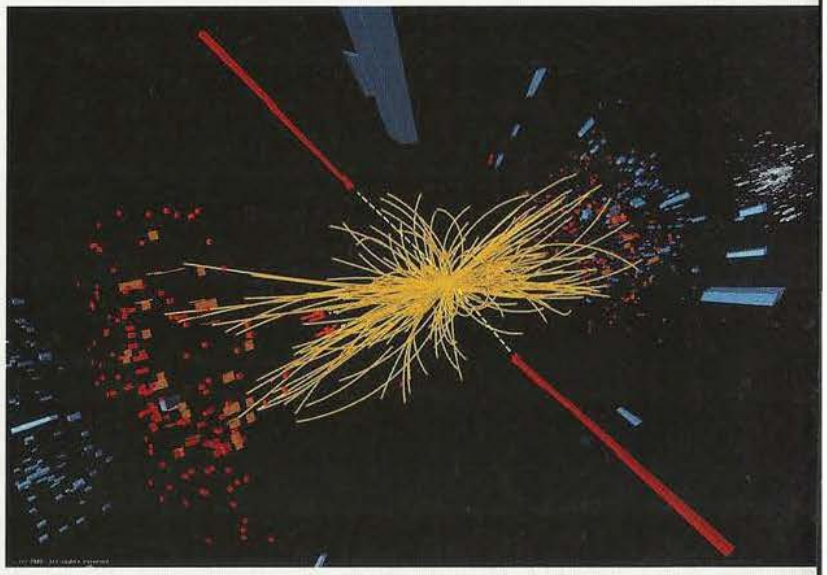
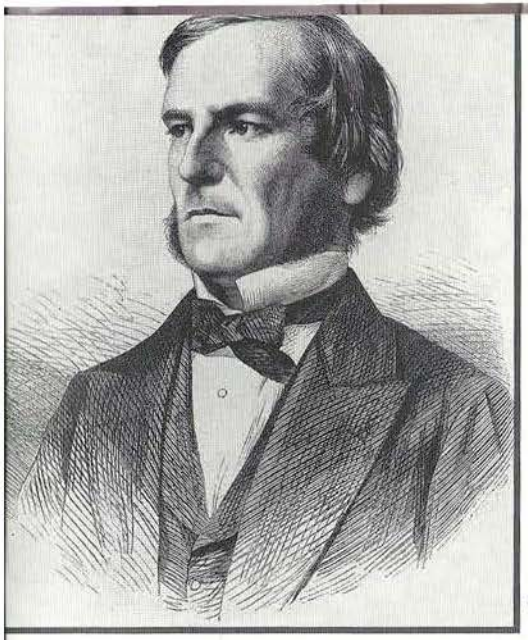


Foto - CERN (European Laboratory for Particle Physics) - sumulação de colisão de partículas.



George Boole

Matemático e filósofo britâncio, George Boole criou a Álgebra Booleana, baseada na teoria de Verdadeiro e Falso de Leibniz, que foi fundamental para o desenvolvimento do trabalho de Turing na computação moderna.
A Álgebra Booleana trabalha com o conceito de Consistência e de Completude, ou seja, não pode ter contradição nesta lógica, ou é 0 ou é 1, nunca 1 ½ e para cada hipótese existem dois resultados: ou a sua confirmação ou a sua negação.
Para a computação isto se traduz na redução de falhas do seu computador.

Conseqüências

Uma conseqüência da indecidibilidade do problema da parada é que não pode existir um algoritmo genérico que decida se um dado enunciado sobre os números naturais é verdadeiro ou falso. A razão para isso é que a proposição que afirma que certo algoritmo vai parar a cada certa entrada, pode ser convertido em um enunciado equivalente sobre os números naturais. Se nós tivéssemos um algoritmo que pudesse resolver todo enunciado sobre os números naturais, ele certamente poderia resolver tal enunciado, mas isso determinaria se o problema original pára, o que é impossível, já que o problema da parada é indecidível.

A Máquina de Turing

É um dispositivo imaginário que consolidou a estrutura lógica para fundamentar a ciência da computação moderna.
Criada por Alan Turing, que demonstrou que a ação de computar operações de leitura, escrita e exclusão de símbolos binários poderia ser satisfeita por uma máquina que contivesse uma fita de comprimento ilimitado, com campos de entrada de dados de tamanho definido sobre ela e um dispositivo com um número finito de estados, que realizava as operações na fita.

Em 1936 foi formalizado o termo algoritmo: um conjunto finito de instruções simples e precisas, que são descritas com um número finito de símbolos.

> "Qualquer processo aceito por nós homens como um algoritmo é precisamente o que uma máquina de Turing pode fazer" (Alonzo Church, matemático).

Números Computáveis

A teoria foi publicada em 1936, no artigo "On Computable Numbers, with an Application on the Entscheidungsproblem", em resposta ao tratamento do problema da decisão, formulado por Hilbert.

Apesar da máquina de Turing não ter sido implementada fisicamente, na totalidade pelo seu autor, o processo computacional foi matematicamente demonstrado e provado neste artigo.

Turing explicitou um dispositivo lógico que ele chamou de "automatic machine" (ou "a-machine"), capaz de ler, escrever e apagar símbolos binários em uma fita de comprimento ilimitado e dividida por quadrados de igual tamanho. Um cabeçote de leitura/gravação se moveria em qualquer direção ao longo da fita, um quadrado por vez, e uma unidade de controle poderia interpretar uma lista de instruções simples, movendo-se para a direita ou esquerda. A regra executada determina o que se convencionou chamar de estado da máquina.

O conceito de máquina de Turing é semelhante ao de uma fórmula ou equação.

Assim, há uma infinidade de possíveis máquinas de Turing, cada uma correspondendo a um método definido ou algoritmo. Turing propôs que cada algoritmo, formalizado como um conjunto finito de instruções bem definidas, pudesse ser interpretado e executado por um processo mecânico.

Formalmente a máquina de Turing pode ser definida como uma máquina que contém:

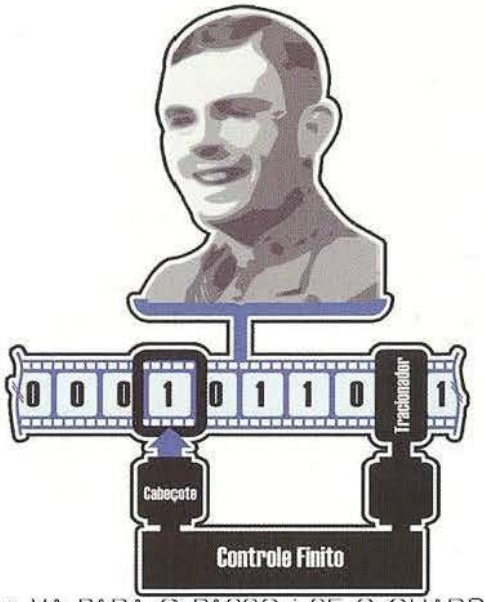
- Um conjunto finito de estados Q com um estado inicial distinto,
- Um conjunto finito de símbolos Σ.

A interpretação e execução dos algoritmos são realizadas por estados e uma função de transição determina o novo conteúdo da fita. Desde modo, por restrição imposta ao algoritmo, pode-se alterar o conteúdo de apenas um quadrado por vez ou movimentar a cabeça, no máximo uma célula em qualquer direção. É permitida também a utilização de qualquer conjunto finito de símbolos para o alfabeto Σ, mesmo que a definição original tenha insistido em Σ = {0,1}. Esta mudança não tem impacto sobre a definição do conjunto de funções computáveis pela máquina.

O que torna uma máquina de Turing capaz de executar uma tarefa é a tabela de regras de transição que compõem o programa da máquina e um determinado estado inicial. O conjunto de instruções conhecidas e processadas pelo módulo de controle finito, como podemos observar na figura a seguir.

e é relacionado abaixo:

- IMPRIMA 0 NO QUADRADO QUE PASSA PELA CABEÇA
- IMPRIMA 1 NO QUADRADO QUE PASSA PELA CABEÇA
- VÁ UM QUADRADO PARA A ESQUERDA
- VÁ UM QUADRADO PARA A DIREITA
- VÁ PARA O PASSO i SE O QUADRADO QUE PASSA PELA CABEÇA CONTÉM 0



- VA PARA O PASSO j SE O QUADRADO QUE PASSA PELA CABEÇA CONTÉM 1
- PARE

Por exemplo: uma computação pode ser representada por três estados, nomeados s0, s1, s2 e com algumas instruções formalizadas no Algoritmo 1:

1. { s0 , 1, s0 , »}
2. { s0 , 0, s1 , 1}

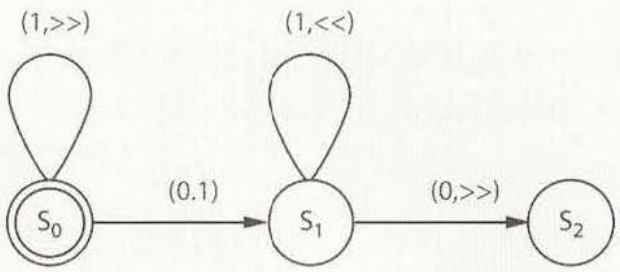
3. { s1 , 1, s1 , «}
4. { s1 , 0, s2 , »}

As duas primeiras instruções (linhas 1 e 2) descrevem o que acontecerá no estado s0. Há duas possibilidades: na primeira, a máquina faz a leitura de um dígito '1', movimentará a cabeça para a direita e permanecerá no estado s0.

Na segunda, se for lido um dígito '0' a máquina deixará o estado s0, entrará no estado s1 e escreverá o dígito '1' nessa transição. As instruções descritas nas linhas 3 e 4 mostram o que acontecerá no estado s1, ou seja, se for lido o dígito '1', a máquina movimentará a cabeça para a esquerda e permanecerá no estado s1. Se for lido o dígito '0', a cabeça será movimentada para a direita e a máquina passará para o estado s2.

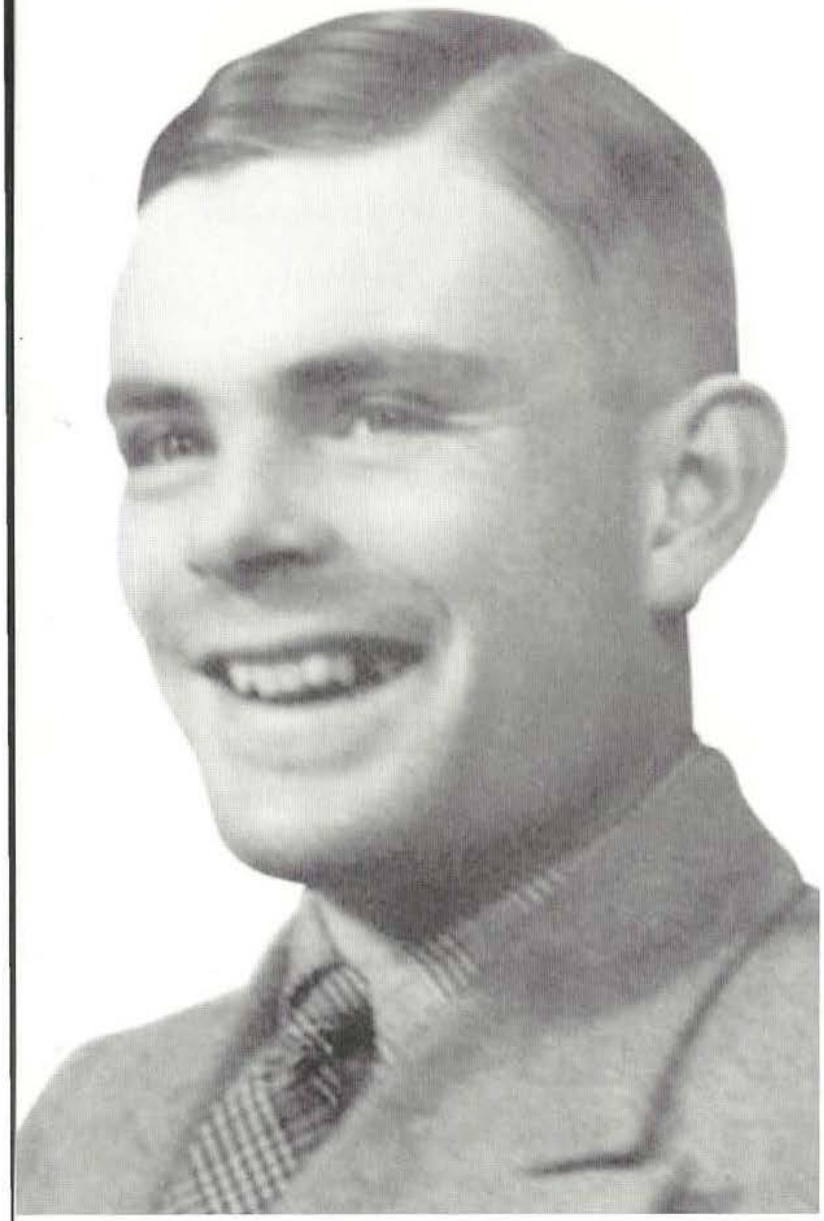
Como não há instruções definidas pelo algoritmo no estado s2, a máquina pára a sua execução (condição de parada) ao atingir este estado.

Quando estamos interessados em examinar o comportamento de uma máquina de Turing, é eficiente representarmos a máquina usando um diagrama de estados. A Figura 2 representada o conjunto de instruções neste formato visual.



Na figura anterior, os estados são representados por círculos, com um círculo duplo identificando o estado inicial. Uma transição é representada por uma aresta ou arco proveniente de um estado para outro ou para o mesmo estado. As arestas são rotuladas por pares (símbolo, ação), constituído primeiro pelo símbolo que deverá ser lido e depois, pela ação que deverá ser executada com a transição. Os símbolos pertencem ao alfabeto Σ e a ação será o símbolo a ser escrito, ou ainda « ou », indicando um movimento para a esquerda ou direita.

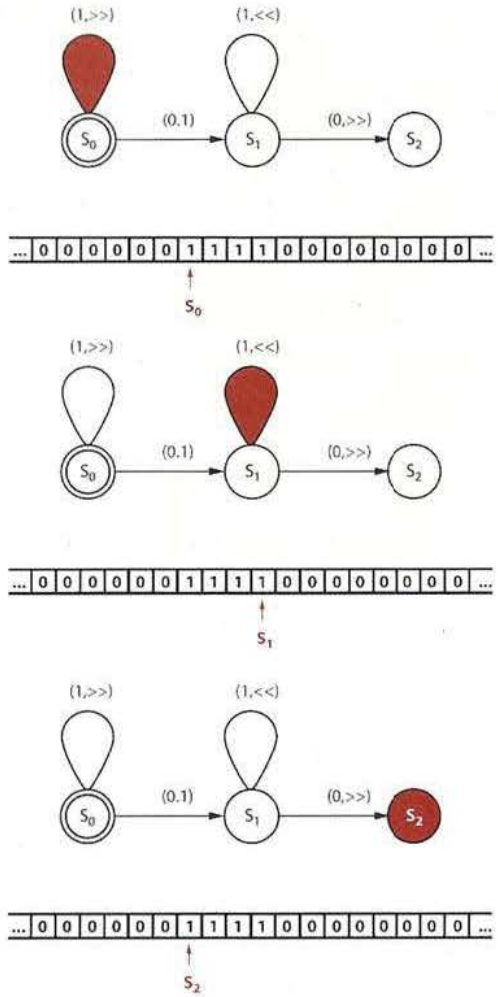
Em nosso exemplo, temos uma máquina que calcula o sucessor de n, como uma execução simples para



fins meramente demonstrativos.

Assim, o estado inicial da fita representa o número n e após executar a sequência de passos, ele irá parar na configuração padrão que representa o número n +1.

Uma possível execução é demonstrada na figura seguinte:



Na atualidade é fácil relacionar um programa de computador com uma máquina de Turing e a tarefa mecânica de interpretação e execução obedecendo ao algoritmo. Assim, a máquina de Turing Universal incorpora o princípio essencial do computador: uma máquina simples que poderá executar qualquer tarefa bem definida, desde que especificada como um programa apropriado. Turing provou que para qualquer sistema formal existe uma máquina

de Turing que pode ser programada para imitá-lo. Era este sistema formal genérico, com a habilidade de imitar qualquer outro sistema formal, o que Turing procurava essencialmente. Tais sistemas chamam-se Máquinas de Turing Universais. O lógico matemático Alonzo Church chegou a definir: "Qualquer processo aceito por nós homens como um algoritmo é precisamente o que uma máquina de Turing pode fazer".

Em abril de 1936, Turing mostrou seus resultados para John Von Neumann em Princeton, quando os computadores, no sentido moderno, ainda não existiam. Turing criou os conceitos e a fundamentação matemática, que nove anos depois seria a tecnologia utilizada para materializar os primeiros computadores eletrônicos, com grande participação de Neumann, ou seja, a transformação da lógica de suas ideias abstratas em engenharia real. Durante este período de tempo, Turing retornou à Inglaterra e a ideia viveu apenas em sua mente. A correspondência entre instruções lógicas, a ação da mente humana e uma máquina, que poderia ser fisicamente construída, foi a contribuição definitiva de Alan Mathison Turing.

Foto: Luis Lamb

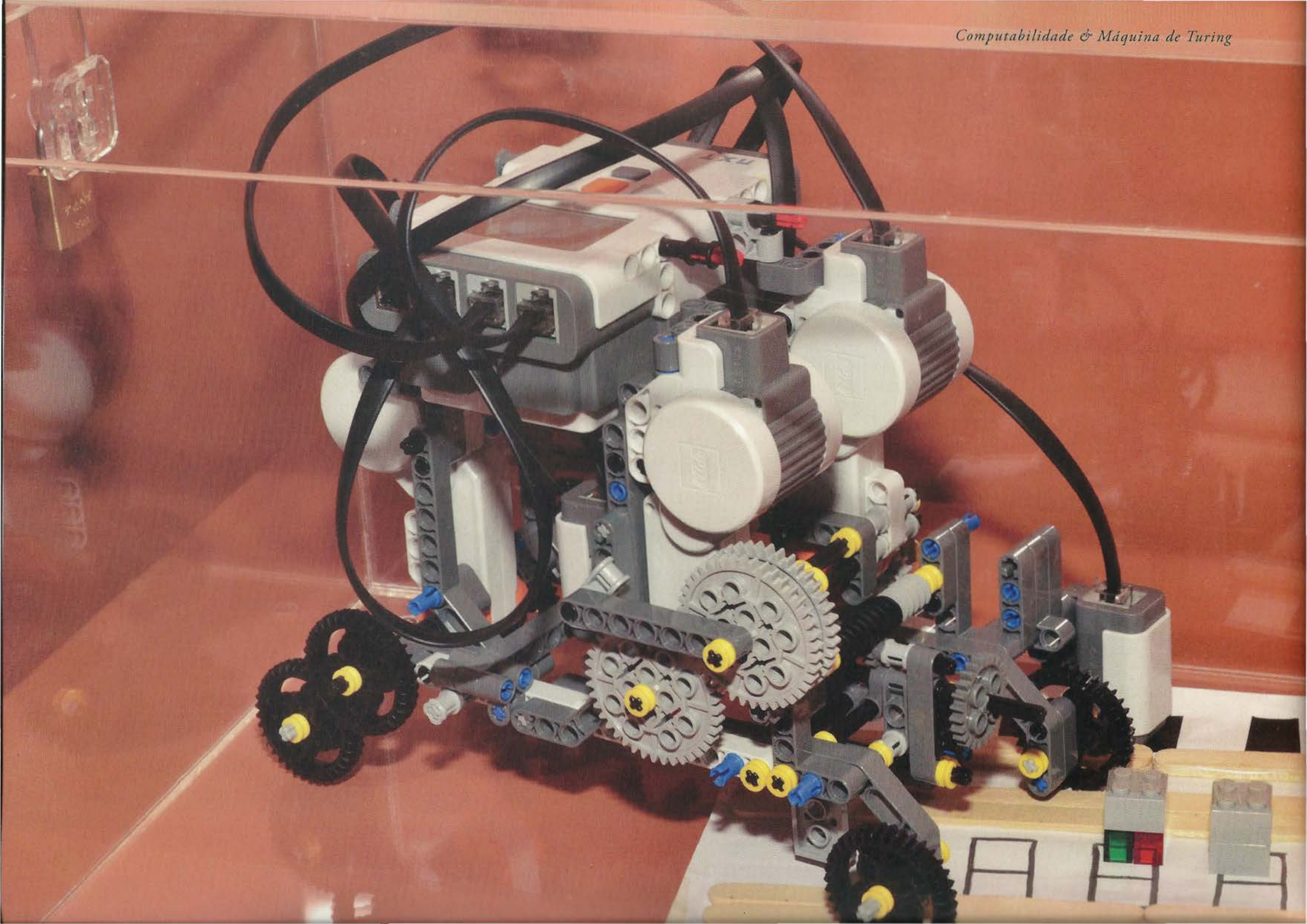


Ilustração de Jim Wicked, representando a máquina de Turing.

Máquina de Turing feita com um kit de peças de Lego, desenvolvida por bolsistas do PET da Computação da UFRGS. Foto: André Furtado.

Os Autores

Dante Augusto Couto Barone - É coordenador do Grupo de Pesquisa de Robótica Inteligente e Visão Artificial do Instituto de Informática da UFRGS. Coordenador do projeto Alan Turing Brasil. Atualmente é Professor Associado da Universidade Federal do Rio Grande do Sul, Consultor do Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco, Consultor da Fundação de Amparo à Pesquisa do Estado de São Paulo, Sócio da Sociedade Brasileira de Computação, foi membro da Direção e Administração da Sociedade Brasileira para o Progresso da Ciência - SP, Avaliador Institucional do Instituto Nacional de Estudos e Pesquisas Educionais Anisio Teixeira, Revisor de periódico da Pattern Recognition Letters.

André Noronha Furtado de Mendonça - É professor do curso de Design do Instituto Federal Sul-Rio-Grandense de Pelotas e doutorando em Informática na Educação pelo PGIE/UFRGS. É Mestre em Design e Tecnologia pelo PGDesign/UFRGS.
Desenvolve trabalhos em design gráfico, projeto editorial, expografia, design de produto, design para web, design de interface, jogos digitais, modelagem em 3D, animação, videografia, ilustração, pintura digital e fotografia.

Ivan Jorge Boesing - Graduado em Fisica Licenciatura Plena pela UNISINOS (1996) e mestre em Engenharia Mecânica pela UFRGS (2001). Atua no grupo de pesquisa Robótica Inteligente e Visão Artificial do Instituto de Informática da UFRGS.

Marcelo Walter - É conselheiro da Sociedade Brasileira de Computação (SBC). De 2009 a 2011 foi vice-presidente da SBC. De 2007 a 2009 foi o Diretor de Eventos e Comissões Especiais.

Mathias Seibel Luce - Professor do Departamento de História da UFRGS. É coordenador do HEDLA/UFRGS - Núcleo de História Econômica da Dependência Latino-Americana; Doutor em História pelo PPGHIST/UFRGS.

Cairo Antonio Oliveira Rocha - Graduado em Ciência da Computação na UFPI, atualmente é mestrando do programa de pós-graduação em computação da UFRGS.

Camila Mozzini - Formada em Jornalismo na UFRGS e Mestre em Psicologia Social e Institucional na UFRGS, atualmente estuda o estatuto da conexão digital no contemporâneo a partir de uma perspectiva foucaultiana.

Clarissa Felkl Prevedello - Designer formada pela UFSM, Especialista em Design de Moda pela UEL, mestre em Design e doutoranda em Informática na Educação pelo PGIE/UFRGS. Designer do IFSUL-Pelotas e professora pesquisadora e tutora da Universidade Aberta do Brasil.

Elias de Almeida Ramos - Possui graduação em Licenciatura Plena em Matemática pela Universidade Tiradentes. Tem experiência na área de Ciência da Computação, com ênfase em Processamento Gráfico, Engenharia de Software, Linguagem de Programação, Redes Neurais e Sistemas Dinâmicos.

Gélio José da Silva Júnior - Graduado em Ciência da Computação pela Universidade do Extremo Sul Catarinense, Brasil e professor da Escola de Educação Básica Bulcão Viana.

Heinrich Felix Marmitt - Graduando em Engenharia de Computação (UFRGS). Atua principalmente em injeção de falhas e redes de computadores.

Jean Michel Winter - Engenheiro Eletricista pela UFRGS (2010). Atuou em projetos de pesquisa na área de Ambientes Inteligentes e Redes de Sensores sem Fio. Mestrando do Programa de Pós-Graduação de Engenharia Elétrica (UFRGS) - ênfase em Sistemas de Automação.

Joel Luis Carbonera - Bacharel em Ciência da Computação pela UCS. É mestrando em Inteligência Artificial na UFRGS.

Luiz Otávio Vilas Bôas Oliveira - Graduação em Ciência da Computação pela Universidade Federal de Itajubá, Brasil (2010).

Marcelo de Gomensoro Malheiros - Engenheiro de Computação pela Unicamp (1996) e Mestre em Engenharia Elétrica pela Unicamp (1999). É professor do Centro Universitário UNIVATES, atuando em: modelagem, síntese e processamento de imagens, 3D interativo e tecnologias de código aberto.

Marco Aurélio Schünke - Graduado em Ciência da Computação pela Universidade de Santa Cruz do Sul - UNISC. Atualmente aluno regular de Mestrado em Ciência da Computação da Universidade Federal do Rio Grande do Sul - UFRGS.

Renan de Queiroz Maffei - É formado em matemática.

Rogério Xavier de Azambuja - Graduado e Mestre em Ciência da Computação. Estudante de Pós-Graduação no PPGC-UFRGS.

Sérgio Montazzolli Silva - Graduado em Ciências da Computação pela UEL, aluno da Pós Graduação em Ciências da Computação na UFRGS.

Vinícius Costa de Souza - Mestre em Computação Aplicada (2005) pela UNISINOS e Bacharel em Informática - Hab. Análise de Sistemas (2002) pela mesma instituição.

ISBN 978-85-64701-01-4