Uma abordagem modelo-teórica da computabilidade de Turing clássica

Anderson de Araújo

Tese apresentada ao
Instituto de Filosofia e Ciências Humanas da
Universidade Estadual de Campinas,
para a obtenção do grau de
Doutor em Filosofia (Área de Lógica)

Orientador: **Prof. Dr. Walter Alexandre Carnielli**

Durante a elaboração deste trabalho,
o autor recebeu apoio financeiro da FAPESP.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IFCH - UNICAMP
Bibliotecária: Cecília Maria Jorge Nicolau CRB nº 3387

de Araújo, Anderson
Ar15a Uma abordagem modelo-teórica da computabilidade de Turing clássica / Anderson de Araújo. - - Campinas, SP: [s. n.], 2011.

Orientador: Walter Alexandre Carnielli.
Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Filosofia e Ciências Humanas.

1. Lógica simbólica e matemática. 2. Turing, Máquinas de. 3. Aritmética. I. Carnielli, Walter A. (Walter Alexandre). II. Universidade Estadual de Campinas. Instituto de Filosofia e Ciências Humanas. III.Título.

Título em inglês: A model-theoretical approach to classical Turing computability

Palavras chaves em inglês (keywords): Symbolic logic and mathematical
Turing machines
Arithmetic

Área de Concentração: Filosofia
Titulação: Doutor em Filosofia
Banca examinadora: Walter Alexandre Carnielli, Antônio Mariano Nogueira Coelho, Marcelo Finger, Hugo Luiz Mariano, Arnaldo Vieira Moura.
Data da defesa: 18-03-2011
Programa de Pós-Graduação: Filosofia

UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE FILOSOFIA E CIÊNCIAS HUMANAS

A Comissão Julgadora dos trabalhos de Defesa de Tese de Doutorado, em sessão pública realizada em 18 de março de 2011, considerou o candidato ANDERSON DE ARAÚJO aprovado.

Este exemplar corresponde à redação final da Tese defendida e aprovada pela Comissão Julgadora.

Prof. Dr. Walter Alexandre Carnielli _____

Prof. Dr. Antônio Mariano Nogueira Coelho _____

Prof. Dr. Marcelo Finger _____

Prof. Dr. Hugo Luiz Mariano _____

Prof. Dr. Arnaldo Vieira de Moura _____

Dedico aos meus pais, Marlene e Vilmar, por tudo.

L'essentiel est invisible pour les yeux.
Antoine de Saint-Exupéry

AGRADECIMENTOS

Em *Les Travailleurs de la Mer*, Victor Hugo afirmou que o ser humano tem uma triplíce *ananke*: a religião, a sociedade, a natureza. Como instância dessa máxima, no âmbito acadêmico três são nossas necessidades: a família, os amigos, os mestres. A elas só nos cabe agradeçer.

Agradeço à minha família natural. Minha mãe, Marlene, meu pai, Vilmar, minha mana, Mariel, e meu mano pequeno, Matheus, por possibilitarem tudo em minha vida.

Agradeço à minha família advinda. Dona Lúcia, seu Vadney, minha guriazinha, Viviane, e seus irmãos, Patrícia e Erick, por fazerem parte de minha vida.

Agradeço aos meus amigos. André Luis da Silva, pelo exemplo de garra em tudo; José Guimarães, pelo companheirismo no universo da computação; Samir Gorski, pelos diálogos filosóficos; Juan Carlos Agudelo, pelas discussões sobre computabilidade; Igor Oliveira, pelas discussões sobre complexidade; Leandro Suguitani, Rodrigo Freire, Teófilo Reis e Rafael Testa, pelos estudos de lógica.

Agradeço aos meus mestres. Frank Thomas Sautter, por introduzir-me na lógica; Dirk Greimann, por ensinar-me a ser um pesquisador; Ronai Pires da Rocha e Robson Ramos dos Reis, pelos exemplos de sistematicidade conceitual; Walter Carnielli, por ensinar-me o caminho do mestre; Marcelo Coniglio e Ítala D'Ottaviano, por me apontarem as sutilezas das demonstrações matemáticas; Amílcar Sernadas, pelas sugestões cruciais a totalidade desta tese; Jeff Paris e Costas Dimitracopoulos, por me ajudarem com os modelos aritméticos; Hugo Mariano e Marcelo Finger, pelas indicações pontuais e centrais acerca de uma versão preliminar desta tese.

Em especial, agradeço à Viviane, por ser mais do que uma ananke, por ser minha fortaleza.

RESUMO

Esta tese propõe uma nova abordagem da computabilidade de Turing clássica, denominada *abordagem modelo-teórica*. De acordo com essa abordagem, estruturas e teorias são associadas às máquinas de Turing a fim de investigar as características de suas computações. Uma abordagem modelo-teórica da computabilidade de Turing através da lógica de primeira ordem é desenvolvida, e resultados de correspondência, correção, representação e completude entre máquinas, estruturas e teorias de Turing são demonstrados. Nessa direção, os resultados obtidos a respeito de propriedades tais como estabilidade, absoluticidade, universalidade e logicidade enfatizam as potencialidades da computabilidade modelo-teórica de primeira ordem. Demonstra-se que a lógica subjacente às teorias de Turing é uma lógica minimal intuicionista, sendo capaz, inclusive, de internalizar um operador de negação clássico. As técnicas formuladas nesta tese permitem, sobretudo, investigar a computabilidade de Turing em modelos não-padrão da aritmética. Nesse contexto, uma nova perspectiva acerca do fenômeno de Tennenbaum e uma avaliação crítica da abordagem de Dershowitz e Gurevich da tese de Church-Turing são apresentadas. Como conseqüência, postula-se um princípio de internalidade aritmética na computabilidade, segundo o qual o próprio conceito de computação é relativo ao modelo aritmético em que as máquinas de Turing operam. Assim, a tese unifica as caracterizações modelo-aritméticas do problema $\boldsymbol{P}$ versus $\boldsymbol{NP}$ existentes na literatura, revelando, por fim, uma *barreira modelo-aritmética* para a possibilidade de solução desse problema central em complexidade computacional no que diz respeito a certos métodos. Em sua totalidade, a tese sustenta que características cruciais do conceito de computação podem ser vislumbradas a partir da dualidade entre finitude e infinitude presente na distinção entre números naturais padrão e não-padrão.

ABSTRACT

This PhD thesis proposes a new approach to classical Turing computability, called a *model-theoretic approach*. In that approach, structures and theories are associated to Turing machines in order to study the characteristics of their computations. A model-theoretic approach to Turing computability through first-order logic is developed, and first results about correspondence, soundness, representation and completeness among Turing machines, structures and theories are proved. In this line, the results about properties as stability, absoluteness, universality and logicality emphasize the importance of the model-theoretic standpoint. It is shown that the underlying logic of Turing theories is a minimal intuicionistic logic, being able to internalize a classical negation operator. The techniques obtained in the present dissertation permit us to examine the Turing computability over nonstandard models of arithmetic as well. In this context, a new perspective about Tennenbaum's phenomenon and a critical evaluation of Dershowitz and Gurevich's account on Church-Turing's thesis are given. As a consequence, an *arithmetic internality principle* is postulated, according to which the concept of computation itself is relative to the arithmetic model that Turing machines operate. In this way, the dissertation unifies the existing model-arithmetic characterizations of the $\boldsymbol{P}$ versus $\boldsymbol{NP}$ problem, leading, as a by-product, to a *model-arithmetic barrier* to the solvability of that central problem in computational complexity with respect to certain techniques. As a whole, the dissertation sustains that crucial characteristics of the concept of computation may be understood from the duality between finiteness and infiniteness inherent within the distinction between standard and nonstandard natural numbers.

SUMÁRIO

1. INTRODUÇÃO

Nothing can come of nothing.
William Shakespeare

A presente tese versa sobre os fundamentos da computação. O estudo desses fundamentos constitui o objeto da disciplina chamada *teoria da computabilidade*. Conforme J. R. Shoenfield [Shöenfield(1993), p.1], uma *computação* é um processo que associa elementos de um conjunto de *entradas* a elementos de um conjunto de *saídas* através de um *algoritmo*. *Intuitivamente*, um algoritmo é uma seqüência fixa e finita de regras mecânicas. O objetivo principal da teoria da computabilidade é determinar *matematicamente* quais objetos podem ser especificados através de computações bem como investigar suas propriedades e extensões.

De acordo com R. I. Soare [Soare(1999)], os estudos em teoria da computabilidade subdividem-se basicamente em três grandes áreas. A primeira é a *computabilidade clássica*, a qual estuda as funções computáveis e não-computáveis sobre os números naturais. A segunda é a *computabilidade ordinal*, onde se investiga a computabilidade sobre ordinais transfinitos. A terceira é a *computabilidade de ordem superior*, que se ocupa com as computações na hierarquia dos tipos[1].

[1] Referências atuais sobre os principais problemas em teoria da computabilidade podem ser encontradas no *Handbook of Computability Theory* [Griffor(1999)]. Um apanhado geral sobre a computabilidade clássica pode ser encontrado em [Enderton(1977)]. Quanto à computabilidade ordinal, uma boa indicação é [Soare(1977)]. No que respeita à computabilidade de ordem superior, as principais técnicas podem ser encontradas em [Kechris and Moschovakis(1977)].

Esta tese restringir-se-á à computabilidade clássica, sem pressupor os números naturais como previamente dados. Mais especificamente, ela apresentará uma abordagem do conceito de *máquina de Turing* que permitirá investigar a influência da distinção entre objetos *finitários* e *infinitários* no universo das funções computáveis[2].

1.1 Temática

As investigações fundacionais sobre o conceito de computação remontam explicitamente à palestra de D. Hilbert no Congresso Internacional dos Matemáticos, proferida em agosto de 1900 em Paris. Nessa palestra, Hilbert apresentou uma lista de problemas matemáticos para o século XX, publicada posteriormente em [Hilbert(1901)] na forma de 23 problemas. O segundo problema da lista hilbertiana sugeria a possibilidade de demonstração finitária da consistência da aritmética. O décimo problema perguntava sobre a existência de um algoritmo para decidir se toda equação diofantina tem solução. Obviamente, para solucionar esses problemas, era necessário, antes, saber o que é uma *demonstração finitária* e o que é um *algoritmo*. Em alguns trabalhos [Hilbert(1900), Hilbert(1967)], Hilbert procurou esclarecer esses conceitos, e ambos envolviam de alguma forma o conceito de computação, mas não era totalmente claro o que Hilbert tinha em mente.

Em 1931, K. Gödel publicou o artigo [Gödel(1931)] no qual propôs definir parte do conceito de computabilidade em termos das funções recursivas primitivas. Ao fazer isso, Gödel conseguiu demonstrar seus dois famosos teoremas da incompletude, provendo uma resposta negativa ao segundo problema de Hilbert. Em 1934, Gödel [Gödel(1986a)] apresentou uma aborda-

[2] Nesta tese a distinção entre *objetos finitários* e *objetos infinitários* é simplesmente a distinção entre *conjuntos finitos* e *conjuntos infinitos*. De forma alguma, ao fazermos essa identificação, tencionamos dizer que a distinção original é imprecisa ou irrelevante. Isso apenas indica que assumiremos a teoria dos conjuntos ZFC, e, desse modo, todas as distinções efetuadas nesta tese remontam-se à ZFC tal como apresentada, por exemplo, em [Jech(2002)].

gem das funções computáveis em geral, baseada em trabalhos de J. Herbrand não-publicados bem como no seu artigo [Herbrand(1931)]. Esse trabalho de Gödel foi aprimorado por S. K. Kleene [Kleene(1936)] e, desse modo, surgiu a *teoria das funções recursivas*, uma das abordagens clássicas das operações computáveis.

Em 1933, A. Church apresentou sua abordagem das operações computáveis no artigo [Church(1933)], chamada de *lambda cálculo*, o qual também fora desenvolvida por Kleene [Kleene(1935)]. A abordagem de Church consistia na definição de um sistema formal com regras de formação de termos e substituição de termos que representam funções de forma abstrata. Com a formulação do lambda cálculo, Church [Church(1936)] conseguiu demonstrar que a verificação da validade de uma fórmula de primeira ordem não pode ser efetuada de forma algorítmica, problema este chamado *Entscheidungsproblem*. Esse resultado foi um passo importante na direção da solução negativa ao décimo problema de Hilbert, apresentada por Y. Matijasevic [Matijasevic(1970)].

Também em 1936, A. M. Turing [Turing(1936)] chegou a mesma conclusão que Church com respeito ao *Entscheidungsproblem*, mas com uma abordagem independente e diferente. Turing proveu uma análise das operações computáveis enquanto operações efetuáveis por um ser humano ideal que mecanicamente segue regras. A definição formal desse dispositivo de computação passou a ser chamada de *máquinas de Turing*. De acordo com Kleene [Kleene(1994)], para a maioria dos contemporâneos de Turing, o conceito de máquina de Turing apresentava os princípios fundamentais da computabilidade. Essa perspectiva foi corroborada pelo próprio Turing [Turing(1937)] em sua demonstração de que as máquinas de Turing, apesar de sua simplicidade, são capazes de computar todas as funções recursivas e as funções definíveis no lambda cálculo.

Historicamente, a *computabilidade de Turing*, enquanto estudo das funções computáveis por máquinas de Turing, consolidou-se como a abor-

dagem mais elementar da computabilidade. Para Soare [Soare(1996)] e Kleene [Kleene(1994)], há mais do que uma razão histórica para a escolha da computabilidade de Turing como a abordagem fundamental da computação, existem duas razões teóricas para isso: (1) as máquinas de Turing definem toda a classe das funções computáveis que conhecemos, e (2) a análise de Turing constitui-se no modelo de computação mais elementar que dispomos. Por isso, esta tese tematizará um conjunto de problemas associados à computabilidade de Turing clássica, e quando nos referirmos à *computabilidade* estamos identificando-a com *computabilidade de Turing* - a menos que explicitemos outro modelo de computação.

1.2 Problemática

Atualmente, um dos desdobramentos importantes da computabilidade via máquinas de Turing é a complexidade computacional. Na *teoria da complexidade computacional* estuda-se a complexidade das computações de funções *totais* computáveis por máquinas de Turing no que respeita ao *tempo* de execução e ao *espaço* utilizado em suas computações [Arora and Barak(2009)]. Por diversas razões, o principal problema em aberto é saber se a classe das linguagens computáveis em tempo polinomial por máquinas de Turing deterministas ($\boldsymbol{P}$) e não-deterministas ($\boldsymbol{NP}$) é igual ou diferente; tal problema é chamado *problema* $\boldsymbol{P}$ *versus* $\boldsymbol{NP}$[3].

De um ponto de vista estrito, a complexidade computacional constitui-se em uma teoria desenvolvida independentemente da teoria da computabilidade. No entanto, teóricos como P. G. Odifreddi [Odifreddi(1992b)] e S. B. Cooper [Cooper(2004)] consideram que as soluções de problemas de complexidade computacional que dependam de considerações sobre a computabilidade devem ser considerados como parte da teoria da computabilidade em sentido lato. Um caso paradigmático descoberto recentemente é a vinculação

[3] Cf. [Goldreich(2008)] para detalhes sobre os problemas de complexidade computacional.

entre a complexidade das computações das máquinas de Turing e os modelos não-padrão da aritmética.

Os modelos não-padrão da aritmética foram descobertos por S. Skolem [Skolem(1922), Skolem.(1929), Skolem.(1934)] em formalizações da teoria dos números. A existência de tais modelos é um fenômeno lógico abrangente associado às propriedades das linguagens de primeira ordem. Devido às características infinitárias dos números não-padrão, sabemos que os modelos não-padrão enumeráveis da aritmética tem um segmento inicial isomorfo aos números naturais padrão seguido por segmentos cuja ordem é isomorfa à ordem nos números inteiros, os quais estão, por sua vez, densamente ordenados. Costumamos denotar esse tipo de ordem por $\omega + (\omega^* + \omega) \cdot \eta$, onde ω é a ordem dos números naturais padrão, ω^* é a ordem inversa de ω, e η é a ordem dos números racionais[4].

Em 1982, J. Paris e C. Dimitracopoulos [Paris and Dimitracopoulos(1982)] descobriram que a hierarquia das fórmulas aritméticas limitadas $\Delta_0^{\mathbb{N}}$ está vinculada à definibilidade uniforme da teoria completa de um modelo não-padrão. Consequentemente, eles demonstraram que $P \neq NP$ se, e somente se, existirem dois modelos não-padrão enumeráveis da aritmética $\mathcal{M}_1$ e $\mathcal{M}_2$ que, apesar de serem polinomialmente indistinguíveis por máquinas de Turing, não são elementarmente equivalentes. Na esteira de Paris e Dimitracopoulos, em 1990, A. Máté [Máté(1990)] especificou quais as características dos modelos não-padrão da aritmética que implicariam na diferença entre a classe NP e sua complementar $coNP$. Trata-se de algo bastante conhecido em teoria da complexidade computacional que se $NP \neq coNP$ então $P \neq NP$. Em outras palavras, a solução de um problema modelo-teórico da aritmética implicaria na solução do principal problema em complexidade computacional. Em 1995, C. Sureson [Sureson(1995)] mostrou que a solução $NP \neq coNP$ é de fato equivalente às especificações propostas por Máté.

[4] Cf. [Kaye(1991)] para mais informações sobre os modelos não-padrão da aritmética.

Nesses trabalhos citados, todos consideram, em algum ponto de suas demonstrações, computações de máquinas de Turing sobre números não-padrão. No entanto, ninguém apresentou uma análise da computabilidade sobre tais números. A estratégia adotada sempre foi usar fórmulas aritméticas que definem computações de máquinas de Turing sobre números naturais não-padrão, ao invés de considerar as computações de máquinas de Turing. Essa estratégia foi suficiente para os propósitos mencionados, mas algumas dúvidas ficam em aberto.

Do ponto de vista da aritmética, os números não-padrão podem ser representados como seqüências binárias finitas, mas, de um ponto de vista metateórico conjuntista, nós sabemos que essas seqüências não são suficientes para representar tais números, em virtude de existir uma quantidade não-enumerável de modelos não-padrão enumeráveis da aritmética[5]. Ademais, do ponto de vista da aritmética, um número natural não-padrão é um objeto finitário, mas de um ponto de vista metateórico conjuntista, podemos demonstrar que eles são objetos infinitários. Assim, cabe a pergunta:

P1: Quais seriam as características da computabilidade de Turing sobre os números naturais não-padrão?

Como dissemos acima, a descoberta da relação entre a complexidade das computações de máquinas de Turing e os modelos não-padrão da aritmética é surpreendente, porque em teoria da complexidade computacional estamos interessados apenas em funções totais, ou seja, as computações consideradas são todas finitas. Isso significa que os problemas de complexidade computacional são problemas sobre objetos finitários. No entanto, os trabalhos mencionados mostram que ao permitirmos a computabilidade sobre os números naturais não-padrão, que são elementos infinitários de um modelo de uma aritmética, podemos estabelecer vínculos entre a computabilidade e a complexidade computacional. Portanto, a pergunta fundamental é esta:

[5] Cf. [Kaye(1991), Cap. 1], para detalhes.

P2: Como é possível que problemas de complexidade computacional (que tratam de objetos finitários) estejam relacionados com problemas de modelos aritméticos (que tratam de objetos infinitários)?

Nesta tese, apresentaremos uma abordagem da computabilidade que possibilitará analisar esses problemas.

1.3 Proposta

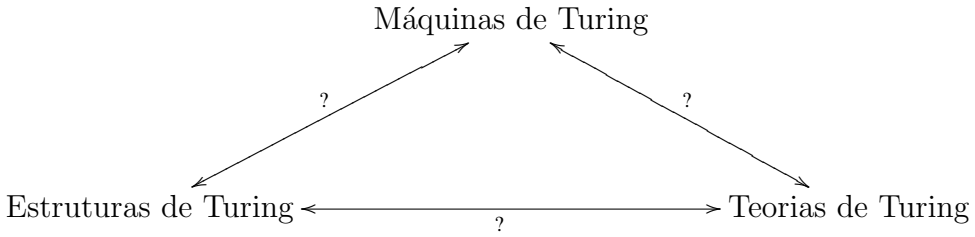
Tendo em vista que não há um tratamento adequado dos problemas apresentados na problematização, várias direções de pesquisa são possíveis. No universo dessas possibilidades, a proposta desta tese é definir uma abordagem geral da computabilidade, denominada *computabilidade modelo-teórica*, que permitirá, em particular, investigar a influência da distinção entre números naturais padrão e não-padrão no conceito de computação. A computabilidade modelo-teórica está definida em termos dos seguintes passos metodológicos:

Primeiro passo: Dada uma classe qualquer de máquinas de Turing $\boldsymbol{Maq}$, associamos uma classe de $\boldsymbol{L}$-estruturas $\boldsymbol{Mod}$, chamadas *estruturas de Turing*, e uma classe de $\boldsymbol{L}$-teorias $\boldsymbol{Teo}$, chamadas *teorias de Turing*, à $\boldsymbol{Maq}$, onde $\boldsymbol{L}$ é uma linguagem lógica, por exemplo, uma linguagem de primeira ordem, capaz de expressar propriedades de $\boldsymbol{Maq}$.

Segundo passo: Investigamos a relação entre máquinas, estruturas e teorias de Turing, i.e., estudamos a relação entre $\boldsymbol{Maq}$, $\boldsymbol{Mod}$ e $\boldsymbol{Teo}$.

A idéia de uma computabilidade modelo-teórica está inspirada na abordagem modelo-téorica das classes de complexidade desenvolvida na área denominada *complexidade descritiva*. A teoria descritiva da complexidade analisa os problemas de complexidade computacional definíveis em sistemas lógicos. Dada uma classe de complexidade, a questão em complexidade descritiva é

saber se existe um sistema lógico que define essa classe[6]. Em contrapartida, na abordagem modelo-teórica que propomos, o estudo das máquinas de Turing através da formulação de sistemas lógicos é efetuado para os fins de análise da computabilidade propriamente dita. Em termos diagramáticos, podemos conceber o tipo de abordagem que estamos propondo do seguinte modo:



Assim, dada uma máquina de Turing $\boldsymbol{M}$ com determinadas características, a questão em *computabilidade modelo-teórica* é, primeiramente, saber se existe uma maneira de associarmos estruturas e teorias à $\boldsymbol{M}$ e, caso exista, também saber qual a relação entre os três conceitos: máquinas de Turing, estruturas de Turing e teorias de Turing. Isso não exclui a possibilidade de efetuar caracterizações de problemas acerca da complexidade computacional; tal estudo pode ser feito *a posteriori*.

Na revisão de literatura, apresentaremos um apanhado geral dos trabalhos que podem ser considerados como antecessores da computabilidade modelo-teórica. Cabe destacar, todavia, que tanto quanto seja do conhecimento do autor desta tese, a idéia de um estudo modelo-teórico da computabilidade tal como formulado aqui é original. A computabilidade modelo-teórica foi concebida pelo autor desta tese a partir de sua intuição inicial de que seria importante investigar a computabilidade em modelos da aritmética, fossem eles padrão ou não. Posteriormente, o autor concebeu que seria apropriado

[6] Cf. [Immerman(1999)] e [Ebbinghaus and Flum(1999), p.119-164] para mais detalhes e referências sobre a complexidade descritiva.

considerar esse problema em um contexto geral. Daí surgiu a idéia de uma computabilidade modelo-teórica.

1.3.1 Objetivos

Em princípio, qualquer modelo de computação baseado em máquinas de Turing pode ser abordado em termos modelo-teóricos e, por isso, o escopo da computabilidade modelo-teórica é amplo[7]. Na presente tese nos restringiremos a uma abordagem modelo-teórica das máquinas de Turing que possibilite um tratamento dos problemas mencionados na problemática acerca da relação entre complexidade computacional e modelos não-padrão da aritmética.

Objetivo geral da tese: Desenvolver uma abordagem modelo-teórica das máquinas de Turing que possibilite analisar a computabilidade sobre os números naturais padrão ou não.

As metas específicas elencadas abaixo contemplam conjuntamente esse objetivo geral.

Primeiro objetivo específico: Explicitar quais são as características da computabilidade de Turing que podem ser analisadas com lógica de primeira ordem.

Segundo objetivo específico: Examinar as computações de máquinas de Turing sobre números naturais padrão e não-padrão.

Terceiro objetivo específico: Identificar a razão pela qual problemas de complexidade computacional podem ser tratados através da construção de modelos aritméticos.

[7] Como está indicado na conclusão desta tese, a idéia de uma computabilidade modelo-teórica possibilita o tratamento de uma série de problemas em computabilidade.

O primeiro objetivo específico está vinculado ao projeto geral de uma computabilidade modelo-teórica. O segundo trata especificamente da computabilidade modelo-teórica sobre números não-padrão. O terceiro aponta para as motivações inicias desta tese, associadas à compreensão da relação entre complexidade computacional e modelos aritméticos. É importante enfatizar o caráter fundacional do objetivo geral.

O programa de Hilbert tinha como meta principal mostrar que poderíamos justificar o uso de entidades infinitárias na matemática, usando apenas métodos finitários [Zach(2003)]. Como dissemos, a computabilidade surgiu em decorrência do programa de Hilbert. Por isso, nos primeiros trabalhos fundacionais em computabilidade, a relação entre finitute e infinitude era uma questão recorrente. Ora, de um ponto de vista de qualquer teoria de primeira ordem finitária que tenha axiomas para os números naturais, tais números são objetos finitários, sejam eles números padrão ou não. De um ponto de vista metateórico conjuntista, a dualidade entre finitude e infinitude está presente na distinção entre números padrão e não-padrão, em virtude de podermos demonstrar que cada número natural não-padrão é maior do que qualquer número natural padrão. Portanto, ao propormos um estudo da computabilidade em modelos da aritmética de primeira ordem, em última análise estamos propondo uma investigação acerca da dualidade entre finitude e infinitude no contexto da computabilidade. Em certo sentido, esta tese propõe um retorno às preocupações originais da teoria da computabilidade.

1.3.2 Justificativa

A proposta de desenvolver uma abordagem modelo-teórica da computabilidade está baseada na intuição de que o estudo das máquinas de Turing através da lógica deve ser situado em um contexto amplo. Em particular, a investigação da computabilidade sobre números naturais não-padrão tem uma relevância patente, uma vez que trata de problemas em aberto que são considerados centrais na atualidade. Além disso, historicamente, a teoria da

computabilidade tem desempenhado um papel importante no estudo dos modelos aritméticos, mas pouco foi feito com relação ao papel que tais modelos podem ter para a teoria da computabilidade.

Para perceber a centralidade da computabilidade à compreensão dos modelos aritméticos, basta considerar que a definição de D. Scott [Scott(1962)] dos *conjuntos de Scott* baseia-se na combinação de técnicas da teoria da computabilidade e teoria dos grafos. Na esteira do trabalho de Scott, H. Friedman [Friedman(1973)] definiu as funções *indicadoras*, as quais identificam segmentos iniciais computáveis dos modelos aritméticos. Usando as funções indicadoras, J. Paris e L. Harrigton [Paris and Harrington(1977)] conseguiram demonstrar o resultado central em teoria dos modelos aritméticos, chamado de *teorema de Paris-Harrington*[8].

Entre os principais trabalhos que tentam estabelecer um vínculo entre computabilidade e modelos aritméticos, devemos mencionar o artigo de G. Kreisel [Kreisel(1965a)], no qual ele demonstra que a definibilidade invariante sobre o modelo padrão da aritmética e a recursividade coincidem. O trabalho de Kreisel mostra que a computabilidade pode ser estudada de um ponto de vista modelo-teórico[9].

Para os fins de análise da computabilidade, apenas M. M. Richter e M. E. Szabo [Richter and Szabo(1983)] estudaram explicitamente algumas propriedades de programas sobre os números naturais não-padrão. Mais tarde, em [Richter and Szabo(1988)] Richter e Szabo investigaram a computação analógica a partir de computações sobre números não-padrão. No entanto, seus trabalhos não explicitam as características da computabilidade sobre os números naturais não-padrão, o único resultado relevante nesse sentido é uma demonstração de que, ao efetuarmos certas generalizações sobre a parada de computações sobre números não-padrão, isso implica que toda função recursiva parcial é total.

[8] Cf. Kaye [Kaye(1991)] para mais detalhes sobre o uso da computabilidade na teoria dos modelos aritméticos.

[9] Cf. [Moschovakis(1969)] para mais referências sobre esse tema.

Recentemente, P. H. Potgieter e E. E. Rosinger [Potgieter and Rosinger(2008), Potgieter and Rosinger(2010)] investigaram a possibilidade de definição de máquinas de Turing com entradas não-padrão, usando ultraprodutos. Entretanto, o foco de Potgieter e Rosinger foi a construção de modelos de hipercomputação. Isso limitou bastante a abordagem das máquinas de ultrafiltros definida pelos autores, a qual não apresenta resultados que permitam compreender a relação entre modelos aritméticos e complexidade computacional.

Há também uma linha de pesquisa na qual se estuda os graus de Turing dos modelos da aritmética. O *grau de Turing* de um modelo enumerável $\mathcal{M}$ da aritmética de Peano de primeira ordem é o grau de insolubilidade do diagrama atômico de $\mathcal{M}$. Os principais resultados nessa área podem ser encontrados no artigo [McAllister(2004)] de A. M. McAllister. Os estudos dos graus de Turing dos modelos não-padrão da aritmética, todavia, também não explicam a relação entre complexidade computacional e modelos não-padrão da aritmética.

Como mencionamos acima, os trabalhos de Paris, Dimitracopoulos, Máté e Sureson mostram a importância da computabilidade em modelos aritméticos não-padrão para a compreensão da complexidade computacional. Ainda não existe um tratamento adequado sobre esse tema, a presente tese constitui-se em uma tentativa de preencher essa lacuna.

1.3.3 Metodologia

Como desenvolver uma abordagem modelo-téorica da computabilidade de Turing que seja capaz de contribuir para a compreensão da relação entre modelos aritméticos e complexidade computacional?

A estratégia será definir uma linguagem de primeira ordem capaz de expressar fatos sobre as computações de máquinas de Turing. Assim, associaremos estruturas e teorias de primeira ordem às computações das máquinas de Turing de modo que a representação diagramática entre máquinas, estru-

turas e teorias de Turing que caracteriza a computabilidade modelo-teórica constitua-se em uma tríplice correspondência com relação às computações. Consequentemente, poderemos verificar as características da computabilidade de Turing, usando lógica de primeira ordem.

Ao desenvolvermos a computabilidade modelo-téorica em lógica de primeira ordem, a existência de modelos não-padrão das teorias de Turing será uma conseqüência imediata. Escolheremos, portanto, uma apresentação das estruturas de Turing que seja semelhante à estrutura dos números naturais, a fim de podermos transferir as análises já existentes na teoria dos modelos aritméticos para o contexto dos modelos não-padrão de teorias de Turing. Uma vez que analisaremos a computabilidade sem pressupormos os números naturais padrão como dados, assumiremos desde o príncipio que o conceito de finitude usado nas máquinas de Turing é relativo ao modelo aritmético considerado.

Como disporemos de recursos modelo-teóricos para investigar as computações em modelos aritméticos, poderemos explicitar as caracterizações modelo-aritméticas de problemas em complexidade computacional, analisando as características da computabilidade sobre números não-padrão. Basicamente, a estratégia será adaptar as técnicas de Paris, Dimitracopoulos, Máté e Sureson, que usam fórmulas aritméticas, para o contexto dos modelos não-padrão de teorias de Turing que definiremos nesta tese.

1.4 Revisão de literatura

Como dito na justificativa, no que tange à computabilidade sobre números naturais não-padrão, pouco foi desenvolvido. Entretanto, existem trabalhos que podemos considerar como antecessores da idéia de uma computabilidade modelo-teórica; no que se segue apresentamos uma descrição daqueles que nos parecem ser amostras representavivas.

Já no artigo original de Turing [Turing(1936)] encontramos uma definição

de uma linguagem de primeira ordem com a capacidade de representar as configurações de máquinas de Turing. Foi justamente a definição de tal linguagem que Turing usou para demonstrar a indecidibilidade da lógica de primeira ordem. Mais tarde, o seu aluno R. Gandy [Gandy(1980), Gandy(1988)] generalizou a linguagem de Turing para máquinas em geral; e não apenas para máquinas baseadas nas computações humanas, tal como Turing o fez. Nesta tese, apresentamos uma linguagem similar a apresentada por G. Boolos *et alli* [Boolos et al.(2002)Boolos, Burgess, and Jeffrey, p.126-136], a qual resulta, por sua vez, de um aprimoramento das linguagens definidas por Turing e Gandy.

Boolos et alli [Boolos et al.(2002)Boolos, Burgess, and Jeffrey, p.126-136] também esboçam uma estrutura, baseada nos números inteiros, a qual representa em certo sentido a estrutura das máquinas de Turing. No entanto, as definições das interpretações dos símbolos relacionais são apresentadas de modo intuitivo. J. Agudelo e W. Carnielli [Agudelo and Carnielli(2010)] efetuaram alguns refinamentos no trabalho de Boolos et alli. Uma vez que o propósito de Agudelo e Carnielli era a análise da relação entre máquinas de Turing paraconsistentes e máquinas de Turing quânticas, eles também não definiram as condições de verdade das sentenças da linguagem de Boolos et alli. A abordagem de Agudelo e Carnielli consistiu em interpretar a linguagem de Boolos et alli através da representabilidade numa axiomática associada a máquinas de Turing.

Um tipo de abordagem da computabilidade no qual as máquinas de Turing foram explicitamente tratadas em termos de estruturas de primeira ordem foi proposto por Y. Gurevich [Gurevich(2000)]. Em realidade, o trabalho de Gurevich é uma generalização da noção de máquina de Turing através do conceito de máquina de estados-abstratos; as máquinas de Turing são um caso particular de máquinas de estados-abstratos. A idéia de Gurevich foi considerar as configurações de uma máquina, o que ele chama de *estados abstratos*, enquanto estruturas de primeira ordem e uma computação enquanto

um sistema dinâmico de mudanças de estados abstratos. N. Dershowitz e Y. Gurevich mostraram em [Dershowitz and Gurevich(2008)] que esse tipo de abordagem permite apresentar uma axiomatização para o conceito de computabilidade. Mais do que isso, Dershowitz e Gurevich propuseram uma demonstração das teses de Church e Turing a partir de uma axiomática para máquinas de estados-abstratos.

A proposta de Gurevich é relevante para muitas aplicações em ciência da computação, mas ela não é adequada para analisarmos o papel da distinção entre finitude e infinitude da computabilidade. Ademais, essa abordagem, tal como está, não permite a definição das condições de verdade de sentenças de uma linguagem capaz de representar as configurações das máquinas de Turing. Para isso, precisamos de interpretações dos símbolos relacionais usados para expressar fatos sobre as máquinas de Turing que, por um lado, sejam fixas, mas, por outro, dependam da dinâmica das computações de uma máquina de Turing.

Existem muitas formulações abstratas das máquinas de Turing. Podemos mencionar os trabalhos de A. Nerode[10], a abordagem de A. Gajardo e J. Mazoyer [Gajardo and Mazoyer(2007)] via sistemas dinâmicos, ou ainda outras abordagens nas quais máquinas de Turing são definidas enquanto estruturas para fins diversos. Por exemplo, nos trabalhos de F. Cohen [Cohen(1987), Cohen(1989)] sobre formalizações dos vírus computacionais, aparecem definições de máquinas de Turing em termos de estruturas, mas não de primeira ordem. Em complexidade descritiva [Immerman(1999)] ou em teoria dos modelos finitos [Ebbinghaus and Flum(1999)] também encontramos descrições das operações das máquinas de Turing usando lógica. Existem tantas possibilidades de formalização das máquinas de Turing através da lógica e possibilidades de extensões que E. Steinhart em [Steinhart(2002)] até tentou apresentar um tratamento sobre quais são as abstrações possíveis das máquinas de Turing.

[10] Cf. [Remmel and Crossley.(1993)]

Ficará claro ao longo desta tese que estamos propondo algo bastante diverso desses trabalhos prévios. A computabilidade modelo-teórica não objetiva apresentar abstrações ou formalizações das máquinas de Turing. O objetivo é avaliar a relação entre máquinas de Turing, estruturas de Turing e teorias de Turing, a fim de compreender as características da computabilidade e, por extensão, da complexidade computacional.

1.5 Resultados

No capítulo 2 a tese apresenta uma abordagem modelo-teórica da computabilidade em lógica de primeira ordem. Inicialmente, uma interpretação fenomenológica da análise de Turing é proposta. As definições 2.3.2 de estrutura de Turing e 2.4.1 de teoria de Turing são pontos cruciais do capítulo 2, pois dessas definições derivam-se os seus principais resultados: os teoremas 2.3.1 e 2.5.2.

No capítulo 3 algumas características da computabilidade são analisadas a partir de um ponto de vista modelo-teórico. Nesse capítulo uma série de resultados é apresentada. Destacamos os seguintes resultados: 3.2.1, 3.3.2, 3.4.1, 3.4.2, 3.5.1 e 3.5.2. Esses resultados conjuntamente mostram a potencialidade de uma abordagem modelo-teórica para o estudo das características da computabilidade.

No capítulo 4 a computabilidade em modelos aritméticos não-padrão é investigada. Nessa ocasião, vários resultados acerca da computabilidade sobre números não-padrão são formulados e demonstrados, dentre os quais destacamos estes: 4.2.1, 4.2.1, 4.3.1, 4.3.2, 4.4.1, 4.4.2, 4.5.1, 4.5.2 e 4.5.3. Tais resultados possibilitam uma explicação para o problema $P1$ formulado nesta tese. A existência de um *princípio de internalidade aritmética* na computabilidade é descoberto em decorrência disso, segundo o qual o próprio conceito de computação é relativo ao modelo aritmético em que as máquinas de Turing operam

No capítulo 5 são formuladas caracterizações modelo-aritméticas global e local do problema P versus NP. Os principais resultados desse capítulo são estes: 5.2.1, 5.3.1, 5.4.1, 5.4.2 e 5.5.2. Esses resultados indicam uma barreira aritmética para a possibilidade de solução do problema P versus NP. Ademais, eles também possibilitam uma explicação para o problema $P2$ formulado nesta tese.

As definições e resultados indicados aqui são formulações originais do autor desta tese. Quando algum resultado foi obtido de trabalhos prévios, as referências estão indicadas ao longo da tese.

2. COMPUTABILIDADE MODELO-TEÓRICA CLÁSSICA DE PRIMEIRA ORDEM

Am Anfang ist das Zeichen.
David Hilbert

Este capítulo formula as bases de uma abordagem modelo-teórica da computabilidade clássica via lógica de primeira ordem. Na seção 2.1, apresentamos a análise de Turing da computabilidade, interpretando-a enquanto uma análise fenomenológica da computação. Na seção 2.2, formalizamos as máquinas de Turing clássicas, i.e., máquinas de Turing que computam funções sobre números naturais. Na seção 2.3, associamos estruturas de primeira ordem às máquinas de Turing clássicas. Na seção 2.4, definimos teorias de primeira ordem para cada máquina de Turing. Na seção 2.5, analisamos a completude das teorias de Turing relativamente às estruturas de Turing.

2.1 A análise de Turing da computabilidade

Em 1936, A. M. Turing [Turing(1936)] definiu o modelo de computação hoje conhecido como *máquina de Turing*. Para obter essa definição, ele analisou o que acontece quando uma pessoa calcula mecanicamente sobre uma folha de papel. Turing [Turing(1936), p.249-252] constatou que alguns aspectos desse fenômeno não são essenciais para sua mecanicidade; por exemplo, o tamanho da folha e sua bidimensionalidade, o tamanho da letra, o tipo de material usado para escrever, etc. A partir dessa análise, Turing determinou os *componentes* minimais que definem uma pessoa enquanto uma pessoa que

efetua uma computação:

1. Uma *fita* unidimensional linear dividida em celas;
2. Um *cabeçote* conectado à fita com a capacidade de efetuar as seguintes operações: (1) ler e escrever uma quantidade finita de símbolos nas celas e (2) mover-se à direita ou à esquerda uma quantidade finita de celas.

No que diz respeito ao processo enquanto um processo mecânico, Turing concluiu que os componentes interagem da seguinte forma:

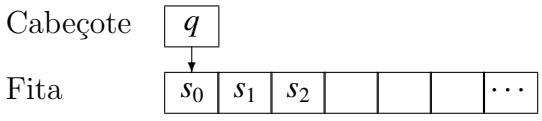
T1 O cabeçote tem uma quantidade finita de estados internos;

T2 Para executar suas operações, o cabeçote segue uma quantidade finita de instruções ordenadas por seus estados internos;

T3 As instruções no cabeçote determinam completamente as ações efetuadas sobre as celas da fita em cada instante;

T4 Apenas um parcela finita da fita pode ser visualizada pelo cabeçote em cada instante.

Para Soare [Soare(1987), p.12] podemos dizer que o *modelo intencionado* de uma máquina de Turing é uma estrutura na qual seu conjunto de instruções está dentro do cabeçote da máquina e a fita está associada ao cabeçote da seguinte forma:



Onde o número q dentro do cabeçote indica o estado da máquina de acordo com seu conjunto de instruções, e a seqüência $s_0 s_1 s_2$ são os símbolos

inscritos sobre a fita, sendo que a máquina no momento está lendo o primeiro. Os três pontos a direita na fita indicam que ela é ilimitada à direita[1].

Como mencionamos na introdução desta tese, Turing desenvolveu essa análise do processo de computação tendo em vista um dos problemas levantados por D. Hilbert acerca da noção de *procedimento efetivo* ou *algoritmo*. Esse problema era chamado *problema da decisão para a lógica de primeira ordem* ou *Entscheidungsproblem* e consistia em saber se existiria um algoritmo para determinar quando uma fórmula da lógica de primeira ordem é uma verdade lógica. Por isso, a primeira aplicação vislumbrada por Turing de seu modelo de computação consistiu justamente numa resposta negativa a tal problema.

No mesmo ano, 1936, A. Church [Church(1936)] também demonstrou que não existe um algoritmo para determinar quando uma fórmula de primeira ordem é uma verdade lógica. No entanto, para K. Gödel [Gödel(1986d)], a demonstração de Church não era conclusiva porque ela estava baseada no pressuposto de que todos os procedimentos efetivos podem ser formalizados de acordo com o conceito de funções recursivas. Essa afirmação foi chamada por Kleene [Kleene(1943)] de *tese de Church*. Para Gödel [Gödel(1986d)], somente o modelo de computação de Turing apresentou a análise conclusiva do conceito de *algoritmo*. Isso porque as restrições *finitárias* associadas ao conceito de algoritmo estão explicitamente definidas na análise de Turing, basta observar que todas as ações de uma máquina de Turing tem algum limitante. Ademais, as máquinas de Turing são um modelo de computação *simples* e *intuitivo*, o que garante, segundo Gödel, a proficuidade da análise de Turing.

Contrariamente a Gödel, E. Post [Post(1994)] sustentou que a análise de Turing não é conclusiva, ela é apenas uma "análise psicológica do procedimento mental envolvida no processo combinatório matemático" que possi-

[1] Na verdade, Soare representa a fita como sendo infinita em ambas as direções, mas sabemos que basta uma única direção [Lewis and Papadimitriou(1998), p.207-208].

bilita uma definição mais precisa do conceito de *algoritmo*. No entanto, a análise de Turing depende de confirmação empírica, sendo a verdade da tese de Church uma questão de fato a ser testada constantemente.

Para R. Gandy [Gandy(1988)] e W. Sieg [Sieg(2002)], Post não levou em conta que a análise de Turing tem dois passos que a distinguem, sobremaneira, da análise de Church. Primeiro, Turing definiu o conceito de computor, a saber: um *computor* é um ser humano sem limitações de tempo e espaço que satisfaz as condições $T1 - T4$. Segundo, Turing de fato demonstrou o seguinte teorema:

Teorema de Turing: Se uma operação é efetivamente calculável por um computor, então ela é computável por uma máquina de Turing.

A *tese de Turing* consiste na afirmação de que se uma função é efetivamente calculável, então ela pode ser computada por uma máquina de Turing. O enunciado da tese de Turing é mais forte do que o enunciado do teorema de Turing, porque este se limita às operações calculáveis por um computor, ao passo que aquele afirma algo sobre operações computáveis em geral. Como indicado também por R. I. Soare [Soare(1999)], a tese de Turing é *intensionalmente* diferente da tese de Church pois trata de algoritmos em geral enquanto a tese de Church refere-se aos algoritmos que podem ser representados através de funções numéricas. A tese de Church é equivalente a tese de Turing *módulo* codificação, ou seja, elas são apenas *extensionalmente* equivalentes.

O intrigante é que na demonstração do teorema de Turing, ele explicitamente diz que seu argumento faria "um apelo direto à intuição" [Turing(1936), p.249]. Muitos trabalhos foram escritos discutindo se Turing teria ou não apresentado uma demonstração, mas é incrível que ninguém tenha percebido que a análise de Turing é uma análise fenomenológica. Do ponto de vista do autor desta tese, essa é a razão pela qual a análise de Turing é conclusiva com relação ao conceito de computor.

O método fenomenológico foi formulado por E. Husserl em vários trabalhos, sobretudo, em [Husserl(1901(2008)), Husserl(1913(2008))]. Segundo W. Stegmüller [Stegmüller(1976), 49-95], uma análise fenomenológica consiste basicamente de três passos: *epoché*, *redução eidética* e *redução transcendental*. Na *epoché* suspendemos nossos juízos prévios acerca de um dado fenômeno. Na *redução eidética* eliminamos os componentes não essenciais à compreensão do sentido de um fenômeno. Na *redução transcendental* os dados da consciência particular são universalizados para uma consciência transcendental pura, ou seja, esses dados são transformados em um conceito abstrato válido para qualquer ser humano.

Claramente, Turing efetuou uma análise fenomenológica do processo de computação. A *epoché* na análise de Turing consistiu em suspender o juízo ao olhar para uma pessoa que efetua ações seguindo regras. A *redução eidética* é visível. A fita surge como uma abstração da página sobre a qual uma pessoa efetua a computação, e Turing apresentou argumentos para mostrar que vários aspectos são inessenciais, exceto o fato de termos uma fita unidimensional dividida em celas. O cabeçote é uma abstração da pessoa lendo e escrevendo símbolos sobre uma página. A limitação na amplitude da visão de uma pessoa expressa-se na forma da localidade da leitura do cabeçote. As instruções representam o procedimento que a pessoa segue ao efetuar suas ações, e os estados representam a memória da pessoa que efetua a computação, ambos necessários, pois sem memória não podemos seguir regras, uma vez que não temos outra maneira de saber em que estágio da computação nos encontramos. A *redução transcendental* é justamente o argumento de que qualquer pessoa chegará inevitavelmente ao conceito de máquina de Turing quando analisar quais são os componentes necessários e suficientes para que um fenômeno possa ser considerado um procedimento computável.

Compreender a análise de Turing enquanto uma espécie de análise fenomenológica explica sua aceitação. Ela é intuitiva porque uma análise fenomenológica baseia-se em nossas intuições associadas a uma dada experiência, no

caso em questão nossa experiência de efetuar procedimentos mecânicos. Ela é simples porque uma análise fenomenológica reduz nossa compreensão do fenômeno estudado ao essencial e tem, portanto, um cunho transcendental. Esse aspecto transcendental é o que garante a equivalência entre computores e máquinas de Turing.

Neste capítulo mostraremos como o conceito de *máquina de Turing* pode ser tratado em linguagens de primeira ordem. Em especial, veremos que a distinção entre finitude e infinitude está presente na análise fenomenológica de Turing, sendo, em certo sentido, a insolubilidade do problema da decisão uma conseqüência das limitações finitárias das máquinas de Turing. Esse será o primeiro resultado que exibe as potencialidades de uma abordagem modelo-teórica da computabilidade clássica.

2.2 Máquinas de Turing

Existem várias maneiras de formalizar o conceito de máquina de Turing, porque isso depende de diversos fatores, sobretudo, o universo de objetos considerados e a representação desses objetos como entradas e saídas nas máquinas de Turing. Em conformidade com as práticas usuais em teoria de computabilidade[2], nesta tese adotaremos as seguintes convenções:

Convenção sobre o universo: Restringiremos, sem perda de generalidade, a computabilidade às funções definidas sobre os números naturais (padrão ou não), as quais chamaremos de *funções numéricas*.

Convenção sobre a representação: Usaremos, sem perda de generalidade, a *representação binária* dos números naturais (padrão ou não) nas máquinas de Turing.

Note a ênfase de que essas convenções não implicam em perda de generalidade. A restrição às operações numéricas é inócua porque toda seqüência fi-

[2] [Odifreddi(1992b), Cooper(2004)]

nita de símbolos pode ser representada de forma efetiva como uma seqüência finita de números naturais e, portanto, toda operação sobre símbolos arbitrários pode ser considerada como uma operação numérica. Já a representação binária dos números naturais também pode ser feita de forma efetiva e, por conseguinte, não implica em alterações na classe das funções numéricas estudadas[3].

Observação 2.2.1. *Nas convenções sobre o universo e a representação já ocorre uma instância metateórica do conceito de finitude. No capítulo 4, veremos que a noção de finitude usada em computabilidade deve sempre ser relativa ao modelo aritmético fixado.*

Devido à observação a 2.2.1, precisamos de algumas definições adicionais. Considere o conjunto de símbolos aritméticos usuais $\{0, 1, +, \times, <\}$ como a *assinatura aritmética*. A estrutura intencionada da assinatura aritmética é o *sistema dos números naturais* $\mathcal{N}$, cujo domínio são os números naturais $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$, os elementos designados são o número 0 e o número 1, as operações são a adição + e a multiplicação $\times$, e a relação é a ordem estrita $<$. A *aritmética de primeira ordem completa*, ou simplesmente *aritmética*, é o conjunto de todas as sentenças da linguagem gerada pela assinatura aritmética que são verdadeiras no sistema dos números naturais; denotamos essa teoria por $Th(\mathcal{N})$. O *modelo aritmético padrão* de $Th(\mathcal{N})$ é $\mathcal{N}$, qualquer outro modelo de $Th(\mathcal{N})$ que não seja isomórfico à $\mathcal{N}$ é chamado de *modelo aritmético não-padrão*. No capítulo 4, apresentaremos uma discussão um pouco mais detalhada sobre essas definições, por hora basta apenas fixá-las. Feito isso, definiremos máquinas de Turing como funções, na mesma linha de H. B. Enderton [Enderton(1977), p.531].

Definição 2.2.1. *Seja $\mathcal{M}$ um modelo aritmético. Uma* Máquina de Turing *é uma função M tal que, para algum $n \in dom(\mathcal{M})$ com $n \geq 0$,*

$$dom(M) \subseteq \{0, 1, \ldots, n\} \times \{0, 1, 2\},$$

[3] [Boolos et al.(2002)Boolos, Burgess, and Jeffrey, p.12-14]

$$cod(M) \subseteq \{0, 1, \ldots, n\} \times \{0, 1, 2\} \times \{\triangleleft, \triangleright\},$$

Onde:

1. *Os elementos do primeiro fator de* $dom(M)$ *e* $cod(M)$ *são os* estados *de* M*, sendo* 0 *o* estado inicial *e* n *o* estado final*;*

2. *Os elementos do segundo fator de* $dom(M)$ *e* $cod(M)$ *são os* símbolos *de* M*, sendo* 0 *o símbolo* zero*,* 1 *o símbolo* um*, e* 2 *o símbolo* branco*;*

3. *Os elementos do terceiro fator de* $cod(M)$ *são os* movimentos *de* M*, sendo* $\triangleleft$ *o movimento* à esquerda *e* $\triangleright$ *o movimento* à direita.

As quintuplas em M *são chamadas* instruções *da máquina de Turing.*

Intuitivamente, uma instrução da forma $(q, s, p, r, \triangleleft)$ significa que a máquina ao estar no estado q, lendo o símbolo s, muda para o estado p, escreve o símbolo r no lugar de s e movimenta-se uma cela à esquerda. Se a máquina estiver no começo da fita, ela fica na mesma cela.

Uma máquina de Turing é um objeto que efetua computações. Essas computações são, por sua vez, transições entre configurações da máquina. A partir da definição formal de máquina de Turing 2.2.1, também podemos definir formalmente suas computações, mas agora na linha de R. I. Soare [Soare(1987), p.12].

Definição 2.2.2. *Seja* $\mathcal{M}$ *um modelo aritmético,* M *uma máquina de Turing com estados em* $\{0, 1, \ldots, n\}$ *e* $\{0, 1, 2\}^\star$ *o conjunto de todas as seqüência finitas, no sentido de* $\mathcal{M}$*, formadas por símbolos de* M*. Assim, uma* computação *de* M *com* entrada $\vec{s} = s_1, \ldots, s_m \in \{0, 1, 2\}^\star$ *é uma seqüência parcial ou total* $C_M^{\vec{s}} : dom(\mathcal{M}) \to \{0, 1, 2\}^\star \times \{0, 1, \ldots, n\} \times \{0, 1, 2\}^\star$ *definida em* $dom(\mathcal{M})$ *da seguinte forma:*

1. $C_M^{\vec{s}}(0) = (\oslash, 0, (2, 2, s_1, \ldots, s_m, 2, 2))$*. Essa seqüência é chamada de* configuração inicial *de* M[4]*.*

[4] Os símbolos brancos "22" antes e depois da entrada $s_1, \ldots, s_m$ são necessários para delimitar o começo e o final da entrada, pois o símbolo branco '2' pode ocorrer em $s_1, \ldots, s_m$.

2. *Se* $C_M^{\vec{s}}(i) = ((a_1, \ldots, a_k), q, (b_1, \ldots, b_l))$ *e* $M(q, b_1)$ *está definida, então* $C_M^{\vec{s}}(i+1)$ *também está definida, é chamada* configuração instantânea *de* M *no* passo $i+1$*, e tem a seguinte forma:*

   (a) $C_M^{\vec{s}}(i+1) = ((a_1, \ldots, a_k, s), p, (b_2, \ldots, b_l))$ *se* $M(q, b_1) = (p, s, \triangleright)$ *e* $(b_2, \ldots, b_l) \neq \oslash$*;*

   (b) $C_M^{\vec{s}}(i+1) = ((a_1, \ldots, a_k, s), p, 2)$ *se* $M(q, b_1) = (p, s, \triangleright)$ *e* $(b_2, \ldots, b_l) = \oslash$*;*

   (c) $C_M^{\vec{s}}(i+1) = ((a_1, \ldots, a_{k-1}), p, (a_k, s, b_2, \ldots, b_l))$ *se* $M(q, b_1) = (p, s, \triangleleft)$ *e* $(a_1, \ldots, a_k) \neq \oslash$*;*

   (d) $C_M^{\vec{s}}(i+1) = (\oslash, p, (s, b_2, \ldots, b_l))$ *se* $M(q, b_1) = (p, s, \triangleleft)$ *e* $(a_1, \ldots, a_k) = \oslash$*.*

3. *Se* $C_M^{\vec{s}}(i) = ((a_1, \ldots, a_k), q, (b_1, \ldots, b_l))$ *e* $M(q, b_1)$ *não está definida, então* $C_M^{\vec{s}}(j)$ *não está definida para todo* $j > i$*. Nesse caso,* $C_M^{\vec{s}}(i)$ *é chamada de* configuração final de M *e* $C_M^{\vec{s}}$ *é dita uma* computação de $i+1$ passos*.*

Dada uma configuração $C_M^{\vec{s}}(i) = ((a_1, \ldots, a_k), q, (b_1, \ldots, b_l))$*, o símbolo* b_1 *é chamado de* símbolo corrente *e a seqüência de símbolos* $(2, 2, c_1, \ldots, c_l, 2, 2)$ *com* $c_j \in \{a_1, \ldots, a_k, b_1, \ldots, b_l\}$ *é chamada de* conteúdo da fita*.*

Para D. Barker-Plummer [Barker-Plummer(2004)], o modelo intencionado de uma máquina de Turing não tem um papel explícito na definição das máquinas de Turing, porque podemos definir uma computação apenas em termos de símbolos e estados, sem fazermos referência aos conceitos de fita ou cabeçote. Na visão de Barker-Plummer, o modelo intencionado tem uma importância apenas histórica. Na definição 2.2.2 fica claro que essa afirmação é inapropriada, pois as seqüências de símbolos em uma configuração estão definidas a partir da estrutura intencionada da fita e a mudança de símbolo corrente representa os deslocamentos do cabeçote. O que

possibilitou a afirmação de Barker-Plummer foi o fato de que aparentemente podemos definir as configurações desconsiderando as características da fita, mas as cláusulas 2b e 2d da definição 2.2.2 mostram que isso não é verdade.

Em uma abordagem modelo-teórica da computabilidade, seremos capazes de representar o modelo intencionado de uma máquina de Turing mediante estruturas de primeira ordem. Antes disso, precisamos definir quando uma máquina de Turing computa uma função numérica.

Convenção sobre a parada de uma máquina de Turing : Uma máquina de Turing M *pára* para a entrada $\vec{s} \in \{0,1,2\}^*$ se, e somente se, a computação $C_M^{\vec{s}}$ de M com entrada $\vec{s}$ tem uma configuração final; caso contrário, M *não pára*. Se M pára para $\vec{s}$, escrevemos $M[\vec{s}] \downarrow$, mas se a máquina M não pára, escrevemos $M[\vec{s}] \uparrow$. Ademais, escrevemos $M[\vec{s}] = \vec{r}$ com $\vec{r} = r_1, \ldots, r_l \in \{0,1,2\}^*$ para indicar que M pára para $\vec{s}$ em uma configuração final $C_M^{\vec{s}}(i) = ((a_1, \ldots, a_k), n, (2, 2, r_1, \ldots, r_l, 2, 2, b_{l+3}, \ldots, b_{l+j}))$, onde n é o estado final de M, e, nesse caso, dizemos que $\vec{r}$ é a *saída* de $C_M^{\vec{s}}$. Em algumas ocasiões, quando não houver risco de confusão, escreveremos simplesmente $M[a] = b$ para indicar que a máquina de Turing M tendo como entrada a representação binária de a gera como saída a representação binária de b.

Definição 2.2.3. *Seja $\mathcal{M}$ um modelo aritmético e $\underline{m}$ a representação binária do número natural $m \in dom(\mathcal{M})$. Uma máquina de Turing M* computa *a função numérica (parcial) $f : dom(\mathcal{M})^m \to dom(\mathcal{M})$ quando, para todo $x_1, \ldots, x_n$:*

- *$f(\underline{x}_1, \ldots, \underline{x}_n) = \underline{y}$ se, e somente se, $M[22\underline{x}_1 2\underline{x}_2 2 \cdots 2\underline{x}_n 22] = \underline{y}$,*
- *$f(\underline{x}_1, \ldots, \underline{x}_n)$ não está definida se, e somente se, $M[\vec{s}] \uparrow$.*

Uma função é Turing computável *quando existe uma máquina que a computa.*

Com essa última definição, concluímos a formalização das máquinas de Turing suficiente para os propósitos desta tese. Como trataremos de funções numéricas definidas sobre modelos aritméticos padrão ou não, convencionamos distingui-las entre *padrão* e *não-padrão*, conforme seja necessário determinar, respectivamente, que uma função toma como entrada apenas números naturais padrão ou pode também abranger os números naturais não-padrão.

2.3 Estruturas de Turing

Na seção anterior apresentamos uma formalização do conceito de máquina de Turing, agora definiremos uma linguagem capaz de expressar fatos sobre máquinas de Turing. Na seqüência mostraremos como associar estruturas de Turing a máquinas de Turing.

Definição 2.3.1. *Seja $\mathcal{M}$ um modelo aritmético e M uma máquina de Turing definida em $\mathcal{M}$. A* assinatura de Turing *é a assinatura $\mathcal{S} = \{\underline{n} : n \in dom(\mathcal{M})\} \cup \{+1, -1, \prec, H, T\}$, cujos símbolos são, respectivamente, as constantes $\underline{n}$ para cada* número natural *$n \in dom(\mathcal{M})$, os símbolos funcionais unários para as funções* sucessor *e* predecessor *(truncada), o símbolo relacional binário para a* relação de ordem*, os símbolos relacionais ternários para o* cabeçote *e a* fita*. A* linguagem de Turing *$\mathcal{L}$ é a linguagem de primeira ordem gerada por $\mathcal{S}$. Em particular, a* linguagem de Turing restrita *é a linguagem de primeira ordem gerada pela assinatura $\{\underline{n} : n \in dom(\mathcal{M})\} \cup \{+1, -1, \prec\}$.*

Pressuporemos todas as noções sintáticas tal como definidas em [Hinman(2005)]. Em particular, denotaremos a *relação de demonstrabilidade* clássica por $\vdash$. Por simplicidade, escreveremos $\overline{n}$ e $-\overline{n}$ como abreviações do termo $\underline{0}$ seguido, respectivamente, por n aplicações de $+1$ e -1; em particular, chamaremos de *numerais* os termos da forma $\overline{n}$, ou seja, $\underline{0}$ seguindo por n aplicações de $+1$. Dado um modelo aritmético $\mathcal{M}$, assumiremos que, para todo numero natural padrão n em $dom(\mathcal{M})$, $\underline{n}$ é justamente o numeral $\overline{n}$.

Em uma abordagem modelo-teórica da computabilidade, atribuímos significado aos símbolos da linguagem de Turing através da associação de uma $\mathcal{S}$-estrutura a cada máquina de Turing, tendo como parâmetro uma entrada. Tomar como parâmetro uma determinada entrada é importante para analisarmos as computações das máquinas de Turing.

Definição 2.3.2. *Seja $\mathcal{M}$ um modelo aritmético e M uma máquina de Turing definida em $\mathcal{M}$. Uma* estrutura de Turing *associada à máquina de Turing M com entrada $\vec{s} = s_1, \ldots, s_n$ é uma $\mathcal{S}$-estrutura de primeira ordem $\mathcal{A}_M^{\vec{s}} = (dom(\mathcal{A}_M^{\vec{s}}), \{\underline{n}^{\mathcal{A}_M^{\vec{s}}}\}_{n \in dom(\mathcal{M})}, +1^{\mathcal{A}_M^{\vec{s}}}, -1^{\mathcal{A}_M^{\vec{s}}}, \prec^{\mathcal{A}_M^{\vec{s}}}, H^{\mathcal{A}_M^{\vec{s}}}, T^{\mathcal{A}_M^{\vec{s}}})$ onde:*

1. $dom(\mathcal{A}_M^{\vec{s}}) = dom(\mathcal{M})$;
2. $\underline{n}^{\mathcal{A}_M^{\vec{s}}}$ *é o número* n *em* $dom(\mathcal{M})$, *para cada* $\underline{n} \in \mathcal{S}$;
3. $+1^{\mathcal{A}_M^{\vec{s}}}$ *é a função sucessor sobre* $dom(\mathcal{M})$;
4. $-1^{\mathcal{A}_M^{\vec{s}}}$ *é a função predecessor (truncada) sobre* $dom(\mathcal{M})$;
5. $\prec^{\mathcal{A}_M^{\vec{s}}}$ *é a relação de ordem estrita sobre* $dom(\mathcal{M})$;
6. $H^{\mathcal{A}_M^{\vec{s}}}$ *e* $T^{\mathcal{A}_M^{\vec{s}}}$ *são as menores relações em* $dom(\mathcal{M})^3$ *tais que:*
   - $(0,0,0) \in H^{\mathcal{A}_M^{\vec{s}}}$, $(2,0,0) \in T^{\mathcal{A}_M^{\vec{s}}}, (2,1,0) \in T^{\mathcal{A}_M^{\vec{s}}}, (s_1,2,0) \in T^{\mathcal{A}_M^{\vec{s}}}, \ldots, (s_n, n+1, 0) \in T^{\mathcal{A}_M^{\vec{s}}}, (2, n+2, 0) \in T^{\mathcal{A}_M^{\vec{s}}}, (2, n+3, 0) \in T^{\mathcal{A}_M^{\vec{s}}}$ *e, para todo* $y > n+3$, $(2, y, 0) \in T^{\mathcal{A}_M^{\vec{s}}}$;
   - *Se* $(q,e,t) \in H^{\mathcal{A}_M^{\vec{s}}}$, $(s,e,t) \in T^{\mathcal{A}_M^{\vec{s}}}$ *e* $(q,s,p,r,\triangleright) \in M$, *então* $(p, e+1, t+1) \in H^{\mathcal{A}_M^{\vec{s}}}$, $(r,e,t+1) \in T^{\mathcal{A}_M^{\vec{s}}}$ *e* $(r',c,t+1) \in T^{\mathcal{A}_M^{\vec{s}}}$ *para todo* c *tal que* $c \neq e$ *e* $(r',c,t) \in T^{\mathcal{A}_M^{\vec{s}}}$;
   - *Se* $(q,e,t) \in H^{\mathcal{A}_M^{\vec{s}}}$, $(s,e,t) \in T^{\mathcal{A}_M^{\vec{s}}}$ *e* $(q,s,p,r,\triangleleft) \in M$, *então* $(p, e-1, t+1) \in H^{\mathcal{A}_M^{\vec{s}}}$, $(r,e,t+1) \in T^{\mathcal{A}_M^{\vec{s}}}$ *e* $(r',c,t+1) \in T^{\mathcal{A}_M^{\vec{s}}}$ *para todo* c *tal que* $c \neq e$ *e* $(r',c,t) \in T^{\mathcal{A}_M^{\vec{s}}}$.

Dada uma tripla $(q,e,t) \in H^{\mathcal{A}^{\vec{s}}_M}$ *e uma tripla* $(s,e,t) \in T^{\mathcal{A}^{\vec{s}}_M}$ *chamamos* q *de* estado*,* s *de* símbolo*,* e *de* cela *e* t *de* instante*. O conjunto de todas as celas é chamado* espaço *e o conjunto de todos os instantes é chamado* tempo[5]*.*

Doravante, pressuporemos nas definições de máquinas de Turing um modelo aritmético $\mathcal{M}$ previamente fixado, mencionando-o apenas quando for relevante. Ademais, quando for necessário distinguir estados, símbolos, celas e instantes e não houver risco de confusão, convencionaremos usar as letras q e p para denotar constantes que nomeiam estados, s e r para denotar constantes que nomeiam símbolos, e e c para denotar constantes que nomeiam celas, t e u para denotar constantes que nomeiam instantes. Em especial, a letra n em fórmulas da forma $H(n,e,t)$ sempre designará o estado final da máquina de Turing em questão.

Observação 2.3.1. *Poderíamos facilmente apresentar uma definição mais geral do que 2.3.2, na qual estruturas de Turing estariam associadas a máquinas de Turing* M *sem depender da entrada. Uma vez que podemos listar e ordenar as entradas, por exemplo, em ordem lexicográfica, bastaria definir as relações cabeçote e fita como relações quaternárias* $H(q,e,t,i)$ *e* $T(s,e,t,i)$ *onde o parâmetro* i *especificaria a entrada considerada na lista. Esse expediente é necessário, pois, caso contrário, poderia haver conflito na aplicação das instruções. Na medida em que estamos interessados apenas nas características das computações, não precisamos de tanta generalidade.*

Quando não houver risco de confusão, os superescritos nos componentes de uma estrutura de Turing serão omitidos. Assim, denotaremos uma estrutura de Turing $\mathcal{A}^{\vec{s}}_M$ por $(dom(\mathcal{A}^{\vec{s}}_M), \{\underline{n}\}_{n\in\mathcal{M}}, +1, -1, <, H, T)$. Também pressuporemos as definições de todas as noções semânticas tal como apresentadas em

[5] Note que poderíamos apresentar uma estrutura de Turing enquanto uma estrutura de primeira ordem quadrisortida, a fim de efetuar essa distinção entre estados, símbolos, espaço e tempo. No entanto, sabemos que a expressividade da lógica polisortida é igual a expressividade da lógica de primeira ordem, então isso seria uma complicação desnecessária [Hodges(2005), p.202-206].

[Hinman(2005)]. Em particular, denotaremos a *relação de satisfatibilidade* clássica por $\vDash$.

O resultado abaixo apresenta a primeira justificativa técnica para o desenvolvimento da computabilidade modelo-teórica. Ele expressa a relação entre máquinas de Turing e estruturas de Turing que precisamos para os fins de análise das computações em modelos aritméticos.

Definição 2.3.3. *Seja M uma máquina de Turing. Uma $\mathcal{L}$-fórmula* $\varphi(x_1, \ldots, x_n)$ corresponde *à configuração $C_M^{\vec{s}}(t)$ da computação $C_M^{\vec{s}}$ quando*

$$\mathcal{A}_M^{\vec{s}} \vDash \varphi(r_1, \ldots, r_k, q, r'_1, \ldots, r'_m, e, e+1, \ldots, e+k+m-1, t) \text{ se, e somente se, } C_M^{\vec{s}}(t) = ((r_1, \ldots, r_k), q, (r'_1, \ldots, r'_m)).$$

Dizemos que uma configuração $C_M^{\vec{s}}(t)$ é correspondível *na linguagem de Turing $\mathcal{L}$ quando existe uma fórmula de $\mathcal{L}$ que corresponde à $C_M^{\vec{s}}(t)$.*

Teorema 2.3.1 (Correspondência)**.** *Seja $\mathcal{M}$ um modelo aritmético, M uma máquina de Turing definida em $\mathcal{M}$, e $C_M^{s_1,\ldots,s_n}$ uma computação de M. Então toda configuração em $C_M^{\vec{s}}$ é correspondível na linguagem de Turing $\mathcal{L}$.*

Demonstração. A demonstração é por indução sobre os passos de $C_M^{\vec{s}}$, internamente em $\mathcal{M}$. No passo 0, $C_M^{\vec{s}}(0) = (\oslash, 1, (2, 2, s_1, \ldots, s_n, 2, 2))$. Nesse caso, a fórmula $H(1,0,0) \wedge T(2,0,0) \wedge T(2,1,0) \wedge T(s_1,2,0) \wedge \cdots \wedge T(s_n, n+1, 0) \wedge T(2, n+2, 0) \wedge T(2, n+3, 0) \wedge \forall y(n+3 \prec y \rightarrow T(2, y, 0))$ de $\mathcal{L}$ interna em $\mathcal{M}$ corresponde à configuração $C_M^{\vec{s}}(0)$.

Suponha que para todos os passos menores do que t a máquina M está definida e no passo t a fórmula $\varphi(r_1, \ldots, r_k, q, r'_1, \ldots, r'_m, e, e+1, \ldots, e+k+m-1, t)$ interna em $\mathcal{M}$ corresponde à configuração $C_M^{\vec{s}}(t) = ((r_1, \ldots, r_k), q, (r'_1, \ldots, r'_m))$. A definição 2.2.2 estabelece que temos quatro casos a considerar, conforme $M(q, r'_1)$ esteja definida - caso $M(q, r'_1)$ não esteja definida, então a hipótese de indução já garante o resultado. Ou $M(q, r'_1) = (p, s, \triangleleft)$ ou $M(q, r'_1) = (p', s', \triangleright)$ no passo t (possivelmente $p = p'$ e $s = s'$).

Considere o primeiro caso. Assim, existem duas possibilidades ou $(r_1, \ldots, r_k) = \oslash$ ou $(r_1, \ldots, r_k) \neq \oslash$. Analisaremos o primeiro subcaso - o outro subcaso é análogo. Por hipótese de indução, $\mathcal{A}^{\vec{s}}_M \vDash \varphi(r_1, \ldots, r_k, q, r'_1, \ldots, r'_m, e, e+1, \ldots, e+k+m-1, t)$ e, por definição de estrutura de Turing, $\mathcal{A}^{\vec{s}}_M \vDash H(q, e+k, t) \wedge T(r_1, e, t) \wedge T(r_2, e+1, t) \wedge \cdots \wedge T(r_k, e+k-1, t) \wedge T(r'_1, e+k, t) \wedge T(r'_2, e+k+1, t) \wedge \cdots \wedge T(r'_m, e+k+m-1, t) \wedge \forall y(\neg y \approx e \wedge \neg y \approx e+1 \wedge \cdots \wedge \neg y \approx e+k+m-1 \rightarrow T(2, y, t))$. Consequentemente, $\mathcal{A}^{\vec{s}}_M \vDash \varphi(r_1, \ldots, r_k, q, r'_1, \ldots, r'_m, e, e+1, \ldots, e+k+m-1, t) \leftrightarrow H(q, e+k, t) \wedge T(r_1, e, t) \wedge T(r_2, e+1, t) \wedge \cdots \wedge T(r_k, e+k-1, t) \wedge T(r'_1, e+k, t) \wedge T(r'_2, e+k+1, t) \wedge \cdots \wedge T(r'_m, e+k+m-1, t) \wedge \forall y(\neg y \approx e \wedge \neg y \approx e+1 \wedge \cdots \wedge \neg y \approx e+k+m-1 \rightarrow T(2, y, t))$. Como $M(q, r'_1) = (p, s, \triangleleft)$, segue-se que $(p, e+k-1, t+1) \in H^{\mathcal{A}^{\vec{s}}_M}$ e $(s, e+k, t+1) \in T^{\mathcal{A}^{\vec{s}}_M}$. Assim, $\mathcal{A}^{\vec{s}}_M \vDash H(p, e+k-1, t+1) \wedge T(r_1, e, t+1) \wedge T(r_2, e+1, t+1) \wedge \cdots \wedge T(r_k, e+k-1, t+1) \wedge T(s, e+k, t+1) \wedge T(r'_2, e+k+1, t+1) \wedge \cdots \wedge T(r'_m, e+k+m-1, t+1) \wedge \forall y(\neg y \approx e \wedge e+1 \wedge \cdots \wedge \neg y \approx e+k+m-1 \rightarrow T(2, y, t))$, o que acontece se, e apenas se, $C^{\vec{s}}_M(t+1) = ((r_1, \ldots, r_{k-1}), p, (r_k, s, r'_2, \ldots, r'_m))$ se, e somente se, a sentença $\varphi(r_1, \ldots, r_k, q, r'_1, \ldots, r'_m, e, e+1, \ldots, e+k+m-1, t)[q/p, r'_1/s, t/t+1]$, na qual substituímos simultaneamente q, r'_1, t por $p, s, t+1$, é verdadeira em $\mathcal{A}^{\vec{s}}_M$.

No segundo caso, $M(q, r'_1) = (p', s', \triangleright)$, também existem dois subcasos: ou $(r'_2, \ldots, r'_k) = \oslash$ ou $(r'_2, \ldots, r'_k) \neq \oslash$. Argumentando de forma similar ao parágrafo anterior, não é difícil constatarmos que a configuração $C^{\vec{s}}_M(t+1)$ também será correspondível em ambos subcasos. □

Observação 2.3.2. *A demonstração do teorema da correspodência usa indução interna em um modelo aritmético $\mathcal{M}$ previamente fixado por duas razões: (1) as computações de máquinas de Turing estão definidas internamente em $\mathcal{M}$; (2) o princípio de indução vale apenas internamente em $\mathcal{M}$, pois externamente $\mathcal{M}$ não é bem-ordenado - vale lembrar que a ordem de modelos aritméticos enumeráveis não-padrão é $\omega + (\omega^* + \omega) \cdot \eta$.*

Observação 2.3.3. *O teorema da correspondência mostra que uma fórmula ϕ da forma $H(x, y+k, z) \wedge T(v_1, y, z) \wedge T(v_2, y+1, z) \wedge \cdots \wedge T(v_k, y+k-1, z) \wedge T(v_{k+1}, y+k, z) \wedge T(v_{k+2}, y+k+1, z) \wedge \cdots \wedge T(v_{k+m}, y+k+m-1, z) \wedge \forall w(\neg w \approx y \wedge \neg w \approx$*

$y+1 \wedge \cdots \wedge \neg w \approx y+k+m-1 \rightarrow T(2,w,z))$ *sempre é equivalente a qualquer fórmula que corresponde à configuração* $C_M^{\vec{s}}(t) = ((r_1,\ldots,r_k),q,(r'_1,\ldots,r'_m))$ *quando substituímos* z *por* t*,* x *por* q *e assim por diante em* ϕ*. Por isso, podemos univocamente denominar tal fórmula* ϕ *de* a *fórmula correspondente à configuração* $C_M^{\vec{s}}(t)$*.*

Cabe observar que a demonstração do teorema da correspondência apresentada acima se trata de um argumento similar aquele usado por S. A. Cook [Cook(1971)] na demonstração do conhecido Teorema de Cook.

2.4 Teorias de Turing

Na seção anterior, associamos estruturas de primeira ordem às máquinas de Turing, agora associaremos teorias de primeira ordem. Antes de tudo, precisamos de um esquema de axioma que identifique os numerais com as constantes que denotam números naturais padrão, pois estamos usando implicitamente esssa convenção:

(A0) Dado um modelo aritmético $\mathcal{M}$, para cada número natural padrão $n \in \mathcal{M}$, $\overline{n} \approx \underline{n}$.

Também precisamos de um axioma que expresse as propriedades básicas da ordem dos estados, símbolos, celas e instantes, os quais foram identificados com os elementos de um modelo aritmético previamente fixado. O primeiro axioma nessa direção é o seguinte:

(A1) $\forall x \neg(x \prec x) \wedge \forall x \forall y \forall z(x \prec y \wedge y \prec z \rightarrow x \prec z) \wedge \forall y \forall z(x \prec y \vee x \approx y \vee y \prec x) \wedge \forall x(0 \prec x \vee 0 \approx x) \wedge \forall x \exists z \forall y(x \prec z \wedge (x \prec y \rightarrow z \prec y \vee z \approx y)))$[6]

Embora cada máquina de Turing tenha uma quantidade de estados e símbolos finita, a quantidade de estados pode variar de uma máquina para

[6] Note que tal z é único, devido à tricotomia.

outra, ou seja, ela é potencialmente infinita; o mesmo também vale para a quantidade de celas e instantes usados nas computações. Disso decorre a possibilidade de identificá-los com os números naturais. O axioma $A1$ expressa que os estados, símbolos, celas e instantes em máquinas de Turing estão linearmente ordenados, conforme uma ordem estrita e discreta com mínimo e sem máximo, tal como os números naturais.

Além do axioma $A1$, precisamos de um axioma que represente o comportamento das funções sucessor e predecessor (truncada) sobre a ordem estrita nos números naturais. Como dito acima, os estados, símbolos, celas e instantes estão representados pelos números naturais nas estruturas de Turing e, mais uma vez, isso explica a naturalidade do próximo axioma:

(A2) $\forall x \neg x + 1 \approx 0 \wedge \forall x \forall y (x + 1 \approx y + 1 \rightarrow x \approx y)) \wedge 0 - 1 \approx 0 \wedge \forall x ((x + 1) - 1 \approx x) \wedge \forall x \forall y (x \prec y + 1 \leftrightarrow x \prec y \vee x \approx y)$

Ainda no que se refere aos axiomas relativos aos estados, símbolos, celas e instantes enquanto representados por números naturais nas estruturas de Turing, assumiremos todas as instâncias do principio de indução para as fórmulas ϕ da linguagem de Turing $\mathcal{L}$:

(A3) $(\phi(0, \vec{x}) \wedge \forall y (\phi(y, \vec{x}) \rightarrow \phi(y + 1, \vec{x})) \rightarrow \forall z \phi(z, \vec{x})$

Como veremos logo abaixo, o esquema de axioma $A3$ nos permitirá obter um resultado de completude entre teorias de Turing e estruturas de Turing. Além desse resultado, no capítulo 4 veremos que existe uma evidência técnica, associada à não-expressabilidade dos números em linguagens de Turing, que mostra que o princípio de indução deve ser inserido em teorias de Turing.

Agora passamos aos axiomas que especificam características peculiares às máquinas de Turing. O primeiro desses axiomas expressa propriedades relacionadas ao fato de que máquinas de Turing são funções de estados e símbolos para estados, símbolos e movimentos, bem como as demais convenções subjacentes nas definições 2.2.1 e 2.2.2:

(A4) $\forall v\forall x\forall y\forall z((T(0,y,z) \vee T(1,y,z) \vee T(2,y,z)) \wedge (T(v,y,z) \wedge T(x,y,z) \rightarrow v \approx x)) \wedge \forall x\forall y\forall z(T(x,y,z) \rightarrow x \prec 2 \vee x \approx 2) \wedge \forall x\forall y\forall z\forall v\forall w((H(x,y,z) \wedge (\neg v \approx x \vee \neg w \approx y)) \rightarrow \neg H(v,w,z)) \wedge \forall v\forall x\forall y\forall z\forall w((H(n,y,z) \wedge T(v,x,z) \wedge z \prec w) \rightarrow (H(n,y,w) \wedge T(v,x,w))) \wedge \forall x\forall y\forall z(H(x,y,z) \rightarrow x \prec n \vee x \approx n)$

A primeira propriedade que $A4$ expressa é que cada cela de uma máquina de Turing tem um, e apenas um, símbolo em cada instante e eles podem ser apenas o símbolo zero, um ou vazio. A segunda é que uma máquina de Turing pode estar em único estado, em uma única cela, em cada instante e depois que ela entra no estado final nada mais acontence. Os próximos dois axiomas representam as instruções das máquinas de Turing:

(A5) $\forall y\forall z(H(q,y,z) \wedge T(s,y,z) \rightarrow H(p,y+1,z+1) \wedge T(r,y,z+1) \wedge \forall v\forall x(\neg v \approx y \rightarrow (T(x,y,z) \leftrightarrow T(x,v,z+1))))$ se $(q,s,p,r,\triangleright) \in M$.

(A6) $\forall y\forall z(H(q,y,z) \wedge T(s,y,z) \rightarrow H(p,y-1,z+1) \wedge T(r,y,z+1) \wedge \forall v\forall x(\neg v \approx y \rightarrow (T(x,y,z) \leftrightarrow T(x,v,z+1))))$ se $(q,s,p,r,\triangleleft) \in M$.

Note que $A5$ e $A6$ são em realidade *esquemas de axiomas*, i.e., eles são fórmulas que representam várias instâncias de axiomas, uma para cada tipo de instrução da máquina M. Além disso, cabe observar que $A5$ e $A6$ são apresentados somente na medida em que fixamos uma máquina de Turing M. A sentença $\forall v\forall x(\neg v \approx y \rightarrow (T(x,y,z) \leftrightarrow T(x,v,z+1)))$ em $A5$ e $A6$ expressa que as instruções agem sempre *localmente*, ou seja, uma instrução altera apenas uma única cela em cada passo da computação.

Como estamos interessados nas computações das máquinas de Turing, ainda precisamos de um axioma que especifique qual é a entrada da computação considerada. Digamos que seja a computação $C_M^{\vec{s}}$ da máquina M com $\vec{s} = s_1, \ldots, s_n$ definida no modelo aritmético $\mathcal{M}$. Nesse caso, o axioma abaixo representa a configuração inicial da computação em questão:

(A7) $H(0,0,0) \wedge T(2,0,0) \wedge T(2,1,0) \wedge T(s_1,1,0) \wedge \cdots \wedge T(s_n,n+1,0) \wedge T(2,n+2,0) \wedge T(2,n+3,0) \wedge \forall y(n+3 \prec y \rightarrow T(2,y,0))$, para $n \in \mathcal{M}$.

Observação 2.4.1. *Note que o axioma A7 é, internamente em $\mathcal{M}$, equivalente à fórmula correspondente à configuração inicial de $C_M^{s_1,\ldots,s_n}$, conforme estabelecido na observação 2.3.3. Mais do que isso, o axioma A7 está definido apenas internamente em $\mathcal{M}$, pois, quando $n \in \mathcal{M} - \mathcal{N}$, A7 é externamente uma fórmula infinita, o que significa que de fato A7 é uma fórmula interna em $\mathcal{M}$*[7].

Isso completa a descrição do modo como associamos teorias de primeira ordem às máquinas de Turing.

Definição 2.4.1. *Seja $\mathcal{M}$ um modelo aritmético e M uma máquina de Turing definida em $\mathcal{M}$. Uma* teoria de Turing *associada à M com entrada $s_1, \ldots, s_n$ é a $\mathcal{L}$-teoria de primeira ordem clássica $\mathcal{T}_M^{s_1,\ldots,s_n}$ especificada pelos axiomas A1, A2, A4 e A7 mais as instâncias dos esquemas de axiomas A0, A3, A5 e A6 descritos acima.*

Assim, dada uma máquina de Turing M e uma entrada $\vec{s}$, podemos associar uma estrutura $\mathcal{A}_M^{\vec{s}}$ e uma teoria de Turing $\mathcal{T}_M^{\vec{s}}$ à M internamente ao modelo aritmético $\mathcal{M}$ no qual M está definida. A fim de viabilizarmos uma computabilidade modelo-téorica em primeira ordem, cabe demonstrarmos a corretude de $\mathcal{T}_M^{\vec{s}}$ com relação à $\mathcal{A}_M^{\vec{s}}$.

Teorema 2.4.1 (Corretude)**.** *Seja M uma máquina de Turing. Então, para todas as sentenças φ de $\mathcal{L}$, $\mathcal{T}_M^{\vec{s}} \vdash \varphi$ implica $\mathcal{A}_M^{\vec{s}} \vDash \varphi$.*

Demonstração. Claramente, os axiomas $A0$, $A1$, $A2$ e $A3$ são verdadeiros na estrutura $\mathcal{A}_M^{\vec{s}}$ porque ela é uma extensão da estrutura dos números naturais com zero, sucessor, predecessor (truncada) e ordem estrita.

Quanto ao axioma $A4$, considere primeiramente a sentença $\forall x \forall y \forall z \forall v \forall w((H(x,y,z) \wedge (\neg v \approx x \vee \neg w \approx y)) \rightarrow \neg H(v,w,z))$. Suponha que $\mathcal{A}_M^{\vec{s}} \vDash H(q,e,t) \wedge (\neg p \approx q \vee \neg c \approx e)$ e $\mathcal{A}_M^{\vec{s}} \vDash H(p,c,t)$. Se $\mathcal{A}_M^{\vec{s}} \vDash \neg p \approx q$, então no passo t da computação $C_M^{\vec{s}}$, M está nos estados diferentes p e

[7] No capítulo 4 apresentaremos uma análise detalhada desse fenômeno.

q na cela e, lendo um símbolo, digamos s. Daí, M em $t-1$ deve ser da forma $M(w,s) = (q,r,*)$ e $M(w,s) = (p,r',*)$ com $* \in \{\triangleleft, \triangleright\}$. Nesse caso, mesmo que $r = r'$, M não seria função, porquanto $p \neq q$. Isso contraria a definição de M. Se $\mathcal{A}_M^{\vec{s}} \vDash p \approx q \wedge \neg c \approx e$, então M está nas celas e e c no estado $q = p$, lendo símbolos possivelmente diferentes, digamos s e r. Daí, M em $t-1$ deve ser da forma $M(w,s') = (q,s,*)$ e $M(w,r') = (q,r,*)$ com $* \in \{\triangleleft, \triangleright\}$, sendo que possivelmente $s' = r'$ e $s \neq r$, o que contradiz novamente que M é função. Portanto, derivamos um absurdo ao supormos que $\mathcal{A}_M^{\vec{s}} \vDash H(q,e,t) \wedge (\neg p \approx q \vee \neg c \approx e)$ e $\mathcal{A}_M^{\vec{s}} \vDash H(p,c,t)$. Com uma indução nos passos da computação $C_M^{\vec{s}}$, podemos mostrar que $\forall v \forall x \forall y \forall z((T(0,y,z) \vee T(1,y,z) \vee T(2,y,z)) \wedge (T(v,y,z) \wedge T(x,y,z) \rightarrow v \approx x))$ é verdadeira, pois ao supormos que fosse falsa derivamos uma contradição com o fato de que M é uma função, tal como no argumento anterior. Já as fórmulas $\forall x \forall y \forall z(T(x,y,z) \rightarrow x \prec 2 \vee x \approx 2)$ e $\forall x \forall y \forall z(H(x,y,z) \rightarrow x \prec n \vee x \approx n)$ são verdadeiras pela definição de máquinas de Turing enquanto funções finitas. Quanto à fórmula $\forall v \forall x \forall y \forall z \forall w((H(n,y,z) \wedge T(v,x,z) \wedge z \prec w) \rightarrow (H(n,y,w) \wedge T(v,x,w)))$, suponha que $\mathcal{A}_M^{\vec{s}} \vDash H(n,e,t) \wedge T(s,c,t) \wedge t \prec u$ e $\mathcal{A}_M^{\vec{s}} \nvDash H(n,e,u) \wedge T(s,c,u)$ para algum instante u. Se $\mathcal{A}_M^{\vec{s}} \vDash \neg H(n,e,u)$, então deve existir algum estado $p \neq n$ tal que $\mathcal{A}_M^{\vec{s}} \vDash H(p,e,u)$. Ora, isso significa que $M(n,s)$ está definida para algum símbolo s no instante t, o que contraria a convenção sobre a parada de uma máquina de Turing. Se $\mathcal{A}_M^{\vec{s}} \vDash \neg T(s,c,u)$, então deve existir algum símbolo $r \neq s$ tal que $\mathcal{A}_M^{\vec{s}} \vDash T(r,c,u)$. No entanto, isso somente seria possível se no instante t a máquina M aplicasse alguma instrução, mas por hipótese $\mathcal{A}_M^{\vec{s}} \vDash H(n,e,t)$. Portanto, todas as sentenças que compõem o axioma $A4$ são verdadeiras.

Quanto aos axiomas $A5$ e $A6$, basta considerar um deles, pois a veracidade do outro pode ser constatada de forma análoga. Suponhamos, então, que $\mathcal{A}_M^{\vec{s}} \vDash H(q,e,t) \wedge T(s,e,t)$, sendo que $(q,s,p,r,\triangleright) \in M$. Daí, segue-se que $(p, e+1, t+1) \in H^{\mathcal{A}_M^{\vec{s}}}$ e $(r,e,t+1) \in T^{\mathcal{A}_M^{\vec{s}}}$ pela definição de $\mathcal{A}_M^{\vec{s}}$, ou seja,

$\mathcal{A}_M^{\vec{s}} \vDash H(p, e+1, t+1) \wedge T(r, e, t+1)$. Seja c uma cela diferente de e. Suponha que $\mathcal{A}_M^{\vec{s}} \vDash T(s', c, t)$ mas $\mathcal{A}_M^{\vec{s}} \nvDash T(s', c, t+1)$. Então, M lê o símbolo s' na cela c no instante t, mas no passo seguinte M não lê mais o símbolo s' nessa cela c. Nesse caso, M deve ter uma instrução $(w, s', v, r', *)$ com $* \in \{\triangleleft, \triangleright\}$, sendo $s' \neq r'$. Considere $s' = s$, $w = q$, $v = p$ e $* = \triangleright$. Assim, $(p, c+1, t+1) \in H^{\mathcal{A}_M^{\vec{s}}}$ e $(p, e+1, t+1) \in H^{\mathcal{A}_M^{\vec{s}}}$ com $c+1 \neq e+1$. A partir disso, ao considerarmos a sentença $\forall x \forall y \forall z \forall v \forall w((H(x,y,z) \wedge (\neg v \approx x \vee \neg w \approx y)) \rightarrow \neg H(v,w,z))$ do axioma $A4$, derivamos uma contradição. Conversamente, se $\mathcal{A}_M^{\vec{s}} \vDash T(s', c, t+1)$ mas $\mathcal{A}_M^{\vec{s}} \nvDash T(s', c, t)$, derivamos uma contradição de modo similar. Isso mostra que o axioma $A5$ é verdadeiro.

Para finalizar, a veracidade do axioma $A7$ em $\mathcal{A}_M^{\vec{s}}$ foi constatada no teorema da correspondência. □

2.5 Completude

Turing [Turing(1936)] demonstrou que toda teoria capaz de representar as computações das máquinas de Turing também tem a capacidade de representar o problema da parada, o qual sabemos ser indecidível. O próximo resultado mostra que as teorias definidas em 2.4.1 têm essa capacidade.

Teorema 2.5.1 (Representação)**.** *Para todo t, se φ é a fórmula de $\mathcal{L}$ correspondente à configuração $C_M^{\vec{s}}(t)$ da máquina de Turing M, então $\mathcal{T}_M^{\vec{s}} \vdash \varphi$.*

Demonstração. A demonstração é por indução nos passos da computação $C_M^{s_1,\ldots,s_n}$ da máquina de Turing M. Para $t = 0$, $C_M^{\vec{s}}(0) = (\oslash, 1, (2, 2, s_1, \ldots, s_n, 2, 2))$ e, pelo axioma $A7$, sabemos que o resultado é verdadeiro.

Assuma que no passo t o resultado também é verdadeiro, e considere $C_M^{\vec{s}}(t) = ((r_1, \ldots, r_k), q, (r'_1, \ldots, r'_m))$. Ou M pára no passo $t+1$ ou M não pára. Primeiramente, consideraremos quando M não pára. Desse modo, no passo t, ou M aplica uma instrução da forma $(q, r'_1, p, s, \triangleright)$ ou $(q, r'_1, p', s', \triangleleft)$ (possivelmente $s = s'$ e $p = p' = q$). Se $(q, r'_1, p, s, \triangleright) \in M$, temos que considerar os

subcasos em que $r'_2, \ldots, r'_m = \oslash$ e $r'_2, \ldots, r'_m \neq \oslash$. Se $(q, r'_1, p', s', \triangleleft) \in M$, temos que analisar os subcasos em que $(r_1, \ldots, r_k) = \oslash$ $(r_1, \ldots, r_k) \neq \oslash$. Avaliaremos dois casos complicados, pois nos outros podemos argumentar de forma similar.

Suponha que $(q, r'_1, p, s, \triangleright) \in M$ e $r'_2, \ldots, r'_m = \oslash$. Então, $C_M^{\vec{s}}(t+1) = ((r_1, \ldots, r_k), q, r'_1)$ e $C_M^{\vec{s}}(t+1) = ((r_1, \ldots, r_k, s), p, 2)$. Por hipótese de indução, $\mathcal{T}_M^{\vec{s}} \vdash H(q, e+k, t) \wedge T(r_1, e, t) \wedge T(r_2, e+1, t) \wedge \cdots \wedge T(r_k, e+k-1, t) \wedge T(r'_1, e+k, t) \wedge \forall y(\neg y \approx e \wedge \neg y \approx e+1 \wedge \cdots \wedge \neg y \approx e+k \rightarrow T(2, y, t))$. Pelo axioma $A5$, $\mathcal{T}_M^{\vec{s}} \vdash H(q, e+k, t) \wedge T(r'_1, e+k, t) \rightarrow H(p, e+k+1, t+1) \wedge T(s, e+k, t+1)$ e, consequentemente, $\mathcal{T}_M^{\vec{s}} \vdash H(p, e+k+1, t+1) \wedge T(s, e+k, t+1)$. Aplicando a condição de localidade do axioma $A5$ na hipótese de indução, derivamos que $\mathcal{T}_M^{\vec{s}} \vdash T(r_1, e, t+1) \wedge T(r_2, e+1, t+1) \wedge \cdots \wedge T(r_k, e+k-1, t+1)$, e precisamos verificar se $\mathcal{T}_M^{\vec{s}}$ demonstra $T(2, e+k+1, t+1)$. Ora, a hipótese de indução também afirma que $\mathcal{T}_M^{\vec{s}} \vdash \forall y(\neg y \approx e \wedge \neg y \approx e+1 \wedge \cdots \wedge \neg y \approx e+k \rightarrow T(2, y, t))$. Obviamente $e+k \neq e+k+1$ e, com uma indução simples, podemos mostrar que $\mathcal{T}_M^{\vec{s}} \vdash \neg e+k \approx e+k+1$. Disso segue-se que $\mathcal{T}_M^{\vec{s}} \vdash T(2, e+k+1, t+1)$ e, por conseguinte, $\mathcal{T}_M^{\vec{s}} \vdash \forall y(\neg y \approx e \wedge \neg y \approx e+1 \wedge \cdots \wedge \neg y \approx e+k+1 \rightarrow T(2, y, t+1))$.

Agora suponha que $(q, r'_1, p', s', \triangleleft) \in M$ e $(r_1, \ldots, r_k) = \oslash$. Nesse caso, $C_M^{\vec{s}}(t+1) = (\oslash, p', (s', r'_2, \ldots, r'_m))$. Por hipótese de indução, $\mathcal{T}_M^{\vec{s}} \vdash H(q, e, t) \wedge T(r'_1, e, t) \wedge T(r'_2+1, e, t) \wedge \cdots \wedge T(r'_m, e+m-1, t) \wedge \forall y(\neg y \approx e \wedge \neg y \approx e+1 \wedge \cdots \wedge \neg y \approx e+m-1 \rightarrow T(2, y, t+1))$ com $e = 0$. Devido ao axioma $A6$, $\mathcal{T}_M^{\vec{s}} \vdash H(q, e, t) \wedge T(r'_1, e, t) \rightarrow H(p', e+1, t+1) \wedge T(s', e, t+1) \wedge \forall v \forall x(\neg v \approx e \rightarrow (T(x, e, t) \leftrightarrow T(x, v, t+1)))$. Portanto, usando um argumento similar ao anterior, derivamos que $\mathcal{T}_M^{\vec{s}} \vdash H(p', 0, t+1) \wedge T(s', 0, t+1) \wedge T(r'_2, 1, t+1) \wedge \cdots \wedge T(r'_m, m-1, t+1) \wedge \forall y(\neg y \approx 0 \wedge \cdots \wedge \neg y \approx m-1 \rightarrow T(2, y, t+1))$, ou seja, $\mathcal{T}_M^{\vec{s}}$ demonstra a fórmula que corresponde a configuração de $C_M^{\vec{s}}$ no passo $t+1$.

A demonstração para quando M não pára é análoga ao caso em que M pára, basta considerar o estado final. □

Corolário 2.5.1. *Seja M uma máquina de Turing e $\vec{s}$ uma entrada para M. Então, existe uma fórmula ϕ de $\mathcal{L}$ tal que $M(\vec{s}) \downarrow$ se, e somente se, $\mathcal{T}_M^{\vec{s}} \vdash \phi$.*

Demonstração. Considere a sentença $\exists y \exists z H(0, y, z)$ de $\mathcal{L}$. Se $\mathcal{T}_M^{\vec{s}} \vdash \exists y \exists z H(0, y, z)$, $\mathcal{A}_M^{\vec{s}} \vDash \exists y \exists z H(0, y, z)$ devido a corretude, e isso significa que M pára. Conversamente, suponhamos que M pára. Então $\mathcal{A}_M^{\vec{s}} \vDash \exists y \exists z H(0, y, z)$; digamos que $y = e$ e $z = t$. Pelo teorema correspondência 2.3.1, existe uma fórmula ψ correspondente a configuração final de M e, aplicando o teorema da representação 2.5.1, concluímos que $\mathcal{T}_M^{\vec{s}} \vdash \psi$, sendo ψ da forma $H(0, e+k, t) \wedge T(r_1, e, t) \wedge T(r_2, e+1, t) \wedge \cdots \wedge T(r_k, e+k-1, t) \wedge \forall y(\neg y \approx e \wedge \neg y \approx e+1 \wedge \cdots \wedge \neg y \approx e+k-1 \rightarrow T(0, y, t))$, onde $r_1, \ldots, r_k$ é o conteúdo da fita. Consequentemente, pela lógica de primeira ordem, $\mathcal{T}_M^{\vec{s}} \vdash \exists y \exists z H(0, y, z)$. □

O teorema da representação também implica que, para todas as sentenças ϕ de $\mathcal{L}$ correspondentes às configurações de $C_M^{\vec{s}}$, se $\mathcal{A}_M^{\vec{s}} \vDash \phi$ então $\mathcal{T}_M^{\vec{s}} \vdash \phi$. Dado que em computabilidade estamos interessados em investigar as computações de máquinas de Turing, representar as configurações das computações em $\mathcal{T}_M^{\vec{s}}$ já seria suficiente para desenvolvermos uma abordagem modelo-teórica em primeira ordem da computabilidade. Ao usarmos o método de ida-e-volta de Fraïssé [Fraïssé(1955)], mostraremos um resultado de completude ainda mais forte.

Definição 2.5.1. *Duas $\mathcal{L}$-estruturas $\mathcal{A}_M^{\vec{s}}$ e $\mathcal{B}_M^{\vec{s}}$ associadas a uma máquina de Turing M são* finitamente isomórficas *quando existir uma seqüência infinita $(I_n)_{n \in \omega}$ composta por isomorfismos parciais não-vazios na classe de todos os isomorfismos parciais entre $\mathcal{A}_M^{\vec{s}}$ e $\mathcal{B}_M^{\vec{s}}$, em símbolos $\mathbf{Part}(\mathcal{A}_M^{\vec{s}}, \mathcal{B}_M^{\vec{s}})$, com as seguintes propriedades:*

Propriedade da ida: Para todo $\xi \in I_{n+1}$ e $a \in \mathcal{A}_M^{\vec{s}}$, existe $\zeta_a \in I_n$ e $b \in \mathcal{B}_M^{\vec{s}}$ tal que $\xi \subseteq \zeta_a$ e $(a, b) \in \zeta_a$.

Propriedade da volta: Para todo $\xi \in I_{n+1}$ e $b \in \mathcal{B}_M^{\vec{s}}$, existe $\zeta_b \in I_n$ e $a \in \mathcal{A}_M^{\vec{s}}$ tal que $\xi \subseteq \zeta_b$ e $(a, b) \in \zeta_b$.

Lema 2.5.1. *Os modelos de uma teoria de Turing $\mathcal{T}_M^{\vec{s}}$ são finitamente isomórficos.*

Demonstração. Sejam $\mathcal{A}^{\vec{s}}_M$ e $\mathcal{B}^{\vec{s}}_M$ dois $\mathcal{L}$-modelos de $\mathcal{T}^{\vec{s}}_M$. Devido aos axiomas $A1$ e $A2$, para verificarmos que $\mathcal{A}^{\vec{s}}_M$ e $\mathcal{B}^{\vec{s}}_M$ são finitamente isomórficos, podemos aplicar uma variação da técnica usada na demonstração de que quaisquer dois modelos da teoria da função sucessor e ordem linear discreta com mínimo mas sem máximo são finitamente isomórficos[8].

Defina a função δ_n-*distância* $\delta_n : \mathcal{A}^{\vec{s}}_M \times \mathcal{A}^{\vec{s}}_M \to \mathbb{N} \cup \{\infty\}$ tal que

$$\delta_n(b,d) = \begin{cases} m & \text{se} \quad b + m = d \text{ e } m \leq 2^n \\ -m & \text{se} \quad d + m = b \text{ e } m \leq 2^n \\ \infty & \text{caso} \quad \text{contrário} \end{cases}$$

Seja a_0 o primeiro elemento de $\mathcal{A}^{\vec{s}}_M$ e b_0 o primeiro elmento de $\mathcal{B}^{\vec{s}}_M$. Considere a sequência $(I_n)_{n\in\omega}$ onde cada I_n é o conjunto de todos os isomorfismos parciais em $\xi \in Part(\mathcal{A}^{\vec{s}}_M, \mathcal{B}^{\vec{s}}_M)$ tais que:

1. O domínio de ξ, denotado por $dom(\xi)$, é finito;
2. $a_0 \in dom(\xi)$;
3. ξ *preserva* n-*distâncias*, i.e., para todo b e d em $dom(\xi)$, $\delta_n(b,d) = \delta_n(\xi(b), \xi(d))$.

Cada conjunto I_n não é vazio. Para verificar isso, considere $\xi = \{(a_0, b_0)\}$. Como a_0 é o menor elemento de $\mathcal{A}^{\vec{s}}_M$ e b_0 é o menor elemento de $\mathcal{B}^{\vec{s}}_M$, temos que $0^{\mathcal{A}^{\vec{s}}_M} = 0^{\mathcal{B}^{\vec{s}}_M} = a_0 = b_0$. Claramente, ξ é um isomorfismo parcial entre $\mathcal{A}^{\vec{s}}_M$ e $\mathcal{B}^{\vec{s}}_M$. Mais do que isso, $\delta_n(a_0, a_0) = \delta_n(\xi(a_0), \xi(a_0)) = \delta_n(b_0, b_0) = 0$ e, conseqüentemente, $\xi \in I_n$. Para obtermos o resultado almejado, mostraremos que $(I_n)_{n\in\omega}$ satisfaz a propriedade da ida, porquanto a propriedade da volta possa ser demonstrada de maneira similar.

Suponhamos que $\xi \in I_{n+1}$, e considere $a \in \mathcal{A}^{\vec{s}}_M$. Pela condição 1, $dom(\xi)$ é finito; digamos que $dom(\xi) = \{a_0, \ldots, a_r\}$ com $a_0 <^{\mathcal{A}} a_1 <^{\mathcal{A}} \cdots <^{\mathcal{A}} a_r$. Devido aos axiomas $A1$ e $A2$, ou $a = a_0$ ou $a_j <^{\mathcal{A}} a \leq^{\mathcal{A}} a_{j+1}$ para $0 \leq j < r - 1$ ou

[8] Cf. [Ebbinghaus et al.(2005)Ebbinghaus, Flum, and Thomas, p.184]

$a_r <^{\mathcal{A}} a$. Mais do que isso, ou existe $a' \in Dom(\rho)$ tal que $|\delta_n(a', a)| \leq 2^n$ ou não existe tal a'. Se existe esse a', devido aos axiomas $A1$ e $A2$, existe um único $b \in \mathcal{B}^{\vec{s}}_M$ que está relacionado com $\xi(a_0), \xi(a_1), \ldots, \xi(a_r)$ na ordem $<^{\mathcal{B}}$ da mesma forma que a está relacionado com $a_0, a_1, \ldots, a_r$ na ordem $<^{\mathcal{A}}$. Uma vez que $\rho \in I_{n+1}$, $\delta_{n+1}(a', a) = \delta_{n+1}(\rho(a'), \rho(a))$ e $|\delta_{n+1}(a', a)| \leq 2^{n+1}$. Assim, $\zeta_a = \rho \cup \{(a, b)\}$ também preserva as δ_n-distâncias e $|\delta_n(\xi(a'), b)| \leq 2^n$, ou seja, $\zeta_a \in I_n$. Se não existe esse a', então escolhemos um $b \in \mathcal{B}^{\vec{s}}_M$ que tal que $\delta_n(\xi(a'), b) = \infty$ para todo $a' \in dom(\xi)$ - esse b existe porque os modelos de $\mathcal{T}^{\vec{s}}_M$ são infinitos. Nesse caso, também constatamos que $\zeta_a = \xi \cup \{(a, b)\}$ preserva as δ_n-distâncias e, novamente, $\zeta_a \in I_n$.

Para finalizar, mostraremos que ζ_a é de fato um isomorfismo parcial entre $\mathcal{A}^{\vec{s}}_M$ e $\mathcal{B}^{\vec{s}}_M$. Em virtude de ζ_a ser uma bijeção que preserva δ_n-distâncias, ζ_a é tal que $\mathcal{A}^{\vec{s}}_M \vDash a_i \prec a_j$ se, e somente se, $\mathcal{B}^{\vec{s}}_M \vDash b_i \prec b_j$ para $b_i = \zeta_a(a_i)$ e $b_j = \zeta_a(a_j)$. Agora suponha que $\mathcal{A}^{\vec{s}}_M \vDash H(q, e, t)$ e $\mathcal{A}^{\vec{s}}_M \vDash T(s, c, u)$ (possivelmente $e = c$ e $t = u$). Pela definição de estrutura de Turing 2.3.2, isso significa que existem configurações $C^{\vec{s}}_M(t)$ e $C^{\vec{s}}_M(u)$ tais que M está no estado q na cela e no instante t e s é o símbolo na cela c no instante u. As configurações $C^{\vec{s}}_M(t)$ e $C^{\vec{s}}_M(u)$ são obtidas a partir da configuração inicial de M com entrada $\vec{s}$ através das instruções de M. Assim, pelo teorema representação $\mathcal{T}^{\vec{s}}_M \vdash H(q, e, t)$ e $\mathcal{T}^{\vec{s}}_M \vdash T(s, c, u)$. Como $\mathcal{B}^{\vec{s}}_M$ é um modelo de $\mathcal{T}^{\vec{s}}_M$, pelo teorema da corretude derivamos que $\mathcal{B}^{\vec{s}}_M \vDash H(q, e, t)$ e $\mathcal{B}^{\vec{s}}_M \vDash T(s, c, u)$. Reciprocamente, se $\mathcal{B}^{\vec{s}}_M \vDash H(q, e, t)$ e $\mathcal{B}^{\vec{s}}_M \vDash T(s, c, u)$, com um argumento similar concluímos que $\mathcal{A}^{\vec{s}}_M \vDash H(q, e, t)$ e $\mathcal{A}^{\vec{s}}_M \vDash T(s, c, u)$. No entanto, se $\mathcal{A}^{\vec{s}}_M \vDash \neg H(q, e, t)$ e $\mathcal{A}^{\vec{s}}_M \vDash \neg T(s, c, u)$, então é necessário uma análise adicional para mostrar que $\mathcal{B}^{\vec{s}}_M \vDash \neg H(q, e, t)$ e $\mathcal{B}^{\vec{s}}_M \vDash \neg T(s, c, u)$ porque nesse caso o fato de $(q, e, t) \notin H^{\mathcal{A}^{\vec{s}}_M}$ e $(s, c, u) \notin T^{\mathcal{A}^{\vec{s}}_M}$ não é uma consequência da aplicação das instruções de M e, assim, o teorema da representação nada garante sobre a demonstrabilidade de $\neg H(q, e, t)$ e $\neg T(s, c, u)$ em $\mathcal{T}^{\vec{s}}_M$. Suponha, então, que $\mathcal{A}^{\vec{s}}_M \vDash \neg H(q, e, t)$ e $\mathcal{A}^{\vec{s}}_M \vDash \neg T(s, c, u)$. Demonstraremos por indução nos passos da computação $C^{\vec{s}}_M$ que $\mathcal{B}^{\vec{s}}_M \vDash \neg H(q, e, t)$ e $\mathcal{B}^{\vec{s}}_M \vDash \neg T(s, c, u)$. Para simplicar a demonstração,

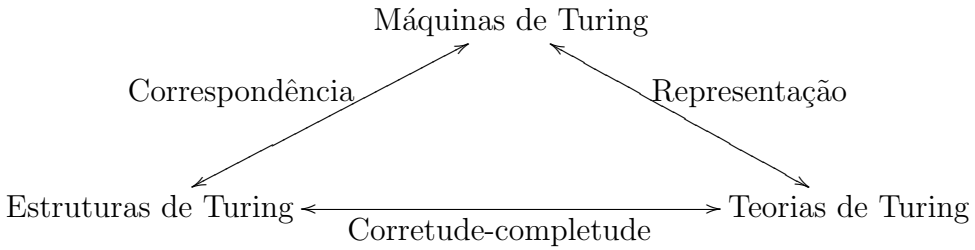
digamos que $t = u$, pois o caso em que $t \neq u$ exige considerações análogas. Suponha, como hipótese de indução, que, para todo $k < t$, $\mathcal{B}^{\vec{s}}_M \vDash \neg H(q, e, k)$ e $\mathcal{B}^{\vec{s}}_M \vDash \neg T(s, c, k)$. Digamos que, no instante t, M está no estado p e na cela e. Ou $M^{\vec{s}} \downarrow$ ou $M^{\vec{s}} \uparrow$ em t. Se $M^{\vec{s}} \downarrow$ no instante t e $p = 0$, então, pela hipótese de indução, podemos afirmar que $\zeta_a(q^{\mathcal{A}^{\vec{s}}_M}) = q^{\mathcal{B}^{\vec{s}}_M}$, $\zeta_a(s^{\mathcal{A}^{\vec{s}}_M}) = s^{\mathcal{B}^{\vec{s}}_M}$, $\zeta_a(e^{\mathcal{A}^{\vec{s}}_M}) = e^{\mathcal{B}^{\vec{s}}_M}$, $\zeta_a(c^{\mathcal{A}^{\vec{s}}_M}) = c^{\mathcal{B}^{\vec{s}}_M}$ e $\zeta_a(k^{\mathcal{A}^{\vec{s}}_M}) = k^{\mathcal{B}^{\vec{s}}_M}$ para todo $k < t$. Dado que ζ_a é uma função bijetora que preserva δ_n-distâncias, deve ser o caso que $\zeta_a(t^{\mathcal{A}^{\vec{s}}_M}) = t^{\mathcal{B}^{\vec{s}}_M}$ e, assim, $\mathcal{B}^{\vec{s}}_M \vDash \neg H(q, e, t)$ e $\mathcal{B}^{\vec{s}}_M \vDash \neg T(s, c, u)$, pois caso contrário $\mathcal{B}^{\vec{s}}_M$ não seria modelo de $\mathcal{T}^{\vec{s}}_M$. Se $M^{\vec{s}} \downarrow$ em t e de fato $p = n$, então as fórmulas $\forall v \forall x \forall y \forall z \forall w((H(n, y, z) \wedge T(v, x, z) \wedge z \prec w) \rightarrow (H(0, y, w) \wedge T(v, x, w)))$ e $\forall x \forall y \forall z(H(x, y, z) \rightarrow x \prec n \vee x \approx n)$ do axioma $A4$ afirmam que, para $w > t$, $C^{\vec{s}}_M(w) = C^{\vec{s}}_M(t)$, o que implica que $\mathcal{A}^{\vec{s}}_M$ e $\mathcal{B}^{\vec{s}}_M$ serão finitamente isomórficas em todos os instantes. Se $M^{\vec{s}} \downarrow$ em t e $p \neq n$, então as fórmulas $\forall x \forall y \forall z \forall v \forall w((H(x, y, z) \wedge (\neg v \approx x \vee \neg w \approx y)) \rightarrow \neg H(v, w, z))$ e $\forall x \forall y \forall z(H(x, y, z) \rightarrow x \prec n \vee x \approx n)$ do axioma $A4$ garantem que $\mathcal{T}^{\vec{s}}_M \vdash \neg H(q, e, t)$ e, por outro lado, temos a garantia de que $\mathcal{T}^{\vec{s}}_M \vdash \neg T(s, c, t)$ devido à condição de localidade dos axiomas $A5$ e $A6$ bem como à fórmula $\forall v \forall x \forall y \forall z((T(0, y, z) \vee T(1, y, z) \vee T(2, y, z)) \wedge (T(v, y, z) \wedge T(x, y, z) \rightarrow v \approx x)) \wedge \forall x \forall y \forall z(T(x, y, z) \rightarrow x \prec 2 \vee x \approx 2)$ do axioma $A4$. Dessa forma, também concluímos que $\mathcal{B}^{\vec{s}}_M \vDash \neg H(q, e, t)$ e $\mathcal{B}^{\vec{s}}_M \vDash \neg T(s, c, u)$ porque $\mathcal{B}^{\vec{s}}_M$ é modelo de $\mathcal{T}^{\vec{s}}_M$. Se $M^{\vec{s}} \uparrow$ no instante t mas $M^{\vec{s}} \downarrow$ em algum instante $w > t$, então o argumento é análogo ao caso em que $M^{\vec{s}} \downarrow$ em t e $p \neq n$. Se $M^{\vec{s}} \uparrow$ no instante t e, para todo instante $w > t$, $M^{\vec{s}} \downarrow$, então pode ser que $q = n$, ou seja, $\mathcal{A}^{\vec{s}}_M \vDash \neg H(n, e, t)$; o que é verdadeiro para todo t porquanto $M^{\vec{s}}$ nunca pare. Nesse caso, o axioma $A3$ assegura que $\mathcal{T}^{\vec{s}}_M \vDash \forall y \forall z \neg H(n, y, z)$ e, em particular, $\mathcal{T}^{\vec{s}}_M \vdash \neg H(n, e, t)$. Como $\mathcal{B}^{\vec{s}}_M$ é modelo de $\mathcal{T}^{\vec{s}}_M$, concluímos que $\mathcal{B}^{\vec{s}}_M \vDash \neg H(q, e, t)$ e, similarmente ao argumento acima, $\mathcal{B}^{\vec{s}}_M \vDash \neg T(s, c, u)$. □

Teorema 2.5.2 (Completude)**.** *Seja M uma máquina de Turing. Então, para todas as sentenças φ de $\mathcal{L}$, $\mathcal{A}^{\vec{s}}_M \vDash \varphi$ implica $\mathcal{T}^{\vec{s}}_M \vdash \varphi$.*

Demonstração. O teorema de Fraïssé [Fraïssé(1955)] afirma que, dadas duas

estruturas de primeira ordem com a mesma assinatura[9], se elas são finitamente isomórficas, então elas também são elementarmente equivalentes. Desse modo, quaisquer dois modelos de $\mathcal{T}_M^{\vec{s}}$ são elementarmente equivalentes devido ao lema . Ora, uma teoria de primeira ordem é completa se, e somente se, todos os seus modelos são elementarmente equivalentes. Portanto, $\mathcal{T}_M^{\vec{s}}$ é uma teoria completa, o que significa que se $\mathcal{T}_M^{\vec{s}} \nvdash \varphi$ então $\mathcal{T}_{\mathscr{M}} \vdash \neg\varphi$ e, pelo teorema da corretude, $\mathcal{A}_M^{\vec{s}} \vDash \neg\varphi$ para qualquer modelo $\mathcal{A}_M^{\vec{s}}$ de $\mathcal{T}_M^{\vec{s}}$. Logo, por contraposição, se $\mathcal{A}_M^{\vec{s}} \vDash \varphi$, então $\mathcal{T}_M^{\vec{s}} \vdash \varphi$. □

Segundo a perspectiva modelo-teórica da computabilidade que estamos propondo nesta tese, associamos estruturas e teorias a cada máquina de Turing em uma determinada classe de máquinas de Turing. Neste capítulo nos concentramos na classe das máquinas de Turing clássicas, apresentando uma abordagem modelo-teórica em lógica de primeira ordem. Como consequência dos resultados obtidos neste capítulo, temos o seguinte diagrama:



Cabe destacar que a infinitude das estruturas de Turing, por um lado, e o princípio de indução das teorias de Turing, por outro, foram cruciais para a formulação desse diagrama. Nessa direção, K. T. Kelly [Kelly(2004)] tem argumentado em favor de uma análise que vincula o problema epistêmico da indução à indecidibilidade do problema da parada. Os resultados deste

[9] De fato o teorema de Fraïssé aplica-se a estruturas relacionais. No entanto, é claro que podemos considerar as funções sucessor e predecessor (truncada) enquanto relações binárias. Isso não implicaria em grandes mudanças na axiomática das teorias de Turing.

capítulo, em especial o teorema da completude, mostram que realmente podemos pensar essa vinculação apontada por Kelly enquanto uma relação entre finitude e infinitude: uma vez que em teorias de Turing adicionamos o princípio de indução, elas são completas com relação às estruturas de Turing; em contrapartida, nas máquinas de Turing não dispomos desse princípio, e disso decorre a indecidibilidade do problema da parada.

Nos próximos capítulos mostraremos que uma abordagem modelo-teórica da computabilidade permite compreender aspectos ainda mais inusitados da análise original de Turing da computação, sobretudo, no que respeita à distinção entre objetos finitários e infinitários.

3. ANÁLISE MODELO-TEÓRICA DAS CARACTERÍSTICAS DA COMPUTABILIDADE CLÁSSICA

Die Grenzen meiner Sprache bedeuten die Grenzen meiner Welt.
Ludwig Wittgenstein

Este capítulo analisa as características da computabilidade clássica de um ponto de vista modelo-teórico, procurando mostrar que a lógica de primeira ordem é suficiente para os propósitos da computabilidade. Na seção 3.1 apresentamos os principais aspectos da computabilidade a serem tratados. Na seção 3.2 argumentamos que a estabilidade das máquinas de Turing está estritamente vinculada à sua simplicidade. Na seção 3.3 investigamos a absoluticidade do conceito de computação através de uma análise das máquinas de Turing não-deterministas. Na seção 3.4 indicamos alguns aspectos finitários e infinitários envolvidos na construção de máquinas de Turing universais. Na seção 3.5 mostramos que ao conceito de computação subjaz uma lógica minimal.

3.1 Aspectos da computabilidade

No capítulo 2 argumentamos que a análise de Turing do fenômeno da computação deve ser vista como uma análise fenomenológica. Essa perspectiva explica o caráter intuitivo do conceito de máquina de Turing, o qual se baseia em nossas intuições básicas acerca do que fazemos quando efetuamos computações. Qualquer pessoa, ao analisar tal fenômeno, chegará às mesmas conclusões que Turing, e disso decorre seu caráter intuitivo.

Após efetuar uma redução eidética do fenômeno da computação, Turing [Turing(1936), p.251] chegou a formulação das operações *simples* que uma máquina de Turing pode efetuar, a saber: imprimir e apagar símbolos e mover-se à direita ou à esquerda. Ao dizer que essas operações são simples, Turing queria indicar que elas não podem ser decompostas em outras operações sem perda do sentido do fenômeno da computação. Neste capítulo mostraremos que a simplicidade das operações que uma máquina de Turing pode perfazer está estritamente vinculada à estabilidade do conceito de computação; um dos aspectos centrais da computabilidade.

Segundo P. G. Odifreddi [Odifreddi(1992a), p.102], as operações computáveis são *estáveis* porque as diferentes maneiras de formalizá-la tornam-se extensionalmente equivalentes. Noutras palavras, sempre que tentamos formalizar o conceito intuitivo de *operação computável* acabamos por definir uma variante do conceito de máquina de Turing tal que a classe das funções computáveis fica inalterada. Por outro lado, se estabelecermos restrições substanciais sobre as características da fita ou sobre as ações capazes de serem efetuadas pelo cabeçote de uma máquina de Turing, esse modelo de computação deixa de computar todas as funções computáveis. Os diversos autômatos que estão em níveis inferiores na hierarquia de Chomsky das linguagens formais mostram justamente isso[1].

Ora, a análise de Turing mostra que não apenas os átomos de um computor são sua fita e cabeçote, mas também que a infinitude do espaço, representado pelo conceito de fita, e a infinitude do tempo para efetuar ações, representada pelo conceito de cabeçote, não podem ser reduzidas sem perdas na classe das funções computáveis. A partir dessa intuição, demonstraremos um resultado que relaciona a estabilidade das máquinas de Turing e a simplicidade de suas operações.

Outro aspecto central da computabilidade fora indicado por K. Gödel [Gödel(1986c)]. Como dissemos no capítulo 2, Gödel [Gödel(1986d)]

[1] Cf. [Sipser(2006)]

mostrou-se cético com relação a veracidade da tese de Church. Ainda assim, ele acreditava que o conceito de computabilidade é absoluto:

> Pode ser demonstrado, ademais, que se uma função é computável em algum dos sistemas S_i, ou mesmo em sistemas de ordem transfinita, ela já era computável em S_1[2]. Assim a noção de "computável" é em certo sentido "absoluta", enquanto que a maioria das outras noções metamatemáticas conhecidas (por exemplo, demonstrável, definível, e assim por diante) essencialmente dependem do sistema adotado. [Gödel(1986c), p.399]

Na esteira de Gödel, P. G. Odifreddi [Odifreddi(1992a), p.102] afirma que a noção de operação computável é *absoluta*, uma vez que ela não depende da capacidade expressiva da linguagem usada para defini-la. Porém, como argumentado por W. Sieg [Sieg(2006)], não existe uma demonstração da absolutidade da computabilidade que não seja equivalente de algum modo à tese de Church-Turing. Neste capítulo também mostraremos que, de um ponto de vista modelo-teórico, tanto a estabilidade quanto a absoluticidade podem ser compreendidas como conseqüências de dois princípios: (1) a lógica de primeira ordem é suficiente para descrever as computações de uma máquina de Turing e (2) as possíveis restrições ou ampliações finitárias das ações de uma máquina de Turing podem ser formalizadas em linguagem de primeira ordem.

Uma perspectiva semelhante a essa que sustentaremos já fora proposta por S. Kripke como tentativa de demonstrar a tese de Church-Turing a partir da "tese de Hilbert" [Kripke(2000)]. A *tese de Hilbert* consiste na afirmação de que qualquer argumento matemático pode ser formalizado em alguma linguagem de primeira ordem. Essa tese implica, em particular, que argumentos sobre os efeitos de aplicar instruções de um algoritmo podem ser formalizados em alguma linguagem de primeira ordem.

[2] O sistema S_1 é a aritmética clássica de primeira ordem enquanto que, para $i > 1$, S_i é o sistema aritmético de ordem i.

A fim de explicitar nossa posição, mostraremos que duas variações comuns nos livros de teoria da computação podem ser descritas em linguagem de primeira ordem: máquinas de Turing com múltiplas fitas e máquinas de Turing não-deterministas. Ao fazermos isso, não estamos, todavia, argumentando em favor de uma demonstração da tese de Church-Turing a partir da tese de Hilbert. Nosso objetivo é somente mostrar que alguns exemplos concretos podem ser tratados via linguagem de primeira ordem. De um ponto de vista modelo-teórico, a tese de Hilbert é ainda mais questionável do que a tese de Church-Turing, uma vez que a possibilidade de formalizarmos todos os algoritmos em linguagens de primeira ordem é um problema em aberto cuja solução talvez seja mais difícil do que a própria tese de Church-Turing.

Nesse sentido, também apresentaremos neste capítulo uma análise da *universalidade* e *logicidade* na computabilidade de Turing. Os resultados que serão apresentados sobre a universalidade e logicidade da computabilidade aparentemente somente são possíveis de serem vislumbrados a partir de uma perspectiva modelo-teórica. Ademais, a totalidade dos resultados deste capítulo indica que a lógica de primeira ordem é suficiente para o desenvolvimento de tal abordagem da computabilidade.

3.2 Estabilidade

A fim de tornar a exposição concreta, nos concentraremos na estabilidade do conceito de memória. Em teoria da computação, *memória* designa o estado informacional do sistema de computação [Lewis and Papadimitriou(1998), p.210]. A memória é composta por *arquivos*, os quais são conjuntos de símbolos. Tomando como parâmetro o *acesso* que o computador tem aos seus arquivos, costuma-se dizer que existem basicamente dois tipos de memória: seqüencial e aleatória. Na memória *seqüencial* o acesso aos arquivos é feito através de sequências de passos, já na memória *aleatória* o acesso a qualquer arquivo pode ser feito através de um único passo. Em máquinas de Turing

a memória está representada pelo conteúdo da fita e seu acesso é seqüencial porque para o cabeçote se dirigir, por exemplo, à cela 10, estando na cela 5, ele precisa efetuar uma seqüência de passos.

Em [Minsky(1967)], M. L. Minsky apresentou um modelo de computação com memória aleatória, chamado *máquina de registros*. No mesmo trabalho, Minsky também demonstrou que as máquinas de Turing são capazes de simular todas as ações das máquinas de registro e vice-versa, ou seja, a estabilidade do conceito de computabilidade foi verificada. Não é difícil verificar a equivalência entre máquinas de Turing e máquinas de registros, embora elas sejam modelos de computação bastante diferentes. As demonstrações usuais não explicitam em que pontos a memória aleatória difere da memória seqüencial e porque elas não interferem na estabilidade da computabilidade, porém com uma abordagem modelo-téorica seremos capazes de avaliar isso.

Definição 3.2.1. *Seja $\mathcal{M}$ um modelo aritmético. Uma* máquina de registros *é uma função parcial finita não-vazia R tal que, para alguns números naturais n e m em $dom(\mathcal{M})$,*

$$dom(R) \subseteq \{1, 2, \ldots, n\} \times \{0, 1, \ldots, m\},$$

$$cod(R) \subseteq \{\oplus, \ominus\} \times \{0, 1, \ldots, n\},$$

Onde:

1. *Os elementos do primeiro fator de $dom(R)$ e último de $cod(R)$ são os* contadores *de R, sendo 1 o* contador inicial *e 0 o* contador final*;*
2. *Os elementos do segundo fator de $dom(R)$ são os* registros *de R;*
3. *Os elementos do primeiro fator de $cod(R)$ são as* operações *de R, sendo $\oplus : \mathbb{N} \to \mathbb{N}$ tal que $\oplus(n) = n+1$ chamada de* incrementação *e $\ominus : \mathbb{N} \to \mathbb{N}$ tal que $\ominus(n) = n - 1$ chamada de* decrementação *(truncada).*

Os elementos de R *são chamados* instruções.

Intuitivamente, uma instrução da forma $(i, n, \oplus, j)$ significa que, estando no contador i, a máquina incrementa o registro n e depois vai para o contador j. Já uma instrução da forma $(i, n, \ominus, j)$ significa que, estando no contador i, a máquina decrementa o registro n e depois vai para o contador j. Se o conteúdo do registro n é vazio, então $(i, n, \ominus, j)$ apenas muda o contador de i para j.

Definição 3.2.2. *Seja* $\mathcal{M}$ *um modelo aritmético e* R *uma máquina de registros com contadores em* $\{0, 1, \ldots, n\}$ *e registros em* $\{0, 1, \ldots, m\}$. *Uma* configuração *é um elemento em* $\{0, 1, \ldots, n\} \times \mathcal{M}^{k+1}$ *com* $k \geq m$, *sendo* $(1, 0, r_1, \ldots, r_k)$ *a* configuração inicial *e* $(0, s, r'_1, \ldots, r'_k)$ *a* configuração final, *onde* $0, r_1, \ldots, r_k$ *é chamado de* conteúdo inicial *dos registros* $0, 1, \ldots, k$ *e* $s, r'_1, \ldots, r'_m$ *é o* conteúdo final. *Uma* computação $C_R^{x_1,\ldots,x_k}$ *com entrada* $x_1, \ldots, x_k$ *é um seqüência total ou parcial* $C_R^{x_1,\ldots,x_k} : \mathcal{M} \to \{0, 1, \ldots, n\} \times \mathcal{M}^{k+1}$ *definida do seguinte modo:*

1. $C_R^{x_1,\ldots,x_k}(0)$ *é a configuração inicial* $(1, 0, x_1, \ldots, x_k)$.

2. *Se* $C_R^{x_1,\ldots,x_k}(t) = (i, r_0, r_1, \ldots, r_u)$, *então* $C_R^{x_1,\ldots,x_k}(t+1)$ *está definida assim:*

   - $C_R^{x_1,\ldots,x_k}(t+1) = (j, r'_0, r'_1, \ldots, r'_u)$ *quando* $R(i, a) = (\oplus, j)$ *com* $r'_l = r_p + 1$ *e* $l = p$ *mas* $r'_l = r_p$ *para todo* $l \neq p$ *(*$l, p \in \{0, 1, \ldots, k\}$*).*
   - $C_R^{x_1,\ldots,x_k}(t+1) = (j, r'_0, r'_1, \ldots, r'_u)$ *quando* $R(i, a) = (\ominus, j)$ *com* $r'_l = r_p - 1$ *e* $l = p$ *mas* $r'_l = r_p$ *para todo* $l \neq p$ *(*$l, p \in \{0, 1, \ldots, k\}$*).*

Tal como nas máquinas de Turing, nas máquinas de registros uma computação é uma mudança de configurações, efetuada conforme as instruções. A diferença crucial é que qualquer registro pode ser acessado e alterado através de um passo. Isso caracteriza o acesso aleatório à memória, que nesse caso é o conteúdo dos registros. Adotaremos as mesmas convenções do capítulo 2 com

relação às máquinas de registros, mas ao invés de usarmos uma representação binária dos números usaremos o próprio número nos registros[3].

Definição 3.2.3. *Uma função parcial ou total* $f : \mathbb{N}^k \to \mathbb{N}$ *é* computável por uma máquina de registros *se, e somente se, existe uma máquina de registros* R *tal que:*

- $f(x_1, \ldots, x_k) = y$ *se, e somente se,* $R[x_1, \ldots, x_k] = y$.
- $f(x_1, \ldots, x_k)$ *não está definida se, e somente se,* $R[x_1, \ldots, x_k] \uparrow$.

Agora podemos associar estruturas de primeira ordem às máquinas de registros. Para tanto, defimos que a *linguagem de resgistros* é a linguagem de primeira ordem $\mathcal{L}^r$ com assinatura $\mathcal{S}^r = \{0, +1, -1, \prec, C, R\}$, onde C é o símbolo relacional binário de *contador* e R é o símbolo relacional ternário de *registro*.

Definição 3.2.4. *Uma* estrutura de registro *associada à máquina de registros* R *com entrada* $\vec{x} = x_1, \ldots, x_k$ *é uma* $\mathcal{S}^r$*-estrutura de primeira ordem* $\mathcal{A}_R^{\vec{x}} = (\mathbb{N}, 0^{\mathcal{A}_R^{\vec{s}}}, +1^{\mathcal{A}_R^{\vec{s}}}, -1^{\mathcal{A}_R^{\vec{s}}}, \prec^{\mathcal{A}_R^{\vec{s}}}, C^{\mathcal{A}_R^{\vec{s}}}, R^{\mathcal{A}_R^{\vec{s}}})$ *igual a uma estrutura de Turing, exceto pelo fato de que* $C^{\mathcal{A}_R^{\vec{s}}}$ *é uma relação binária definida em* $\mathbb{N}^2$ *por indução no segundo fator e* $R^{\mathcal{A}_R^{\vec{s}}}$ *é uma relação ternária definida em* $\mathbb{N}^3$ *por indução no terceiro fator do seguinte modo:*

- $(1, 0) \in C^{\mathcal{A}_R^{\vec{s}}}$ *e* $(1, x_1, 0) \in R^{\mathcal{A}_R^{\vec{s}}}, \ldots, (k, x_k, 0) \in R^{\mathcal{A}_R^{\vec{s}}}$ *e, para todo* y *tal que* $y = 0$ *ou* $y > k$, $(y, 0, 0) \in R^{\mathcal{A}_R^{\vec{s}}}$;
- *Se* $(i, t) \in C^{\mathcal{A}_R^{\vec{s}}}$, $(r, n, t) \in R^{\mathcal{A}_R^{\vec{s}}}$ *e* $R(i, r) = (\oplus, j)$, *então* $(j, t + 1) \in C^{\mathcal{A}_R^{\vec{s}}}$, $(r, n + 1, t + 1) \in R^{\mathcal{A}_R^{\vec{s}}}$ *e* $(s, m, t + 1) \in R^{\mathcal{A}_R^{\vec{s}}}$, *para todo* s *tal que* $s \neq r$ *e* $(s, m, t) \in R^{\mathcal{A}_R^{\vec{s}}}$;
- *Se* $(i, t) \in C^{\mathcal{A}_R^{\vec{s}}}$, $(r, n, t) \in R^{\mathcal{A}_R^{\vec{s}}}$ *e* $R(i, r) = (\ominus, j)$, *então* $(j, t + 1) \in C^{\mathcal{A}_R^{\vec{s}}}$, $(r, n - 1, t + 1) \in R^{\mathcal{A}_R^{\vec{s}}}$ *e* $(s, m, t + 1) \in R^{\mathcal{A}_R^{\vec{s}}}$, *para todo* s *tal que* $s \neq r$ *e* $(s, m, t) \in R^{\mathcal{A}_R^{\vec{s}}}$.

[3] Obviamente, em estudo detalhado das máquinas de registro seria prudente adotar algum sistema de representação numérica.

Não convém definirmos explicitamente teorias de registros, porque basta formularmos axiomas análogos aos axiomas de uma teoria de Turing, considerando as diferenças nas linguagens desses modelos de computação. Note que, todavia, existem duas diferenças cruciais entre máquinas de registros e máquinas de Turing: (1) registros permitem armazenar um número natural qualquer ao passo que celas não e (2) os movimentos em uma máquina de registros estão associados a mudança de contador e não ao deslocamento de um cabeçote. Esses são os traços distintivos entre memória seqüencial e memória aleatória. O fato de que as diferenças entre esses conceitos de memória não interfiram na classe das funções computáveis mostra que o modo como o espaço é utilizado em uma computação não interfere no conceito de computabilidade.

A possibilidade de ambos os modelos de computação serem descritos como extensões da estrutura da função sucessor com ordem estrita indica que, conquanto diferentes, tais estruturas tem uma parte comum. A estabilidade na mudança de um modelo de computação com memória seqüencial para um modelo com memória aleatória pode ser vista como uma conseqüência de dois fatores: (1) a lógica de primeira ordem é suficiente para descrever as estruturas associadas a esses modelos de computação e (2) ambos os modelos tem restrições finitárias. Mais do que isso, note que os conceitos de contador e registro parecem ser análogos aos conceitos de cabeçote e fita. Os próximos resultados mostram que essa semelhança não é casual, ela repousa sobre traços irredutíveis das próprias máquinas de Turing.

Lema 3.2.1. *Não existem estruturas de Turing geradas por assinaturas com símbolos relacionais unários para cada um dos gêneros de uma máquina de Turing (estado, símbolo, espaço e tempo).*

Demonstração. Suponha, a fim de obter uma contradição, que exista uma estrutura de Turing $\mathcal{A}_M^{\vec{s}}$ com uma assinatura $\mathcal{S}' = \{\underline{n}\}_{n\in\mathcal{M}} \cup \{+1, -1, \prec, H, T, E, I\}$ onde H, T, E, I são símbolos relacionais unários para estado, símbolo, espaço e tempo. Considere a máquina de Turing M =

$\{(0,2,0,2,\triangleright),(0,1,1,1,\triangleright),(0,1,1,0,\triangleright)\}$. Assuma que $\mathcal{A}^{\vec{s}}_M$ é a $\mathcal{S}'$-estrutura associada à M. A partir disso, devemos ter algo do tipo $\mathcal{A}^{\vec{s}}_M \vDash \forall x\forall y(H(0) \wedge T(2) \wedge E(x) \wedge I(y)) \rightarrow (H(0) \wedge T(2) \wedge E(x+1) \wedge I(y+1) \wedge \phi)$, $\mathcal{A}^{\vec{s}}_M \vDash \forall x\forall y(H(0) \wedge T(1) \wedge E(x) \wedge I(y)) \rightarrow (H(1) \wedge T(1) \wedge E(x+1) \wedge I(y+1) \wedge \phi)$ e $\mathcal{A}^{\vec{s}}_M \vDash \forall x\forall y(H(0) \wedge T(1) \wedge E(x) \wedge I(y)) \rightarrow (H(1) \wedge T(0) \wedge E(x+1) \wedge I(y+1) \wedge \phi)$, onde ϕ expressa a condição de localidade na linguagem gerada sobre $\mathcal{S}'$. Considere a computação C^1_M de $\mathscr{M}$. Assim, a partir da configuração inicial de M, devemos derivar algo como $\mathcal{A}^{\vec{s}}_M \vDash H(0) \wedge E(0) \wedge I(0)$. Daí, aplicando a primeira instrução duas vezes e depois a segunda instrução uma vez, derivamos que $\mathcal{A}^{\vec{s}}_M \vDash H(1) \wedge E(3) \wedge I(3)$. Como já tinhamos que $\mathcal{A}^{\vec{s}}_M \vDash H(0) \wedge E(0) \wedge I(0)$, agora também temos $\mathcal{A}^{\vec{s}}_M \vDash H(1) \wedge E(0) \wedge I(0)$. Logo, $0 = 1$ pela unicidade dos estados de M com relação ao espaço-tempo; o que é um absurdo. □

Lema 3.2.2. *Não existem estruturas de Turing geradas por assinaturas com símbolos relacionais com exatamente um parâmetro para cada gênero de uma máquina de Turing.*

Demonstração. Suponha, a fim de obter uma contradição, que exista uma estrutura de Turing $\mathcal{A}^{\vec{s}}_M$ com uma assinatura $\mathcal{S}'' = \{\underline{n}\}_{n\in\mathcal{M}} \cup \{+1,-1,\prec,A\}$, onde A é um símbolo relacional quaternário com parâmetros para estado, símbolo, espaço e tempo. Novamente, considere a máquina de Turing $M = \{(0,2,0,2,\triangleright),(0,1,1,1,\triangleright),(0,1,1,0,\triangleright)\}$. Assuma que $\mathcal{A}^{\vec{s}}_M$ é a $\mathcal{S}''$-estrutura associada à M. Assim, devemos ter algo do tipo $\mathcal{A}^{\vec{s}}_M \vDash \forall x\forall y(A(0,2,x,y) \rightarrow A(0,2,x+1,y+1) \wedge \phi)$, $\mathcal{A}^{\vec{s}}_M \vDash \forall x\forall y(A(0,1,x,y) \rightarrow A(1,1,x+1,y+1) \wedge \phi)$ e $\mathcal{A}^{\vec{s}}_M \vDash \forall x\forall y(A(0,1,x,y) \rightarrow A(1,0,x+1,y+1) \wedge \phi)$ onde ϕ expressa a condição de localidade na linguagem gerada sobre $\mathcal{S}''$. Dessa vez, a computação C^{11}_M de $\mathscr{M}$. Nesse caso, a partir da configuração inicial de M, derivamos $\mathcal{A}^{\vec{s}}_M \vDash A(0,1,2,0) \wedge A(0,1,3,0)$ - lembre que a entrada de C^{11}_M é a sequência de símbolos 221122. Daí, aplicando a primeira instrução duas vez e depois a terçeira instrução uma vez, temos $\mathcal{A}^{\vec{s}}_M \vDash A(1,0,3,3)$, mas, pela localidade das ações de M, também deve ser o caso $\mathcal{A}^{\vec{s}}_M \vDash A(1,1,3,3)$. Logo, $0 = 1$, pela unicidade dos símbolos de M com relação ao espaço-tempo. □

Teorema 3.2.1. *Seja M uma máquina de Turing e $\mathcal{L}'''$ uma linguagem gerada por uma assinatura $\mathcal{S}'''$ possivelmente diferente da assinatura de Turing $\mathcal{S}$. Então, vale o teorema da correspondência para $\mathcal{L}'''$ se, e somente se, $\mathcal{S}'''$ tem símbolos relacionais que permitem construir fórmulas equivalentes às fórmulas de $\mathcal{S}$ que correspondem às configurações de M.*

Demonstração. Seja $\mathcal{A}_M^{\vec{s}}$ uma estrutura de Turing com assinatura $\mathcal{S}'''$. Pelo lema 3.2.1, $\mathcal{S}'''$ não pode ter símbolos unários para cada um dos gêneros de M. Assim, $\mathcal{S}'''$ deve ter pelo menos símbolos relacionais de aridade maior ou igual a 2. Obviamente, $\mathcal{S}'''$ deve ter símbolos relacionais com parâmetros para os estados e símbolos, pois caso contrário não representaria as computações de M; digamos que $\mathcal{S}'''$ tenha o símbolo relacional $B(q, s, \tau_1, \ldots, \tau_n)$, sendo $\tau_1, \ldots, \tau_n$ outros possíveis parâmetros de B. Alguns dos parâmetros $\tau_1, \ldots, \tau_n$ devem ser parâmetros para celas, senão seria impossível determinar qual é a posição de M em alguma computação, digamos que seja τ_1. Digamos que $M = \{(0, 2, 0, 2, \triangleright), (0, 1, 1, 1, \triangleright), (0, 1, 1, 0, \triangleright)\}$, e consideremos a computação C_M^{101} de $\mathscr{M}$. Assim, a partir da configuração inicial de M, derivamos $\mathcal{A}_M^{\vec{s}} \vDash B(0, 1, 2, \tau_2, \ldots, \tau_n) \wedge B(x, 0, 3, \tau_2, \ldots, \tau_n) \wedge B(x, 1, 4, \tau_2, \ldots, \tau_n)$ - a entrada tem dois símbolos vazios antes e depois dos símbolos de entrada. Note que devemos ter $x \neq 0$ porque se $x = 0$ então M estaria no mesmo estado inicial, lendo dois símbolos diferentes, o que não é possível em uma máquina de Turing. Todavia, se $x \neq 1$, então M estaria em dois estados no instante inicial, o que também não é possível. Dessa forma, alguns dos termos $\tau_2, \ldots, \tau_n$ devem ser parâmetros para o tempo, ou seja, devemos ter algo do tipo $B(q, s, e, t, \tau_3, \ldots, \tau_n)$. No entanto, isso contraria o lema 3.2.2, e adicionar mais parâmetros para instantes seria reduntante e não alteraria a situação. Se tivermos $B(q, s, e_q, e_s, t, \tau_5, \ldots, \tau_n)$ onde e_q situa q e e_s situa s, então teríamos $\mathcal{A}_M^{\vec{s}} \vDash B(0, 1, 0, 2, 0, \tau_4, \ldots, \tau_n) \wedge B(0, 0, 0, 3, 0, \tau_4, \ldots, \tau_n) \wedge B(0, 1, 0, 4, 0, \tau_4, \ldots, \tau_n)$ e não haveria contradições. Ora, $B(q, s, e_q, e_s, t, \tau_4, \ldots, \tau_n)$ é equivalente à $H(q, e_q, t) \wedge T(s, e_s, t)$. Claramente, podemos generalizar o argumento para qualquer máquina de Turing.

Logo, se $B(q, s, e_q, e_s, t, \tau_5, \ldots, \tau_n)$ está em $\mathcal{S}''$, o teorema da correspondência será válido. □

O teorema 3.2.1 mostra que a dicotomia entre cabeçote e fita em uma máquina de Turing não pode ser reduzida. Na fita estão os símbolos, e no cabeçote estão os estados. Noutras palavras, podemos interpretar o teorema 3.2.1 como expressando que os componentes *fita* e *cabeçote* são objetos simples, no sentido de que não podem ser decompostos em outros objetos, que compõem uma máquina de Turing. A distinção das celas na fita e estados no cabeçote são distinções componentes dos conceitos de fita e cabeçote, mas apenas em sentido explanatório e não ontológico. A fita é composta por celas apenas no sentido de que explicamos o que é uma fita dizendo isso, mas de fato as celas existem na fita, o mesmo ocorrendo com os estados relativamente ao cabeçote. Os conceitos de *contador* e *registro* são, portanto, apenas variações dos conceitos originais de cabeçote e fita, desenvolvidos para permitir o acesso aleatório à memória.

Dizer que o conceito de máquina de Turing é *estável* significa, portanto, pelo menos duas coisas: (1) ontologicamente, significa que os seus componentes, fita e cabeçote, não podem ser decompostos; (2) matematicamente, significa que qualquer restrição sobre esses componentes implica que o modelo de computação não computará todas as funções computáveis.

3.3 Absoluticidade

Começamos a análise da absoluticidade através de uma variação da noção de máquina de Turing bastante conhecida: máquinas de Turing com múltiplas fitas[4]. Esse tipo de máquina é composta por um número fixo e finito de fitas, cada uma tem um cabeçote. Os cabeçotes das fitas são controlados por um processador central. Assim, em um único passo a máquina é capaz de ler os símbolos examinados por todos os seus cabeçotes e, dependendo dos

[4] Cf. [Sipser(2006)] e [Lewis and Papadimitriou(1998)].

símbolos nas fitas e o estado corrente de seu processador central, escrever um símbolo nos locais examinados e mover os seus cabeçotes à direita ou à esquerda. Uma definição formal desse modelo pode ser a seguinte.

Definição 3.3.1. *Seja $\mathcal{M}$ um modelo aritmético. Uma* máquina de Turing com k-fitas *é uma função parcial M^k tal que $dom(M^k) \subseteq \{1,\ldots,n\} \times \{0,1,2\}^k$ e $cod(M^k) \subseteq \{0,\ldots,n\} \times \{0,1,2\}^k \times \{\triangleleft,\triangleright\}^k$ para $n,k \in \mathcal{M}$. Uma* computação *de M^k com entrada $\vec{s} = s_1,\ldots,s_m$ é uma função parcial ou total $C^{\vec{s}}_{M^k} : \mathcal{M} \to (\{0,1,2\}^\star)^k \times \{0,\ldots,n\} \times (\{0,1,2\}^\star)^k$ definida recursivamente tal como na definição para máquinas de Turing exceto pelas seguintes modificações*[5]*:*

1. $C^{\vec{s}}_{M^k}(0) = ((\oslash,1,(2,2,s_1,\ldots,s_m,2,2))_1,(\oslash,1,0)_2,\ldots,(\oslash,1,0)_k)$;
2. *Se* $C^{\vec{s}}_{M^k}(i) = (((a_1,\ldots,a_l),q,(b_1,\ldots,b_m))_1,\ldots,((c_1,\ldots,c_u),q,(d_1,\ldots,d_w)))_k)$, *então a configuração instantânea de M^k no passo $i+1$ é definida componente à componente tal como na definição de uma computação de uma máquina de Turing;*
3. *A configuração final de M^k é definida conforme à definição para máquinas de Turing exceto pelo fato de que se* $C^{\vec{s}}_{M^k}(i) = (((a_1,\ldots,a_l),0,(b_1,\ldots,b_m))_1,\ldots,((c_1,\ldots,c_u),0,(d_1,\ldots,d_w))$, *então a saída de M^k nesse caso está em $(a_1,\ldots,a_l,b_1,\ldots,b_m)_1$.*

Para verificar que essa variação pode ser descrita em primeira ordem, basta considerar uma assinatura $\mathcal{S}^k = \{0,+1,-1,\prec,H_1,\ldots,H_k,T_1,\ldots,T_k\}$, onde $H_1,\ldots,H_k$ são os símbolos relacionais ternários de *cabeçote* e $T_1,\ldots,T_k$ são os símbolos relacionais ternários de *fita*, os demais símbolos são exatamente como na linguagem de Turing.

Definição 3.3.2. *Seja $\mathcal{M}$ um modelo aritmético. Uma* estrutura de Turing com k-fitas *é uma $\mathcal{S}^k$-estrutura de primeira ordem $\mathcal{A}^{\vec{s}}_{M^k}$ como na definição*

[5] Note que usaremos uma enumeração dos fatores nas configurações de M^k. Obviamente essa enumeração é possível de ser feita, uma vez que estamos tratando de seqüência finitas.

de máquina de Turing exceto pelo fato de que as interpretações de $H_1, \ldots, H_k$ *e* $T_1, \ldots, T_k$ *são relações ternárias em* $\mathcal{M}^3$ *definidas recursivamente sobre o terceiro fator da seguinte forma:*

- $(0,0,0) \in H_1^{\mathcal{A}^{\vec{s}}_{M^k}}, \ldots, (1,0,0) \in H_k^{\mathcal{A}^{\vec{s}}_{M^k}}$, $(2,0,0) \in T_1^{\mathcal{A}^{\vec{s}}_{M^k}}, (2,1,0) \in T_1^{\mathcal{A}^{\vec{s}}_{M^k}}, (s_1,2,0) \in T_1^{\mathcal{A}^{\vec{s}}_{M^k}}, \ldots, (s_n, n+1, 0) \in T_1^{\mathcal{A}^{\vec{s}}_{M^k}}, (2, n+2, 0) \in T_1^{\mathcal{A}^{\vec{s}}_{M^k}}, (2, n+3, 0) \in T_1^{\mathcal{A}^{\vec{s}}_{M^k}}$ *e, para todo* $y > n+3$, $(2, y, 0) \in T_1^{\mathcal{A}^{\vec{s}}_{M}}$, *bem como, para todo* z, $(2, z, 0) \in T_2^{\mathcal{A}^{\vec{s}}_{M^k}}, \ldots, (2, z, 0) \in T_k^{\mathcal{A}^{\vec{s}}_{M^k}}$;

- $(q, e, t) \in H_i^{\mathcal{A}^{\vec{s}}_{M^k}}$, $(s_i, e, t) \in T_i^{\mathcal{A}^{\vec{s}}_{M^k}}$ *e* $M^k(q, s_1, \ldots, s_k) = (r_1, \diamond, \ldots, r_k, \diamond, p)$ *para* $\diamond \in \{\triangleleft, \triangleright\}$ *se, e somente se,* $(p, e', t+1) \in H_i^{\mathcal{A}^{\vec{s}}_{M^k}}$ *com* $e' = e - 1$ *ou* $e' = e + 1$ *conforme* $\diamond = \triangleleft$ *ou* $\diamond = \triangleright$ *e* $(r_i, e, t+1) \in T_i^{\mathcal{A}^{\vec{s}}_{M^k}}$.

Não é necessário definirmos os axiomas explícitos de uma teoria de Turing com k-fitas porque basta considerar k versões dos axiomas $A3 - A7$ de acordo com os símbolos $H_1, \ldots, H_k$ e $T_1, \ldots, T_k$. As demonstrações da correspondência e completude seguem o mesmo roteiro do capítulo 2, mas necessitam de alterações relacionadas às peculiaridades de uma máquina de múltiplas fitas, algo laborioso de ser feito e não muito informativo. Ademais, trata-se de fato bastante conhecido que a adição de múltiplas fitas em uma máquina de Turing não interfere na estabilidade da computabilidade.

Uma variação das máquinas de Turing mais interessante é o conceito de máquina de Turing *não-determinista*. Esse tipo de máquina caracteriza-se pela capacidade de, em cada passo de sua computação, aplicar diferentes instruções simultaneamente. Uma definição formal desse modelo pode ser a seguinte.

Definição 3.3.3. *Seja* $\mathcal{M}$ *um modelo aritmético. Uma* Máquina de Turing não-determinista *é uma função parcial* N *tal que* $dom(N) \subseteq \{1, \ldots, n\} \times \{0, 1, 2\}$ *e* $cod(N) \subseteq \wp(\{0, \ldots, n\} \times \{0, 1, 2\} \times \{\triangleleft, \triangleright\})$ *para* $n \in \mathcal{M}$. *Uma* computação *de* N *com entrada* $\vec{s} = s_1, \ldots, s_m$ *é uma função parcial ou total* $C_N^{\vec{s}} : \mathbb{N} \to$

$\wp(\{0,1,2\}^\star \times \{0,\ldots,n\} \times \{0,1,2\}^\star)$ *definida recursivamente tal como na definição de máquina de Turing exceto pelas seguintes modificações:*

- $C_N^{\vec{s}}(0) = \{(\oslash, 0, (2, 2, s_1, \ldots, s_m, 2, 2))\}$;
- *Dada uma configuração instantânea* $C_N^{\vec{s}}(i) = \{((a_{j_1}, \ldots, a_{j_k}), q_j, (b_{j_1}, \ldots, b_{j_l}))_j\}_{j<u}$, *então* $C_N^{\vec{s}}(i+1)$ *é o menor conjunto em* $\wp(\{0,1,2\}^\star \times \{0,\ldots,n\} \times \{0,1,2\}^\star)$ *satisfazendo estas condições:*
- *Se* $(q, s, \triangleright) \notin N(q_j, b_{j_1})$ *para todo* j, *então* $((a_{j_1}, \ldots, a_{j_k}, s), q, (b_{j_2}, \ldots, b_{j_l}))_j \in C_M^{\vec{s}}(i+1)$;
- *Se* $(q, s, \triangleright) \in N(q_j, b_{j_1})$ *para algum* j *e* $(b_{j_2}, \ldots, b_{j_l}) \neq \oslash$, *então* $((a_{j_1}, \ldots, a_{j_k}, s), q, (b_{j_2}, \ldots, b_{j_l}))_j \in C_M^{\vec{s}}(i+1)$;
- *Se* $(q, s, \triangleright) \in N(q_j, b_{j_1})$ *para algum* j *e* $(b_{j_2}, \ldots, b_{j_l}) = \oslash$, *então* $((a_{j_1}, \ldots, a_{j_k}, s), q, 2)_j \in C_N^{\vec{s}}(i+1)$;
- *Se* $(q, s, \triangleleft) \in N(q_j, b_{j_1})$ *para algum* j *e* $(a_{j_1}, \ldots, a_{j_k}) \neq \oslash$, *então* $((a_{j_1}, \ldots, a_{j_{k-1}}), q, (a_{j_k}, s, b_{j_2}, \ldots, b_{j_l}))_j \in C_N^{\vec{s}}(i+1)$;
- *Se* $(q, s, \triangleleft) \in N(q_j, b_{j_1})$ *para algum* j *e* $(a_{j_1}, \ldots, a_{j_k}) = \oslash$, *então* $(\oslash, q, (s, b_{j_2}, \ldots, b_{j_l}))_j \in C_N^{\vec{s}}(i+1)$.

Uma configuração $C_N^{\vec{s}}(i)$ *é uma configuração final se todas as tuplas em* $C_M^{\vec{s}}(i)$ *são configurações finais no sentido da definição de máquina de Turing determinista. Em particular, escrevemos* $N[\vec{s}] = \vec{r}$ *para indicar que todas as configurações finais de* $C_N^{\vec{s}}$ *são iguais. Assim, uma função númerica* k*-ária* f *é* computável por N *quando* $f(\underline{x}_1, \ldots, \underline{x}_k) = \underline{y}$ *se, e somente se,* $N[2, 2, \underline{x}_1, 0, \underline{x}_2, 0, \ldots, 0, \underline{x}_k, 2, 2] = \underline{y}$.

Nessa definição estamos adotando as convenções de [Lewis and Papadimitriou(1998), p.222-223] sobre a parada de uma máquina de Turing não-determinista. Em especial, note a convenção de que

uma função é computável por uma máquina de Turing não-determinista N apenas se todas as possíveis escolhas de N convergem para uma determinada configuração. Essa exigência é necessária porque se alguma escolha divergisse, não teríamos como decidir qual seria o valor da computação. Em nossa proposta, as configurações de computações de máquinas de Turing não-deterministas são conjuntos de configurações de máquinas deterministas. Usualmente, costuma-se representar as computações não-deterministas como árvores nas quais os nós são as tuplas das configurações e os ramos são as diferentes possibilidades de escolha da máquina[6]. Abaixo mostramos que o teorema da correspondência continua valendo no caso de máquinas de Turing não-deterministas.

Teorema 3.3.1 (Correspondência não-determinista)**.** *Seja N uma máquina de Turing não-determinista e $C_N^{s_1,\ldots,s_n}$ uma computação de N. Então toda configuração em $C_N^{\vec{s}}$ é correspondível na linguagem de Turing $\mathcal{L}$.*

Demonstração. A demonstração é análoga ao caso determinístico apenas com a diferença que temos configurações da forma $C_M^{\vec{s}}(t) = \{((a_1,\ldots,a_k),q_1,(s_{1_1},\ldots,s_{1_l}))_1,\ldots,((b_1,\ldots,b_u),q_p,(s_{p_1},\ldots,s_{p_w}))\}$ se, e somente se, $\phi_1 \wedge \cdots \wedge \phi_p \wedge \forall v(\neg v \approx e \wedge \neg v \approx e+1 \wedge \cdots \wedge \neg v \approx e+c \rightarrow T(0,v,t))$ onde cada ϕ_i é da forma $H(q_i,e+n+1,t) \wedge T(a_1,e,t) \wedge T(a_2,e+1,t) \wedge \cdots \wedge T(a_n,e+n-1,t) \wedge T(s_{i_1},e+n,t) \wedge T(b_{i_2},e+n+1,t) \wedge \cdots \wedge T(b_{i_m},e+n+m,t)$ e $e,e+1,\ldots,e+c$ são todas as constantes para celas que ocorrem nas fórmulas $\phi_1,\ldots,\phi_p$. □

Definição 3.3.4. *Uma teoria $\mathcal{T}_N^{\vec{s}}$ associada a uma máquina de Turing não-determinista N com entrada $\vec{s} = s_1,\ldots,s_n$ é a teoria exatamente como uma teoria de Turing exceto pela substituição do axioma A4 pelo axioma A4′ abaixo:*

$$A4'\ \ \forall x\forall y\forall z\forall w(T(0,x,y)\vee T(1,x,y)\vee T(2,x,y))\wedge\forall x\forall y\forall z(2\prec x\rightarrow\neg T(x,y,z))\wedge$$
$$\forall x\forall y_1,\ldots,y_n\forall z\forall w\forall v(H(x,y_1,z)\wedge\cdots\wedge H(x,y_n,z)\wedge(\neg w\approx y_1\wedge\cdots\wedge\neg w\approx$$

[6] Cf. [Sipser(2006), p.138].

$y_n) \rightarrow \neg H(v,w,z)) \wedge \forall z((H(0,e,t) \wedge T(s,c,t) \wedge t \prec z) \rightarrow (H(0,e,z) \wedge T(s,c,z)))$

A sentença $\forall x \forall y_1, \ldots, y_n \forall z \forall w \forall v(H(x,y_1,z) \wedge \cdots \wedge H(x,y_n,z) \wedge (\neg w \approx y_1 \wedge \cdots \wedge \neg w \approx y_n) \rightarrow \neg H(v,w,z))$ garante que uma máquina de Turing não-determinista pode estar em diferentes celas em um dado instante e em diferentes estados na mesma cela, mas existe um limite de estados e celas em que ela pode estar em cada instante. Outra diferença importante é a sentença $\forall z((H(0,e,t) \wedge T(s,c,t) \wedge t \prec z) \rightarrow (H(0,e,z) \wedge T(s,c,z)))$ do axioma $A4'$. Note que ela expressa o comportamento de uma máquina de Turing não-determinista após entrar no estado final em um ramo particular. A sentença universal $\forall v \forall x \forall y \forall z \forall w((H(0,y,z) \wedge T(v,x,z) \wedge z \prec w) \rightarrow (H(0,y,w) \wedge T(v,x,w)))$ da axiomática para máquinas de Turing deterministas pode não ser verdadeira no caso não-determinístico porque os ramos podem entrar no estado final em instantes diferentes. Com relação aos símbolos, retiramos a condição $\forall v \forall w \forall x \forall y(T(v,x,y) \wedge T(w,x,y) \rightarrow v \approx w)$ no axioma $A3$ das máquinas deterministas que estabelecia a unicidade dos símbolos nas celas, essa sentença não é verdadeira pois as configurações de uma máquina de Turing não-determinista em cada passo podem ter várias tuplas. Assim, quando tivermos dois símbolos diferentes na mesma cela não há contradição alguma, acontece apenas que esses símbolos estão em ramos diferentes da computação não-determinista.

Teorema 3.3.2 (Representação não-determinista)**.** *Para todo t, se ϕ é a fórmula de $\mathcal{L}$ correspondente à configuração $C_N^{\vec{s}}(t)$ da máquina de Turing não-determinista N, então $\mathcal{T}_N^{\vec{s}} \vdash \phi$.*

Demonstração. A demonstração é análoga ao caso determinístico porque dado que as configurações são da forma $C_N^{\vec{s}}(t) = \{((a_{j_1}, \ldots, a_{j_k}), q_j, (b_{j_1}, \ldots, b_{j_l}))_j\}_{j<u}$, precisamos verificar que $\mathcal{T}_N^{\vec{s}}$ demonstra cada uma das ϕ_j em $\phi_0 \wedge \cdots \wedge \phi_{u-1}$ e $\forall v(\neg v \approx e \wedge \neg v \approx e+1 \wedge \cdots \wedge \neg v \approx e+c \rightarrow T(0,v,t))$. Em especial, os casos complicados são aqueles em que

$(b_{j_2}, \ldots, b_{j_l}) = \oslash$ em alguma das tuplas $((a_{j_1}, \ldots, a_{j_k}), q_j, (b_{j_1}, \ldots, b_{j_l}))_j$ e $N(q_j, b_{j_1}) = (q, s, \triangleright)$. Considere, então, um desses casos. Por hipótese de indução, $\mathcal{T}_N^{\vec{s}} \vdash H(q_i, e+n, t) \wedge T(a_1, e, t) \wedge T(a_2, e+1, t) \wedge \cdots \wedge T(a_n, e+n-1, t) \wedge T(2, e+n, t) \wedge \forall x(\neg x \approx e \wedge \cdots \wedge \neg x \approx e+n \rightarrow T(2, x, t))$. Pelo axioma $A4$, $\mathcal{T}_N^{\vec{s}} \vdash H(q_i, e+n, t) \wedge T(0, e+n, t) \rightarrow H(q, e+n+1, t+1) \wedge T(s, e+n, t+1) \wedge \forall v \forall y(\neg y \approx e \rightarrow (T(v, y, t) \leftrightarrow T(v, y, t+1)))$ e, consequentemente, $\mathcal{T}_N^{\vec{s}} \vdash H(q, e+n+1, t+1) \wedge T(s, e+n, t+1)$. Como $\mathcal{T}_N^{\vec{s}} \vdash \forall v \forall y(\neg y \approx e+n \rightarrow (T(v, y, t) \leftrightarrow T(v, y, t+1)))$, também temos que $\mathcal{T}_N^{\vec{s}} \vdash T(a_1, e, t+1) \wedge T(a_2, e+1, t+1) \wedge \cdots \wedge T(a_n, e+n-1, t+1)$ e precisamos verificar que $\mathcal{T}_N^{\vec{s}}$ demonstra $T(0, e+n+1, t+1)$. Ora, isso é uma consequência imediata do fato que $\mathcal{T}_N^{\vec{s}} \vdash \forall y(\neg y \approx e \wedge \cdots \wedge \neg y \approx e+n \rightarrow T(0, y, t))$. Ademais, sabemos que se $m \neq n+1$ então $\mathcal{T}_N^{\vec{s}} \vdash \neg m \approx n+1$ e, como $\mathcal{T}_N^{\vec{s}} \vdash \forall y(\neg y \approx e \wedge \cdots \wedge \neg y \approx e+n \rightarrow T(0, y, t))$, concluímos que $\mathcal{T}_N^{\vec{s}} \vdash \forall y(\neg y \approx e \wedge \cdots \wedge \neg y \approx e+n+1 \rightarrow T(0, y, t))$. Com um argumento similar aplicado a cada ϕ_i em $\phi_1 \wedge \cdots \wedge \phi_{u-1}$, teremos que $\mathcal{T}_N^{\vec{s}} \vdash \forall y(\neg y \approx e \wedge \neg y \approx e+1 \wedge \cdots \wedge \neg y \approx e+c \rightarrow T(0, y, t))$, onde $e, e+1, \ldots, e+c$ são todas as constantes para celas que ocorrem nas fórmulas $\phi_0, \ldots, \phi_{u-1}$. □

Devido ao teorema de representação 3.3.2, as mesmas propriedades modelo-teóricas verificadas para máquinas de Turing deterministas pode ser demonstradas para máquinas de Turing não-deterministas.

3.4 Universalidade

Segundo R. Gandy [Gandy(1988)], um dos maiores feitos de Turing em seu artigo [Turing(1936)] foi a definição de uma máquina de Turing universal com a capacidade de simular qualquer máquina de Turing. Na esteira do trabalho de Turing, Kleene [Kleene(1938)] também demonstrou a existência de funções recursivas parciais universais que computam tudo o que as funções recursivas computam. Conquanto as funções recursivas parciais universais de Kleene

sejam equivalentes às máquinas universais de Turing, a forma como esses conceitos foram obtidos é bastante diferente. Para entender essa diferença, precisamos de uma definição.

Definição 3.4.1. *Uma* encodificação *é uma função* $\ulcorner\ \urcorner$ *de* $\mathbf{Maq}$ *para um conjunto* $A \neq \mathbf{Maq}$ *que associa a cada máquina de Turing* $\mathbf{M} = \{\delta_i\}_{i<n}$ *em* $\mathbf{Maq}$ *um* código $\ulcorner \mathbf{M} \urcorner$ *definido conforme alguma atribuição dos estados, símbolos e movimentos em cada instrução* δ_i *para elementos em* A. *Uma* decodificação *é a relação inversa de uma encodificação, e uma* codificação *é um par composto por uma encodificação e decodificação. Quando* A *é um modelo aritmético* $\mathcal{M}$ *dizemos que a codificação é* numérica.

Turing definiu uma máquina de computação universal sem usar codificação numérica. Ele apresentou uma codificação das máquinas de Turing usando letras do alfabeto e, desse modo, definiu explicitamente um conjunto de instruções para uma máquina decodificar qualquer outra máquina e efetuar as operações correspondentes. Kleene usou o seu predicado recursivo primitivo $T_n(e, \vec{x}, y)$, que descreve as árvores de computação das funções recursivas via codificação numérica. Assim, ele definiu uma função recursiva primitiva U que quando aplicada à $\mu y T_n(e, \vec{x}, y)$ gera o valor no último passo da menor computação y da função n-ária com código e sobre a entrada $\vec{x}$. Apesar dessas diferenças, historicamente os trabalhos de Turing e Kleene foram condensados em uma única abordagem da universalidade.

Nos anos 1950, M. Davis [Davis(1956), Davis(1957)] definiu o conceito de máquina de Turing universal em termos do predicado de Kleene T e, conseqüentemente, estabeleceu que a codificação numérica é parte da definição de máquina de Turing universal. Seguindo a concepção inicial de Davis, hoje o uso da codificação numérica no estudo da universalidade é considerado central. Por exemplo, no recente artigo de J-C. Delvenne [Delvenne(2009)] máquinas de Turing universais são definidas módulo codificação. Retornando ao trabalho original de Turing, poderíamos nos perguntar: o que há de especial na codificação numérica? Qual é exatamente o papel da codificação em

geral para dispositivos de computação universais?

Essas perguntas não são desprovidas de motivação. Atualmente, uma das principais áreas de pesquisa em ciência da computação é a *computação quântica* [Nielson and Chuang(2000)]. Em termos gerais, essa área de pesquisa trata do problema de saber quais seriam as características da computação baseada nos princípios da mecânica quântica. Um dos trabalhos pioneiros nessa direção foi publicado em 1985 por Deutsch [Deutsch(1985)]. Nesse artigo Deutsch introduziu uma definição de *máquina de Turing quântica* enquanto um modelo preciso de computação baseada na mecânica quântica e, ademais, também propôs um modelo de máquina de Turing quântica universal. Embora muitos trabalhos tenham desenvolvido a idéia de máquinas de Turing quânticas, ainda hoje não há consenso sobre a existência de máquinas de Turing quânticas que sejam universais [Shi(2002), Fouché et al.(2008)Fouché, Heidema, Jones, and Potgieter]. Basicamente, o problema da universalidade em computação quântica está associado a possibilidade de combinar simulação (algo que envolve codificação) e superposição de configurações em computações quânticas[7].

Com uma abordagem modelo-teórica da computabilidade podemos esclarecer alguns aspectos do papel da codificação para o fenômeno da universalidade na computabilidade.

Lema 3.4.1. *As máquinas de Turing podem ser listadas efetivamente, sem usar codificação numérica.*

Demonstração. Uma vez que toda máquina de Turing tem um conjunto finito não-vazio de instruções, podemos definir uma seqüência $(S_i)_{i\in\mathbb{N}}$ onde cada S_i é o conjunto das máquinas de Turing em Maq com $i+1$ instruções. Por definição, o conjunto de estados de qualquer máquina de Turing tem a forma $\{0,1,\ldots,n\}$, então podemos demonstrar com uma indução simples que cada máquina de Turing em S_i tem no máximo $i+2$ estados, porque no máximo

[7] Cf. [Fouché et al.(2008)Fouché, Heidema, Jones, and Potgieter] para detalhes.

podemos alterar um estado por instrução. Dado que as instruções têm cinco componentes (estado,símbolo,símbolo,movimento,estado) e o número de estados em cada máquina é no máximo $i + 2$, também podemos afirmar que cada S_i tem no máximo

$$\frac{((i+2)^2 \cdot 18)!}{(i+1)! \cdot (((i+2)^2 \cdot 18) - (i+1))!}$$

máquinas de Turing, porquanto esse seja o número correspondente a combinação simples de todas as instruções possíveis em um dado conjunto S_i: $|\{0, 1, 2, \ldots, n\}| \leq i + 2$ e $18 = 3 \times 2 \times 3$, onde $3 = |\{0, 1, 2\}|$ e $2 = |\{\triangleleft, \triangleright\}|$. Logo, cada S_i pode ser efetivamente listado e, por extensão, podemos definir uma lista $(M_i)_{i\in\mathbb{N}}$ que contém todas as máquinas de Turing em Maq. □

Teorema 3.4.1. *Existe uma teoria $\mathcal{T}_U^{\vec{s}}$ tal que, para toda teoria de Turing $\mathcal{T}_M^{\vec{s}}$, $\mathcal{T}_M^{\vec{s}} \vdash \phi$ com ϕ uma fórmula correspondente a uma configuração se, e somente se, $\mathcal{T}_U^{\vec{s}} \vdash \psi$ para exatamente uma ψ da linguagem de $\mathcal{T}_U^{\vec{s}}$. Essa teoria $\mathcal{T}_U^{\vec{s}}$ pode ser efetivamente construída.*

Demonstração. Considere $(M_i)_{i\in\mathbb{N}}$ uma lista efetiva com todas as máquinas de Turing, descrita de acordo com o lema 3.4.1. A partir de $(M_i)_{i\in\mathbb{N}}$ também podemos formar a lista $(\mathcal{T}_i^{\vec{s}})_{i\in\mathbb{N}}$ onde $\mathcal{T}_i^{\vec{s}}$ é a teoria de Turing associada à máquina de Turing M_i. Tome os símbolos relacionais H e T da linguagem de Turing $\mathcal{L}$ e forme as listas $(H_i)_{i\in\mathbb{N}}$ e $(T_i)_{i\in\mathbb{N}}$ onde cada H_i e T_i são símbolos relacionais H e T da teoria de Turing $\mathcal{T}_i^{\vec{s}}$. Agora defina a *linguagem de Turing universal* $\mathcal{L}_U$ que é a linguagem de primeira ordem gerada pela *assinatura de Turing universal* $\mathcal{S}_U = \{0, +1, -1, \prec\} \cup \{H_i\}_{i\in\mathbb{N}} \cup \{T_i\}_{i\in\mathbb{N}}$. Por fim, defina a *teoria de Turing universal* $\mathcal{T}_U^{\vec{s}}$ como sendo o conjunto de sentenças de $\mathcal{L}_U$ que são axiomas de $\mathcal{T}_i^{\vec{s}}$ para cada $\mathcal{T}_i^{\vec{s}}$ em $(\mathcal{T}_i^{\vec{s}})_{i\in\mathbb{N}}$, ou seja, $\mathcal{T}_U^{\vec{s}} = \bigcup\{\mathcal{T}_i^{\vec{s}}\}_{i\in\mathbb{N}}$, onde cada $\mathcal{T}_i^{\vec{s}}$ é exatamente igual a teoria associa a máquina M_i exceto pelo fato de que seus símbolos relacionais H e T foram indexados por i. Seja $\mathcal{T}_M^{\vec{s}}$ uma teoria de Turing e ϕ uma fórmula de $\mathcal{L}$ que corresponde a uma configuração. Assim, pela construção de $(\mathcal{T}_i^{\vec{s}})_{i\in\mathbb{N}}$, $\mathcal{T}_M^{\vec{s}}$ está em $(\mathcal{T}_i^{\vec{s}})_{i\in\mathbb{N}}$, digamos que seja a

teoria $\mathcal{T}_k^{\vec{s}}$. Considere a fórmula ψ da linguagem $\mathcal{L}_U$ que é exatamente igual a fórmula ϕ exceto pelo fato de que os símbolos H e T que ocorrem em ϕ estão indexados por k. Claramente, $\mathcal{T}_k^{\vec{s}} \vdash \phi$ se, e somente se, $\mathcal{T}_U^{\vec{s}} \vdash \psi$. Como a lista $(M_i)_{i\in\mathbb{N}}$ que usamos para definir $\mathcal{T}_U^{\vec{s}}$ é uma lista efetiva e temos um algoritmo para definir $\mathcal{T}_U^{\vec{s}}$, concluímos que $\mathcal{T}_U^{\vec{s}}$ pode ser efetivamente construída. □

Teorema 3.4.2. *A teoria universal de Turing $\mathcal{T}_U^{\vec{s}}$ tem um modelo $\mathcal{A}_U^{\vec{s}}$, chamado* estrutura de Turing universal*, tal que, para toda $M_i \in Maq$, $M_i[\vec{s}] = \vec{r}$ se, e somente se, $\mathcal{A}_U^{\vec{s}} \vDash \phi^i \to \psi^i$ onde ϕ^i e ψ^i são fórmulas correspondentes às configurações inicial e final de $C_{M_i}^{\vec{s}}$.*

Demonstração. Primeiramente, mostraremos que $\mathcal{T}_U^{\vec{s}}$ tem um modelo. Seja Γ um subconjunto finito de $\mathcal{T}_U^{\vec{s}}$. Então Γ é um conjunto finito de sentenças com a forma dos axiomas $A1-A7$ mas com os símbolos relacionais H e T indexados por números naturais. Em virtude desses índices, sabemos que se cada teoria de Turing é consistente, então Γ é consistente - os índices evitam conflitos nas instruções das máquinas de Turing particulares. Ora, ou Γ tem todos os axiomas associados às instruções de uma ou mais máquinas de Turing particulares ou tem apenas os axiomas de uma única máquina de Turing. Em qualquer caso Γ tem um modelo, notadamente, a estrutura onde os símbolos relacionais H_i e T_i que ocorrem nas fórmulas de Γ são interpretados tal como os símbolos H e T das respectivas estruturas $\mathcal{A}_i^{\vec{s}}$ que são modelos das teorias $\mathcal{T}_i^{\vec{s}}$, e, quanto aos símbolos relacionais H_k e T_k da assinatura $\mathcal{S}_U$ que não ocorrem em fórmulas de Γ, basta interpretá-los como sendo a interpretação mínima, qual seja: aquela que satisfaz as condições relativas ao axioma $A7$. Portanto, pelo teorema da compacidade, $\mathcal{T}_U^{\vec{s}}$ tem um modelo.

Considere $\mathcal{A}_U^{\vec{s}} = (\mathbb{N}, 0, +1, -1, <, (H_i)_{i\in\mathbb{N}}, (T_i)_{i\in\mathbb{N}})$ como sendo o modelo intencionado de $\mathcal{T}_U^{\vec{s}}$. Se $M_i[\vec{s}] = \vec{r}$, então $\mathcal{A}_{M_i}^{\vec{s}} \vDash \phi \to \psi$ para fórmulas ϕ e ψ correspondentes às configurações inicial e final de $C_{M_i}^{\vec{s}}$, devido ao teorema da correspondência. Ademais, pelo teorema da completude, $\mathcal{T}_{M_i}^{\vec{s}} \vdash \phi \to \psi$. Como as fórmulas ϕ e ψ de $\mathcal{T}_{M_i}^{\vec{s}}$ tem versões indexadas ϕ^i e ψ^i em $\mathcal{T}_U^{\vec{s}}$ e $\mathcal{A}_U^{\vec{s}}$ é um modelo de $\mathcal{T}_U^{\vec{s}}$, concluímos que $\mathcal{A}_U^{\vec{s}} \vDash \phi^i \to \psi^i$. Conversamente, se

$\mathcal{A}_U^{\vec{s}} \vDash \phi^i \to \psi^i$, então, por uma adaptação direta do teorema da completude para $\mathcal{T}_U^{\vec{s}}$, derivamos que $\mathcal{T}_U^{\vec{s}} \vdash \phi^i \to \psi^i$. Ora, $\mathcal{T}_U^{\vec{s}} \vdash \phi^i \to \psi^i$ se, e somente se, $\mathcal{T}_{M_i}^{\vec{s}} \vdash \phi \to \psi$, onde ϕ e ψ são a fórmulas que geram ϕ^i e ψ^i via indexação. Daí, pelo teorema da corretude de $\mathcal{T}_{M_i}^{\vec{s}}$, $\mathcal{A}_{M_i}^{\vec{s}} \vDash \phi \to \psi$, o que acontece se, e somente se, $M_i[\vec{s}] = \vec{r}$. □

A partir do teorema 3.4.2, poderíamos ser tentados a derivar que existe uma máquina de Turing U a qual foi associada à estrutura de Turing universal $\mathcal{A}_U^{\vec{s}}$. No entanto, esse seria um passo ainda sem justificação, porque uma máquina de Turing deve ser uma função finita não-vazia e $\mathcal{A}_U^{\vec{s}} = (\mathbb{N}, 0, +1, -1, <, (H_i)_{i\in\mathbb{N}}, (T_i)_{i\in\mathbb{N}})$ é uma estrutura com infinitas relações H_i e T_i. Assim, a máquina de Turing que gerou $\mathcal{A}_U^{\vec{s}}$ aparentemente deveria ser uma função infinita. Não obstante, como pode existir uma estrutura de Turing sem existir uma máquina de Turing que a gerou?

Quando Lewis e Papadimitrou definem em seu livro [Lewis and Papadimitriou(1998), p.247-250] uma máquina de Turing universal através do método usual de codificação numérica das máquinas de Turing, eles enfatizam que a existência de tal máquina é uma conseqüência do fato de que "máquinas de Turing também são software". Ora, uma estrutura de Turing está associada ao hardware de uma máquina de Turing, i.e., ao fato de que "máquinas de Turing também são hardware". Portanto, o que podemos dizer a partir da existência da estrutura de Turing universal $\mathcal{A}_U^{\vec{s}}$ é que a universalidade é compatível com o conceito de hardware de uma máquina de Turing, o qual é necessariamente infinitário posto que deve ter a capacidade de simular todas as máquinas.

A finitude do software associado ao hardware que possivelmente gerou $\mathcal{A}_U^{\vec{s}}$ é garantida pela codificação, ou seja, ela é uma maneira de garantir a finitude de uma máquina de Turing universal. Se máquinas de Turing pudessem ser funções infinitas, então o teorema 3.4.2 em si já mostraria a existência de uma máquina de Turing universal. Entretanto, a finitude das máquinas de Turing é uma componente crucial e, por isso, a codificação faz parte da definição das

máquinas de Turing. Em resumo, somente com algum tipo de codificação podemos obter a conjunção entre universalidade e finitude.

3.5 Logicidade

Recentemente, R. Sylvan e J. Copeland [Sylvan and Copeland(2000)] sustentaram que a computabilidade é relativa à lógica, ou seja, o conceito de computabilidade depende das características do sistema lógico que adotamos. Em particular, Sylvan e Copeland sugeriram o desenvolvimento de uma "teoria da computabilidade paraconsistente" [Sylvan and Copeland(2000), p.196], porque seria possível definir máquinas de Turing que efetuam computações com contradições presentes nos dados de sua fita. J. Agudelo e W. Carnielli [Agudelo and Carnielli(2010)] levaram a cabo essa idéia, propondo um modelo rigoroso de máquinas de Turing paraconsistente e exibindo, inclusive, algumas conexões com a computação quântica. No entanto, cabe a pergunta: o conceito de computabilidade é realmente dependente de algum sistema lógico?

Com uma abordagem modelo-teórica da computabilidade, podemos apresentar uma resposta a essa pergunta.

No capítulo 2 definimos teorias de Turing com alguns axiomas que expressam propriedades básicas dos números naturais. A partir desses axiomas, podemos definir a expressão de absurdidade $\perp$ como uma abreviação para $0 \approx 1$. Portanto, em teorias de Turing podemos tratar a negação como um símbolo definido:

Internalização da negação: $\neg\varphi =_{def} \varphi \rightarrow 0 \approx 1$

Esse é um recurso utilizado em aritmética intuicionista, inicialmente proposto por Heyting [Heyting(1930)]. No entanto, M. Dummett [Dummett(1991), p.295] fora o primeiro a perceber que são necessários alguns ajustes técnicos para adotarmos a estratégia de Heyting, porque no seguinte axioma da aritmética já ocorre o símbolo de negação:

A0 $\forall x(0 \approx x + 1 \rightarrow 0 \approx 1)$

Como estamos definindo $\perp$ em termos de $0 \approx 1$ não podemos pressupor esse axioma, caso contrário, a definição seria circular. Em realidade, esse axioma é independente dos demais axiomas da aritmética de Peano e, como observado por R. T. Cook, J. Cogburn [Cook and Cogburn(2000)], os ajustes sugeridos por Dumment não funcionam. Em princípio, isso implica que a negação não é internalizável na aritmética nem em teorias de Turing, a menos que efeturemos modificações não-trivais. Não obstante, considere o seguinte axioma:

A0' $\forall x(0 \approx x + 1 \rightarrow 0 \approx x)$

O seguinte resultado mostra que para os propósitos da aritmética e computabilidade de Turing o axioma $A0'$ é suficiente para internalizar a negação.

Lema 3.5.1. *Qualquer teoria que contenha o axioma* $A0'$ *e o princípio de indução* $(\phi(0, \vec{x}) \wedge \forall y(\phi(y, \vec{x}) \rightarrow \phi(y + 1, \vec{x})) \rightarrow \forall z\phi(z, \vec{x})$ *deriva o axioma* $A0$.

Demonstração. Seja $\mathcal{T}$ uma teoria que tem entre seus axiomas a sentença $A0'$ e o princípio de indução. A demonstração é por indução. Claramente, $\mathcal{T}$ demonsta que $\neg 0 \approx 1$, porque $0 \approx 1 \rightarrow 0 \approx 1$ é uma verdade lógica. Pela hipótese de indução, o resultado é válido para $k + 1$, ou seja, $\mathcal{T}$ demonstra que $0 \approx x + 1 \rightarrow 0 \approx 1$, a qual é uma abreviação para $\neg 0 \approx x + 1$. Ora, pelo axioma $A0'$, $0 \approx (k + 1) + 1 \rightarrow 0 \approx k + 1$ e, por modus tolens, $\mathcal{T}$ demonstra $\neg 0 \approx (k + 1) + 1$. □

Definição 3.5.1. *Uma teoria* $\mathcal{T}$ *é* livre para negação *se* $\mathcal{T}$ *não tem um símbolo de negação mas ele pode ser internalizado em* $\mathcal{T}$ *como um símbolo definido.*

Teorema 3.5.1. *Existem teorias de Turing livres para negação para as quais vale o teorema da completude.*

Demonstração. Seja $\mathcal{T}_M^{\vec{s}}$ uma teoria de Turing. Para começar, retire o esquema de axioma $A3$ de $\mathcal{T}_M^{\vec{s}}$. Agora, mostraremos como eliminar a negação dos axiomas de $\mathcal{T}_M^{\vec{s}}$. A negação ocorre no axioma $A1$ apenas na fórmulas $\forall x(\neg x \prec x)$. Nesse caso, basta substituir $\forall x \neg x \prec x$ pela fórmula $\exists x(x \prec x) \rightarrow \forall y(y \approx 0)$. De fato, se $\mathcal{T}_M^{\vec{s}} \vdash \exists x(x \prec x)$, então, por modus ponens, $\mathcal{T}_M^{\vec{s}} \vdash \forall y(y \approx 0)$ e, por eliminação do universal, $\mathcal{T}_M^{\vec{s}} \vdash 1 \approx 0$. Assim, $\mathcal{T}_M^{\vec{s}} \vdash \exists x(x \prec x) \rightarrow 0 \approx 1$, i.e., $\mathcal{T}_M^{\vec{s}} \vdash \neg\exists x(x \prec x)$, que é o que precisamos. No axioma $A2$ ocorre a sentença $\forall x(0 \approx x + 1 \rightarrow 0 \approx 1)$, mas, pelo lema 3.5.1 podemos substitui-la pela sentença $\forall x(0 \approx x + 1 \rightarrow 0 \approx x)$. No axioma $A4$ basta substituirmos a sentença $\forall x \forall y \forall z \forall v \forall w((H(x, y, z) \wedge (\neg v \approx x \vee \neg w \approx y)) \rightarrow \neg H(v, w, z)) \wedge \forall v \forall x \forall y \forall z \forall w((H(0, y, z) \wedge T(v, x, z) \wedge z \prec w) \rightarrow (H(0, y, w) \wedge T(v, x, w)))$ pela sentença $\forall v \forall w \forall x \forall y(H(v, x, y) \wedge H(w, x, y) \rightarrow v \approx w) \wedge \forall v \forall w \forall x \forall y(H(x, v, y) \wedge H(x, w, z) \rightarrow v \approx w) \wedge \forall v \forall x \forall y \forall z \forall w((H(0, y, z) \wedge T(v, x, z) \wedge z \preccurlyeq w) \rightarrow (H(0, y, w) \wedge T(v, x, w)))$, porque o novo axioma expressa as mesmas propriedades dos símbolos e também expressa a unicidade dos estados com relação ao espaço-tempo, salvaguardando as mesmas propriedades com relação ao término de uma computação. Nos axiomas $A5$ e $A6$ basta substituirmos a sentença $\forall v \forall x(\neg v \approx y \rightarrow (T(x, y, z) \leftrightarrow T(x, v, z + 1)))$ pela sentença $\forall v \forall x(v \approx y \vee (T(x, y, z) \leftrightarrow T(x, v, z + 1)))$. Claramente, o teorema da representação será válido para $\mathcal{T}_U^{\vec{s}}$ definida dessa forma, ou seja, para todo t, se ϕ é a fórmula de $\mathcal{L}$ correspondente à configuração $C_M^{\vec{s}}(t)$ da máquina de Turing determinista M, então $\mathcal{T}_M^{\vec{s}} \vdash \phi$. Conseqüentemente, $\mathcal{T}_M^{\vec{s}}$ é completa com relação às fórmulas que correspondem a configurações. □

Esse teorema 3.5.1 não exclui a possibilidade de analisarmos modificações da negação para fins de análise da computabilidade, ele mostra que a negação em computabilidade é internalizável. De fato, esse resultado destaca que a computabilidade tem uma lógica subjacente mínima com relação à negação, notadamente, a lógica minimal de I. Johasson [Johasson(1936)]:

$$\neg I\ \frac{\begin{array}{c}[\varphi]\\ \vdots\\ \bot\end{array}}{\neg\varphi}$$

$$\neg E\ \frac{\varphi \quad \neg\varphi}{\bot}$$

Em especial, se restringirmos a lógica subjacente de uma teoria de Turing à lógica minimal intuicionista de Johasson, não é necessário impor condições de localidade em teorias de Turing.

Teorema 3.5.2. *Seja $\mathcal{T}^{\vec{s}}_M$ uma teoria de Turing em lógica minimal intuicionista e com os esquemas de axiomas abaixo no lugar de A5 e A6:*

(A5') $\forall y \forall z(H(q,y,z) \wedge T(s,y,z) \rightarrow H(p,y+1,z+1) \wedge T(r,y,z+1))$ *se* $(q,s,p,r,\triangleright) \in M$.

(A6') $\forall y \forall z(H(q,y,z) \wedge T(s,y,z) \rightarrow H(p,y-1,z+1) \wedge T(r,y,z+1))$ *se* $(q,s,p,r,\triangleleft) \in M$.

Assim, para todo t, se ϕ é a fórmula de $\mathcal{L}$ correspondente à configuração $C^{\vec{s}}_M(t)$ da máquina de Turing M, então $\mathcal{T}^{\vec{s}}_M \vdash \phi$.

Demonstração. A demonstração é análoga ao caso de uma teoria de Turing com os esquemas $A5$ e $A6$. Basta observar que nos casos mais complicados, teremos pelo axioma $A2$, $\mathcal{T}^{\vec{s}}_M \vdash \forall y \forall z(T(0,y,z) \vee T(1,y,z) \vee T(2,y,z))$ e, pela lógica intuicionista, $\mathcal{T}^{\vec{s}}_M \vdash T(0,y,z)$ ou $\mathcal{T}^{\vec{s}}_M \vdash T(1,y,z)$ ou $\mathcal{T}^{\vec{s}}_M \vdash T(2,y,z)$ para todo y e z.

Digamos que estamos no caso em que $(q, r'_1, p, s, \triangleright) \in M$, $r'_2, \ldots, r'_m = \oslash$ e $C^{\vec{s}}_M(t) = ((r_1, \ldots, r_k), q, (r'_1, \ldots, r'_m))$. Então, $C^{\vec{s}}_M(t+1) = ((r_1, \ldots, r_k, s), p, 2)$. Por hipótese de indução, $\mathcal{T}^{\vec{s}}_M \vdash H(q, e+n, t) \wedge T(r_1, e, t) \wedge T(r_2, e+1, t) \wedge \cdots \wedge T(r_n, e+n-1, t) \wedge T(r'_1, e+n, t) \wedge \forall y(\neg y \approx e \wedge \cdots \wedge \neg y \approx e+n \rightarrow T(0, y, t))$. Pelo axioma $A5'$, $\mathcal{T}^{\vec{s}}_M \vdash H(q, e+n, t) \wedge T(r'_1, e+n, t) \rightarrow H(p, e+n+1, t+1) \wedge T(s, e+n, t+1)$ e, consequentemente, $\mathcal{T}^{\vec{s}}_M \vdash H(p, e+n+1, t+1) \wedge T(s, e+n, t+1)$. Ora, se

$\mathcal{T}_M^{\vec{s}} \vdash T(2, e+n+1, t+1)$, pela corretude de $\mathcal{T}_M^{\vec{s}}$ teríamos $\mathcal{A}_M^{\vec{s}} \vDash T(2, e+n+1, t+1)$, o que é falso e, portanto, $\mathcal{T}_M^{\vec{s}} \nvdash T(2, e+n+1, t+1)$ e, similarmente, $\mathcal{T}_M^{\vec{s}} \nvdash T(1, e+n+1, t+1)$. Logo, $\mathcal{T}_M^{\vec{s}} \vdash T(0, e+n+1, t+1)$. Com um argumento similar também mostramos que $\mathcal{T}_M^{\vec{s}} \vdash T(r_1, e, t+1) \wedge T(r_2, e+1, t+1) \wedge \cdots \wedge T(r_n, e+n-1, t+1) \wedge T(r'_1, e+n, t)$. Falta apenas mostrar que $\mathcal{T}_M^{\vec{s}} \vdash \forall y(\neg y \approx e \wedge \cdots \wedge \neg y \approx e+n+1 \rightarrow T(0, y, t+1))$.

Suponha que $\mathcal{T}_M^{\vec{s}} \vdash \neg c \approx e \wedge \cdots \wedge \neg c \approx e+n+1$ para uma cela arbitrária c. Pela lógica intuicionista, ou $\mathcal{T}_M^{\vec{s}} \vdash T(0, c, t+1))$ ou $\mathcal{T}_M^{\vec{s}} \vdash T(1, c, t+1))$ ou $\mathcal{T}_M^{\vec{s}} \vdash T(2, c, t+1))$. Se $\mathcal{T}_M^{\vec{s}} \vdash T(2, c, t+1))$, então, pela corretude, $\mathcal{A}_M^{\vec{s}} \vDash T(2, c, t+1))$ e, pela hipótese de indução, isso implicaria que no instante t, a máquina M aplicou duas instruções, o que não é possível. Portanto, $\mathcal{T}_M^{\vec{s}} \nvdash T(2, c, t+1))$. Similarmente, $\mathcal{T}_M^{\vec{s}} \nvdash T(1, c, t+1))$. Logo, $\mathcal{T}_M^{\vec{s}} \vdash T(0, c, t+1))$. □

Esse resultado não vale no caso de teorias de Turing não-deterministas, porque pode acontecer de $\mathcal{A}_M^{\vec{s}} \vDash T(0, e, t)$, $\mathcal{A}_M^{\vec{s}} \vDash T(1, e, t)$ e $\mathcal{A}_M^{\vec{s}} \vDash T(2, e, t)$, ou seja, a conexão entre intuicionismo e computabilidade neste ponto não é corroborada. Isso não significa que uma possível conexão esteja fora de questão; o que o teorema 3.5.2 mostra é que a relação entre localidade e intuicionismo tem características diferentes quando se considera computações deterministas e não-deterministas. Outro ponto interessante seria determinar qual é a propriedade dual da localidade, seu significado e o sistema lógico relacionado. Sabemos que existe certa dualidade entre intuicionismo e paraconsistência[8], então provavelmente tal sistema será alguma lógica paraconsistente, mas não é óbvio qual será a propriedade em questão e se ela tem algum significado relevante para a computabilidade.

[8] Cf. [A.B.M. Brunner(2005)]

4. COMPUTABILIDADE MODELO-TEÓRICA CLÁSSICA NÃO-PADRÃO

Sob a pele das palavras há cifras e códigos.
Carlos Drummond de Andrade

Este capítulo investiga a computabilidade de Turing em modelos não-padrão da aritmética, mostrando, desse modo, que existe um princípio de internalidade aritmética na computabilidade. Na seção 4.1 apresentamos uma caracterização geral dos modelos não-padrão, com ênfase nos modelos aritméticos. Na seção 4.2 exibimos as principais características dos modelos não-padrão de teorias de Turing. Na seção 4.3 estudamos os limites do fenômeno de Tennenbaum, utilizando as técnicas desenvolvidas nesta tese. Na seção 4.4 demonstramos alguns resultados acerca das funções numéricas não-padrão, os quais mostram que existe um princípio de internalidade aritmética na teoria da computabilidade. Na seção 4.5. analisamos as teses de Church e Turing à luz dos resultados obtidos neste capítulo.

4.1 Modelos não-padrão em geral

Segundo H. Gaifman [Gaifman(2004)], um *modelo não-padrão* é uma interpretação de uma teoria que tem características notadamente diferentes daquelas intencionadas. Os modelos não-padrão foram descobertos por S. Skolem [Skolem(1922), Skolem.(1929), Skolem.(1934)] em formalizações da teoria dos conjuntos e teoria dos números. A existência de tais modelos é um fenômeno lógico abrangente associado às características das linguagens

de primeira ordem.

Exemplos notáveis e amplamente investigados de modelos não-padrão são aqueles que encontramos na classe dos modelos da aritmética de primeira ordem completa $Th(\mathcal{N})$. Ao que tudo indica, a primeira pessoa a tentar axiomatizar $Th(\mathcal{N})$ foi H. Grassmann [Grassmann(1861)]. Sua principal contribuição foi a caracterização axiomática da adição e multiplicação que usamos hodiernamente: $x + 0 \approx x$ e $x + (y + 1) \approx (x + y) + 1$ bem como $x \times 0 \approx 0$ e $x \times (y + 1) \approx (x \times y) + x$. Mais tarde, G. Peano [Peano(2002)] e, independentemente, R. Dedekind [Dedekind(1888)], descobriram as propriedades básicas dos números naturais, a saber: $x + 1 \approx y + 1 \rightarrow x \approx y$, $x + 1 \not\approx 0$ e $(\varphi(0) \wedge \forall x(\varphi(x) \rightarrow \varphi(x + 1))) \rightarrow \forall y\varphi(y)$. A axiomática composta por esses axiomas de Grassmann, Peano e Dedekind passou a ser chamada de *aritmética de Peano* devido a sua formulação explícita no trabalho de Peano.

Não obstante a esse progresso na axiomatização da aritmética, em 1931, K. Gödel [Gödel(1931)] mostrou que não é possível apresentar uma axiomática do sistema dos números naturais que seja efetiva, correta e completa. A estratégia de Gödel para demonstrar esse resultado de incompletude envolveu a formulação de uma sentença verdadeira e não-demonstrável em qualquer axiomática efetiva e consistente do sistema dos números naturais; essa sentença passou a ser chamada de *sentença de Gödel*.

Como a sentença de Gödel não é nem demonstrável nem refutável em qualquer axiomática efetiva do sistema dos números naturais, mas é intuitivamente verdadeira em tal estrutura, devem existir pelo menos duas extensões consistentes e não logicamente equivalentes da aritmética de Peano: em uma adicionamos a sentença de Gödel, a qual é verdadeira no sistema dos números naturais, na outra adicionamos sua negação, a qual é falsa no sistema dos números naturais, mas deve ser verdadeira em algum modelo da aritmética de Peano. Esse modelo no qual a negação da sentença de Gödel é verdadeira deve ser um modelo não-padrão.

Gödel [Gödel(1986b)] foi o primeiro a usar essa estratégia para demonstrar

a existência de modelos não-padrão da aritmética de Peano. Ela é um tanto indireta, mas é interessante porque depende apenas das características da própria aritmética. Em especial, esse tipo de demonstração mostra que a existência de modelos não-padrão é um fenômeno advindo, por um lado, da capacidade expressiva da aritmética e, por outro, da limitação expressiva da linguagem de primeira ordem. Esse tipo de argumento é um desafio para a perspectiva *estruturalista* da matemática.

O estruturalismo matemático está baseado no trabalho de P. Benacerraf [Benacerraf(1965)], no qual ele argumenta que, pelo fato de existirem diferentes definições de número natural tecnicamente não equivalentes, isso implica que tratar os números como objetos é algo inadequado. Assim, Benacerraf propôs que a matemática é uma coleção de teorias sobre estruturas, e não sobre objetos. Em outras palavras, o tema de uma teoria matemática é determinado pelas características de seus modelos como um todo, e não pelos objetos que o compõem. Em especial, a aritmética deveria tratar, a menos de isomorfismo, de uma única estrutura. O problema é explicar como identificamos o modelo intencionado da aritmética, face aos modelos não-padrão. Para S. Shapiro [Shapiro(1991), Shapiro(1997)], os modelos não-padrão são uma anomalia que deve ser banida da classe dos modelos aritméticos. Ele argumenta que os modelos não-padrão da aritmética são resquícios da tese de Hilbert, pois ao adotarmos uma aritmética de segunda ordem tais modelos deixam de existir.

Sabemos que a aritmética de segunda ordem é ω-categórica e, por isso, o argumento de Shapiro funciona. Entretanto, V. Halbach e L. Horsten [Halbach and Horsten(2005)], têm argumentado que recorrer à lógica de segunda ordem é apenas transferir o problema, porque a semântica de linguagens de segunda ordem depende da teoria dos conjuntos, a qual também tem modelos não-padrão. Mais do que isso, em linguagens de segunda ordem podemos expressar a hipótese do contínuo, ou seja, sua semântica é terrivelmente indecidível [Ebbinghaus et al.(2005)Ebbinghaus, Flum, and Thomas,

p.132-136].

Tendo essas dificuldades em vista, Halbach e Horsten [Halbach and Horsten(2005)] formularam uma abordagem baseada no teorema de S. Tennembaum [Tennenbaum(1959)]. De acordo com o teorema de Tennembaum, a menos de isomorfismo, o modelo padrão da aritmética de Peano é o único modelo no qual adição e multiplicação são computáveis, i.e., em modelos não-padrão nem uma dessas operações é computável. Assim, Halbach e Horsten defendem que o modelo intencionado da aritmética é aquele no qual adição e multiplicação são computáveis. Ora, a computabilidade clássica está definida sobre os números naturais e, para evitar circularidade, Halbach e Horsten propõem distinguir duas noções de computabilidade. A noção *prática de computabilidade* denota nossas práticas usuais quando efetuamos processos efetivos. Nesse contexto, algoritmos são instruções para manipular símbolos, sejam eles números naturais ou não. A noção *teórica de computabilidade* é a coleção de nossas definições matemáticas usuais do conceito de computabilidade; por exemplo, funções recursivas, computabilidade de Turing, λ-cálculo, e assim por diante.

Dessa forma, Halbach e Horsten recorrem às máquinas de Turing porque elas formalizam a noção prática de computabilidade. Em especial, Halbach e Horsten usam o fato de que máquinas de Turing computam sobre símbolos, o que, segundo eles, implicaria que elas não computam sobre números naturais não-padrão. Símbolos são diferentes, por sua vez, devido à sua configuração gráfica, eles se distinguem entre si diretamente. Assim, podemos evitar a circularidade: o modelo intencionado da aritmética é qualquer sistema notacional numérico, por exemplo, o sistema arábico decimal, onde as operações computáveis satisfazem os axiomas de Peano.

Em [de Araújo and Carnielli(2010)], argumentamos que a estratégia de Halbach e Horsten não funciona, porque o fato de máquinas de Turing computarem sobre símbolos não impede que elas computem sobre símbolos que representam números naturais não-padrão. De fato, no segundo capítulo

desta tese desenvolvemos uma abordagem descritiva da computabilidade em primeira ordem. Neste capítulo construiremos modelos não-padrão de teorias de Turing, o que mostra que o problema da identificação do modelo intencionado da aritmética continua em aberto[1]. No que se segue várias peculiaridades e aspectos intrigantes desse fenômeno serão analisados.

4.2 Modelos não-padrão de teorias de Turing

No capítulo 2 associamos estruturas e teorias de primeira ordem às máquinas de Turing. Agora podemos aplicar o método de Henkin para construir modelos não-padrão dessas teorias.

Definição 4.2.1. *Seja M uma máquina de Turing e $\mathcal{T}_M^{\vec{s}}$ a teoria de Turing associada a M. O* modelo padrão *de $\mathcal{T}_M^{\vec{s}}$ é a estrutura $\mathcal{A}_M^{\vec{s}} = (\mathbb{N}, 0, +1, -1, <, H, T)$, cujos elementos são chamados* elementos padrão*. Qualquer outro modelo de $\mathcal{T}_M^{\vec{s}}$ que não seja isomórfico ao modelo padrão é chamado de* modelo não-padrão*, e seus elementos são chamados* elementos não-padrão*.*

Proposição 4.2.1. *Existem modelos não-padrão de teorias de Turing.*

Demonstração. Seja $\mathcal{S} = \{\underline{n}\}_{n\in\mathbb{N}} \cup \{+1, -1, \prec, H, T\}$ a assinatura de Turing, $\mathcal{T}_M^{\vec{s}}$ uma teoria de Turing qualquer e $\mathcal{A}_M^{\vec{s}}$ o modelo padrão de $\mathcal{T}_M^{\vec{s}}$. Considere a assinatura $\mathcal{S}^a$ obtida de $\mathcal{S}$ pela adição de um novo símbolo de constante a. Defina a $\mathcal{S}^a$-teoria

$$\mathcal{T}_M^{[a]\vec{s}} = \mathcal{T}_M^{\vec{s}} \cup \{n \prec a : \text{para cada } n \in \mathbb{N}\}^2.$$

[1] O autor desta tese conversou com Halbach sobre esse assunto, e Halbach concordou que a existência de elementos não-padrão em máquinas de Turing é um desafio para sua tentativa de identificar o modelo intencionado da aritmética. Ele acredita, todavia, que isso mostra que não há como identificar esse modelo sem pressupor algo como dado. Em sua abordagem, o pressuposto é que máquinas de Turing computam sobre símbolos que denotam objetos finitários.

[2] Vale lembrar que o símbolo n na sentença "$n \prec a$" é uma abreviação para 0 seguido por n aplicações de +1.

Claramente, cada subconjunto finito de $\mathcal{T}_M^{[a]\vec{s}}$ está contido em um conjunto da forma $\mathcal{T}_M^{\vec{s}} \cup \{\underline{n} \prec a : n < b\}$, para algum $b \in \mathbb{N}$ apropriado. Disso segue-se que cada um desses subconjuntos tem um $\mathcal{S}^a$-modelo da forma $\mathcal{A}_M^{[a]\vec{s}} = (\mathbb{N}, \{n\}_{n\in\mathbb{N}}, b, +1, -1, <, H, T)$ onde $a^{\mathcal{A}_M^{[a]\vec{s}}} = b$. Pela compacidade da lógica de primeira ordem, $\mathcal{T}_M^{[a]\vec{s}}$ tem um $\mathcal{S}^a$-modelo $\mathcal{B}_M^{[a]\vec{s}}$ e, portanto, o seu $\mathcal{S}$-reduto $\mathcal{B}_M^{\vec{s}}$ é um modelo de $\mathcal{T}_M^{\vec{s}}$.

Resta mostrar que não existe um isomorfismo entre $\mathcal{B}_M^{\vec{s}}$ e $\mathcal{A}_M^{\vec{s}}$. Suponha, pelo contrário, que $f : \mathcal{A}_M^{\vec{s}} \to \mathcal{B}_M^{\vec{s}}$ seja um isomorfismo. Assim, $f(\underline{n}^{\mathcal{A}_M^{\vec{s}}}) = \underline{n}^{\mathcal{B}_M^{\vec{s}}}$ para todo $n \in \mathbb{N}$, pois $f(\underline{0}^{\mathcal{A}_M^{\vec{s}}}) = \underline{0}^{\mathcal{B}_M^{\vec{s}}}$ e $f(\underline{n+1}^{\mathcal{A}_M^{\vec{s}}}) = \underline{n}^{\mathcal{B}_M^{\vec{s}}} + 1$. Pela definição de $\mathcal{B}_M^{\vec{s}}$, existe um elemento $a \in dom(\mathcal{B}_M^{\vec{s}})$ tal que $n < a$ para todo $n \in \mathbb{N}$. Devido ao axioma $A1$, $\mathcal{B}_M^{\vec{s}} \vDash \forall x \forall y (y \prec x \to \neg x \approx y)$ e, consequentemente, esse elemento a não está na imagem de f sobre $\mathbb{N}$, o que é uma contradição. Logo, por redução ao absurdo, não existe um isomorfismo entre $\mathcal{B}_M^{\vec{s}}$ e $\mathcal{A}_M^{\vec{s}}$. □

A demonstração da proposição 4.2.1 é uma simples aplicação do argumento usual com base no método de Henkin[3]. Cabe destacar, todavia, que ela apenas foi possível porque conseguimos definir estruturas de Turing como extensões da estrutura dos números naturais com o elemento distinguido zero, as funções sucessor e predecessor e a ordem estrita. A partir dessa proposta, também podemos verificar que, similarmente à aritmética, teorias de Turing não reconhecem os elementos não-padrão em seus modelos.

Definição 4.2.2. *Seja $\mathcal{T}_M^{\vec{s}}$ uma teoria de Turing e $\mathcal{B}_M^{\vec{s}}$ um modelo de $\mathcal{T}_M^{\vec{s}}$. Um subconjunto não-vazio I de $dom(\mathcal{B}_M^{\vec{s}})$ é um* corte *de $\mathcal{B}_M^{\vec{s}}$ quando, para todo $y \in I$, $x < y$ implica $x \in I$ e I é fechado sob a função sucessor. O conjunto I é um corte* próprio *se $I \neq dom(\mathcal{B}_M^{\vec{s}})$.*

Teorema 4.2.1. *Seja $\mathcal{T}_M^{\vec{s}}$ uma teoria de Turing e $\mathcal{B}_M^{\vec{s}}$ um modelo não-padrão de $\mathcal{T}_M^{\vec{s}}$. Se I é um corte próprio de $\mathcal{B}_M^{\vec{s}}$, então I não é definível em $\mathcal{B}_M^{\vec{s}}$.*

Demonstração. Suponha, a fim de obtermos uma contradição, que I é um corte próprio definível em um modelo não-padrão $\mathcal{B}_M^{\vec{s}}$ de uma teoria de Turing

[3] Cf. [Hodges(2005)] para detalhes sobre esse método.

$\mathcal{T}_M^{\vec{s}}$. Digamos que $\phi(x, \vec{y})$ é a fórmula que define I em $\mathcal{B}_M^{\vec{s}}$. Então, para alguma seqüência $\vec{a}$ de elementos em $dom(\mathcal{B}_M^{\vec{s}})$, $\phi(x, \vec{a})$ é uma fórmula tal que $I = \{b \in dom(\mathcal{B}_M^{\vec{s}}) : \mathcal{B}_M^{\vec{s}} \vDash \phi(b, \vec{a})\}$.

Sabemos que na aritmética de Peano, cuja assinatura é o conjunto $\{0, 1, +, \times, \prec\}$, não podemos definir um corte próprio de um modelo não-padrão [Odifreddi(1992a), p.45]. Consequentemente, também não podemos definir I com uma fórmula da linguagem de Turing usando apenas os símbolos para as funções sucessor, predecessor e a relação de ordem, porque tais operações e relações também estão disponíveis na aritmética de Peano. Assim, devido à hipótese de que $\phi(x, \vec{a})$ define I, derivamos que os símbolos relacionais H ou T devem ocorrer como símbolos principais em subfórmulas de $\phi(x, \vec{a})$.

Uma vez que I é um conjunto não-vazio fechado sob a função sucessor, temos que $\mathcal{B}_M^{\vec{s}} \vDash \phi(0, \vec{a}) \wedge \forall x(\phi(x, \vec{a}) \rightarrow \phi(x + 1, \vec{a}))$. Pelo axioma $A2$, a função sucessor é injetora e, como ϕ tem subfórmulas com símbolo principal H ou T, os elementos x tais que $\mathcal{B}_M^{\vec{s}} \vDash \phi(x, \vec{a})$ devem ser celas ou instantes (possivelmente ambos). Dessa forma, para cada $x \in dom(\mathcal{B}_M^{\vec{s}})$ existe $y \in dom(\mathcal{B}_M^{\vec{s}})$ tal que $\mathcal{B}_M^{\vec{s}} \vDash \phi(y, \vec{a})$, onde x e y são celas ou instantes. Isso significa que $\mathcal{B}_M^{\vec{s}}$ deve estar associada a uma computação $C_M^{\vec{s}}$ de M que no sentido de $\mathcal{B}_M^{\vec{s}}$ não pára. Dado que I é um corte próprio de $\mathcal{B}_M^{\vec{s}}$, deve existir um $c \in dom(\mathcal{B}_M^{\vec{s}})$ tal que $c \notin I$. Entretanto, isso implica que $\mathcal{B}_M^{\vec{s}} \vDash \neg\phi(c, \vec{a})$ para algum $c \in dom(\mathcal{B}_M^{\vec{s}})$, i.e., $C_M^{\vec{s}}$ não é total em $dom(\mathcal{B}_M^{\vec{s}})$ e, conseqüentemente, $\mathcal{B}_M^{\vec{s}}$ deve estar associada a uma computação $C_M^{\vec{s}}$ de M que no sentido de $\mathcal{B}_M^{\vec{s}}$ pára. Assim, obtemos um absurdo. □

Diferentemente do que ocorre no estudo dos modelos não-padrão da aritmética, na demonstração do teorema 4.2.1 não usamos o axioma de indução. O seguinte resultado mostra que essa demonstração livre do princípio de indução é importante porque ela revela aspectos peculiares das teorias de Turing.

Definição 4.2.3. *Seja $\mathcal{B}_M^{\vec{s}}$ uma estrutura de Turing qualquer. Dizemos que uma relação R n-ária sobre $dom(\mathcal{B}_M^{\vec{s}})$ é* interna *sobre $\mathcal{B}_M^{\vec{s}}$ quando*

existe uma fórmula de primeira ordem $\phi(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{m-1})$ *da linguagem de Turing, com as variáveis livres indicadas, que define* R*, ou seja, a extensão* $\phi^{\mathcal{B}^{\vec{s}}_M}$ *de* ϕ *em* $\mathcal{B}^{\vec{s}}_M$ *é o conjunto* $\{\vec{a} \in dom(\mathcal{B}^{\vec{s}}_M)^n : \mathcal{B}^{\vec{s}}_M \vDash \phi[\vec{a}, \vec{b}]$ *para uma dada seqüência* $\vec{b} \in dom(\mathcal{B}^{\vec{s}}_M)^m\}$*. Caso não exista tal fórmula,* R *é uma relação* externa *sobre* $\mathcal{B}^{\vec{s}}_M$*.*

Corolário 4.2.1. *Se* $\mathcal{B}^{\vec{s}}_M$ *é um modelo não-padrão de uma teoria de Turing* $\mathcal{T}^{\vec{s}}_M$*, então o conjunto* $\{n \in dom(\mathcal{B}^{\vec{s}}_M) : n \in \mathbb{N}\}$ *é externo sobre* $\mathcal{B}^{\vec{s}}_M$*.*

Demonstração. Se $\{n \in dom(\mathcal{B}^{\vec{s}}_M) : n \in \mathbb{N}\}$ fosse interno sobre $\mathcal{B}^{\vec{s}}_M$, então ele seria definível. No entanto, $\{n \in dom(\mathcal{B}^{\vec{s}}_M) : n \in \mathbb{N}\}$ é um corte próprio de $\mathcal{B}^{\vec{s}}_M$. Logo, pelo teorema 4.2.2, esse conjunto é externo sobre $\mathcal{B}^{\vec{s}}_M$. □

Observação 4.2.1. *O corolário 4.2.1 mostra que, do ponto de vista de uma máquina de Turing, os números naturais não são distinguidos internamente entre padrão e não-padrão. Conseqüentemente nossa formalização usual do conceito de* finitude *- um conjunto é finito quando existe uma bijeção entre ele e um número natural - passa a ser relativa ao modelo sob consideração. Dessa forma, uma computação com elemento não-padrão é,* internamente*, um conjunto finito porque podemos efetuar uma bijeção com algum número não-padrão, mas,* externamente*, ela é um conjunto infinito posto que tem pelo menos todos os números naturais padrão. Essa distinção é central porque indica as peculiaridades da computabilidade sobre números naturais não-padrão.*

Em geral, o estudo da computabilidade clássica pressupõe os números naturais padrão como dados, mas no λ-cálculo podemos definir os ordinais de Church-Kleene que são, em realidade, definições dos números naturais padrão como λ-termos[4]. O teorema 4.2.1 e seu corolário 4.2.1 mostram que os números naturais padrão são objetos externos à computabilidade e, portanto, os números naturais também são objetos pressupostos no λ-cálculo.

[4] Cf. [Barendregt(1984)]

Devido às características dos elementos não-padrão, indicadas pela proposição 4.2.1, podemos ademais verificar que, similarmente aos modelos não-padrão enumeráveis da aritmética, a ordem de qualquer modelo não-padrão enumerável de uma teoria de Turing tem um segmento inicial isomórfico aos números naturais padrão seguido por segmentos cuja ordem é isomórfica à ordem nos números inteiros, os quais estão, por sua vez, densamente ordenados. Costumamos denotar esse tipo de ordem por $\omega + (\omega^* + \omega) \cdot \eta$, onde ω é a ordem dos números naturais padrão, ω^* é a ordem inversa de ω, e η é a ordem dos números racionais.

Proposição 4.2.2. *Se $\mathcal{B}^{\vec{s}}_M$ é um modelo não-padrão enumerável de uma teoria de Turing $\mathcal{T}^{\vec{s}}_M$, então $\mathcal{B}^{\vec{s}}_M| \prec \cong \omega + (\omega^* + \omega) \cdot \eta$, onde $\mathcal{B}^{\vec{s}}_M| \prec$ indica a retrição de $\mathcal{B}^{\vec{s}}_M$ à linguagem $\{\prec\}$.*

Demonstração. Esse resultado é uma conseqüência da proposição 4.2.1 juntamente com o seguinte lema.

Lema 4.2.1. *Se $\mathcal{T}^{\vec{s}}_M$ é uma teoria de Turing, então $\mathcal{T}^{\vec{s}}_M \vdash \forall x(x < n+1 \vee x = n+1 \leftrightarrow (x = 0 \vee \cdots \vee x = n \vee x = n+1))$, onde n é o termo 0 seguido por n aplicações de $+1$.*

Demonstração. A demonstração é por indução externa em n. Suponha que $n = 0$. Então $\mathcal{T}^{\vec{s}}_M \vdash x \approx 0 \rightarrow (x \prec 0 \vee x \approx 0)$ pela lógica proposicional. Ademais, $\mathcal{T}^{\vec{s}}_M \cup \{x \prec 0 \vee x \approx 0\} \vdash x \prec 0 \vee x \approx 0$ e, pelo axioma $A1$, $\mathcal{T}^{\vec{s}}_M \vdash \neg x \prec 0$, o que implica que $\mathcal{T}^{\vec{s}}_M \cup \{x \prec 0 \vee x \approx 0\} \vdash x \approx 0$ e, por lógica, $\mathcal{T}^{\vec{s}}_M \vdash (x \prec 0 \vee x \approx 0) \rightarrow x \approx 0$. Agora suponha que o resultado seja verdadeiro para um n arbitrário. Assim, por hipótese de indução $\mathcal{T}^{\vec{s}}_M \vdash x \prec n \vee x \approx n \leftrightarrow (x \approx 0 \vee \cdots \vee x \approx n)$. Devido ao axioma $A2$, $\mathcal{T}^{\vec{s}}_M \vdash x \prec n+1 \leftrightarrow (x \prec n \vee x \approx n)$ e, pela lógica proposicional, $\mathcal{T}^{\vec{s}}_M \vdash (x \prec n+1 \vee x \approx n+1) \leftrightarrow (x \prec n \vee x \approx n \vee x \approx n+1)$. Por fim, usando a hipótese de indução, concluímos que $\mathcal{T}^{\vec{s}}_M \vdash x \prec n+1 \vee x \approx n+1 \leftrightarrow (x \approx 0 \vee \cdots \vee x \approx n \vee x \approx n+1)$. □

Os detalhes da demonstração são análogos ao argumento para os modelos aritméticos e podem ser encontrados em [Kaye(1991), p.75]. □

O axioma $A1$ estabelece que os elementos em qualquer modelo de uma teoria de Turing estão discretamente ordenados. Posto que a proposição 4.2.2 mostra que a ordem dos modelos não-padrão enumeráveis de uma teoria de Turing é densa, deduzimos que esse tipo de ordem é reconhecida apenas *externamente*, porque *internamente* a teoria define uma ordem discreta.

4.3 O fenômeno de Tennenbaum de um ponto de vista modelo-teórico

A possibilidade de demonstração do teorema de Matiyasevich-Robinson-Davis-Putnam (MRDP), segundo o qual todo conjunto computacionalmente enumerável é diofantino, em certas aritméticas fracas, por exemplo aritmética de Buss [Buss(1986)], implica na solução de alguns problemas centrais em complexidade computacional [Kaye(1991), pp.273-275]. Ora, se uma teoria aritmética demonstra o teorema de MRDP, então todo modelo dessa teoria não é computável [Kaye(1993)]. Portanto, determinar qual é a expressividade mínima que uma teoria aritmética deve ter para demonstrar o teorema de Tennenbaum tem estrita conexão com problemas de complexidade computacional.

Por isso, existe uma linha de pesquisa em aritmética que procura descobrir em quais teorias aritméticas o teorema de Tennenbaum é válido. J. C. Shepherdson [Shepherdson(1964)] mostrou que os limites do fenômeno de Tennenbaum estão relacionados com a expressividade do princípio de indução. Shepherdson construiu modelos não-padrão computáveis de teorias aritméticas com o princípio de indução restrito às fórmulas abertas. Mais recentemente A. Berarducci e M. Otero [Berarducci and Otero(1996)] estenderam o trabalho de Shepherdson para teoria aritméticas com o princípio de indução restrito às fórmulas abertas mais o axioma da normalidade $(\forall x, y, z_1, \ldots, z_{n-1}(\neg y \approx 0 \wedge ((x^n + (z_1 \times x^{n-1} \times y + \cdots + z_{n-1} \times x \times y^{n-1}) + y^n) \approx 0) \rightarrow \exists z(y \times z \approx x)))$. Entretanto, K. McAloon [McAloon(1982)] mostrou

que o teorema de Tennenbaum é verdadeiro na aritmética com princípio de indução restrito às fórmulas com quantificadores limitados, e G. Wilmer [Wilmer(1985)] demonstrou o mesmo resultado para a indução restrita às fórmulas existênciais[5].

Um ponto comum nesses estudos sobre os limites do teorema de Tennenbaum é o uso de funções recursivas; nenhum dos trabalhos usa máquinas de Turing. Parece que a inexistência de uma abordagem que possibilitasse analisar a computabilidade de Turing sobre números naturais não-padrão foi o principal fator para isso; algo semelhante ao que mencionamos na introdução acerca dos trabalhos de caracterização modelo-aritmética dos problemas de complexidade computacional. A fim de mostrarmos as potencialidades da computabilidade modelo-teórica, apresentaremos uma demonstração do teorema de Tennenbaum usando máquinas de Turing. Antes disso, transferiremos alguns resultados vinculados ao fenômeno de Tennenbaum para o contexto dos modelos não-padrão de teorias de Turing.

Proposição 4.3.1 (Transbordamento computacional)**.** *Seja $\mathcal{T}_M^{\vec{s}}$ uma teoria de Turing, $\mathcal{B}_M^{\vec{s}}$ um modelo não-padrão de $\mathcal{T}_M^{\vec{s}}$, I um corte próprio de $\mathcal{B}_M^{\vec{s}}$, $\vec{a}$ uma sequência de elementos em $dom(\mathcal{B}_M^{\vec{s}})$ e $\phi(x,\vec{a})$ uma fórmula da linguagem de Turing. Dessa forma:*

1. *Se $\mathcal{B}_M^{\vec{s}} \vDash \phi(b,\vec{a})$ para todo $b \in I$, então existe $c > I$ em $dom(\mathcal{B}_M^{\vec{s}})$ tal que $\mathcal{B}_M^{\vec{s}} \vDash \forall x \leqslant c\phi(x,\vec{a})$.*

2. *Se, para todo $x \in I$, existe $y \in I$ tal que $\mathcal{B}_M^{\vec{s}} \vDash y \geq x \wedge \phi(y,\vec{a})$, então, para cada $c > I$ em $dom(\mathcal{B}_M^{\vec{s}})$, existe b em $dom(\mathcal{B}_M^{\vec{s}})$ com $I < b < c$ e $\mathcal{B}_M^{\vec{s}} \vDash \phi(b,\vec{a})$.*

Demonstração. Para verificar (1), suponha que $\mathcal{B}_M^{\vec{s}} \vDash \phi(b,\vec{a})$ para todo $b \in I$. Se não existisse $c > I$ em $dom(\mathcal{B}_M^{\vec{s}})$ tal que $\mathcal{B}_M^{\vec{s}} \vDash \forall x \leqslant c\phi(x,\vec{a})$, então I seria definível pela fórmula $\forall y(y \prec x \rightarrow \phi(y,\vec{a}))$, o que contraria o teorema 4.2.1.

[5] Cf. [Kaye(2006)] para mais detalhes.

Para obter (2), basta aplicar (1) a fórmula $\exists y(x \leqslant y \prec c \wedge \phi(y, \vec{d}))$, onde c em $dom(\mathcal{B}_M^{\vec{s}})$ é um elemento arbitrário satisfazendo $c > I$. □

Proposição 4.3.2 (Vazamento computacional)**.** *Seja $\mathcal{T}_M^{\vec{s}}$ uma teoria de Turing, $\mathcal{B}_M^{\vec{s}}$ um modelo não-padrão de $\mathcal{T}_M^{\vec{s}}$, I um corte próprio de $\mathcal{B}_M^{\vec{s}}$, $\vec{d}$ uma seqüência de elementos em $dom(\mathcal{B}_M^{\vec{s}})$, e $\phi(x, \vec{d})$ uma fórmula da linguagem de Turing. Dessa forma:*

1. *Se, para todo para $c > I$ em $dom(\mathcal{B}_M^{\vec{s}})$, $\mathcal{B}_M^{\vec{s}} \vDash \phi(c, \vec{d})$, então, para algum $b \in I$, $\mathcal{B}_M^{\vec{s}} \vDash \forall x \geqslant b\phi(b, \vec{d})$;*

2. *Se, para todo $c > I$ em $dom(\mathcal{B}_M^{\vec{s}})$, existe $x > I$ tal que $\mathcal{B}_M^{\vec{s}} \vDash \phi(x, \vec{d}) \wedge x \prec c$, então, para todo $b \in I$, existe $y \in I$ tal que $\mathcal{B}_M^{\vec{s}} \vDash y \geqslant b \wedge \phi(y, \vec{d})$.*

Demonstração. A afirmação (1) pode ser constata por contraposição. Suponha que existam ilimitados $x \in I$ satisfazendo $\mathcal{B}_M^{\vec{s}} \vDash \neg\phi(x, \vec{d})$. Então $M \vDash \neg\phi(x, \vec{d})$ para algum $x > I$, devido à proposição 4.3.1. Similarmente, um simples argumento por contraposição também assegura a veracidade do enunciado (2). Suponha que $b \in I$ e $\mathcal{B}_M^{\vec{s}} \vDash \neg\phi(x, \vec{d})$ para todos $x \geq b$ em I. Então $\mathcal{B}_M^{\vec{s}} \vDash y \prec b \vee \neg\phi(y, \vec{d})$ para todo $y > I$ e, por overspill, existe $c \in I$ tal que $\mathcal{B}_M^{\vec{s}} \vDash \forall y \leqslant c(y \prec b \vee \neg\phi(y, \vec{d}))$, o que implica que o antecedente de (2) é falso. □

Esses resultados de transbordamento e vazamento computacionais mostram que as características básicas dos modelos não-padrão da aritmética podem ser analisadas no contexto dos modelos não-padrão de teorias de Turing. Eles são mais uma evidência de que teorias de Turing devem ter o princípio de indução, porque, conforme [Kaye(1991), p.72], transbordamento para cortes arbitrários é equivalente ao princípio de indução.

Agora passaremos à análise do fenômeno de Tennenbaum.

Definição 4.3.1. *Uma estrutura $\mathcal{M}$ na linguagem aritmética é* computável *se, e somente se, existem números naturais $\underline{0}, \underline{1} \in \mathbb{N}$, funções computáveis*

$\oplus, \otimes : \mathbb{N}^2 \to \mathbb{N}$ *e uma relação computável* $\circledcirc \subseteq \mathbb{N}^2$ *tais que* $\mathcal{M} \cong (\mathbb{N}, \underline{0}, \underline{1}, \oplus, \otimes, \circledcirc)$.

Claramente, o modelo padrão da aritmética $\mathbb{N}$ é computável. Além disso, apenas estruturas aritméticas enumeráveis podem ser computáveis. Para demonstrar o teorema de Tennenbaum, assim como A. McAloon [McAloon(1982)], usaremos conjuntos computacionalmente enumeráveis, com a diferença de que os abordaremos via máquinas de Turing.

Nesse sentido, vale lembrar algumas definições. Dois conjuntos de números naturais A e B são *computacionalmente separáveis* quando existe um conjunto de números naturais computável C tal que $A \subseteq C$ e $B \subseteq \mathbb{N} - C$. Caso contrário, A e B são *computacionalmente inseparáveis*. A aritmética de Robinson é a aritmética de Peano com o axioma $\forall x(x \approx 0 \vee \exists y(x \approx y + 1))$ no lugar do princípio de indução. As fórmulas Δ_0 da linguagem aritmética são as fórmulas com quantificadores limitados, Σ_1-fórmulas são fórmulas da linguagem aritmética da forma $\exists x \phi$ com $\phi \in \Delta_0$ e Π_1-fórmulas são fórmulas da linguagem aritmética da forma $\forall x \psi$ com $\psi \in \Delta_0$. O seguinte lema é folclore na literatura sobre modelos não-padrão da aritmética.

Lema 4.3.1. *Para todo* $A \subseteq \mathbb{N}$*, existe um modelo não-padrão enumerável* $\mathcal{M}$ *da aritmética de Robinson* Q *e um número não-padrão* $a \in dom(\mathcal{M})$ *tal que* a *codfica* A*, i.e.,* $A = \{n \in \mathbb{N} : \mathfrak{M} \vDash \exists x(p_n \times x = a)\}$*, onde* p_n *é o* n*-ésimo número primo.*

Demonstração. Seja $A \subseteq \mathbb{N}$. Considere a teoria $Q^c = Q \cup \{n \prec c : n \in \mathbb{N}\} \cup \{\exists x(p_k \times x \approx c) : k \in A\} \cup \{\neg\exists y(p_k \times y \approx c) : k \notin A\}$. Cada subconjunto finito de Q^c é um subconjunto de $Q^l = Q \cup \{n \prec c : n < l\} \cup \{\exists x(p_k \times x \approx c) : k \in A \wedge k < l\} \cup \{\neg\exists y(p_k \times y \approx c) : k \notin A \wedge k < l\}$ para um $l \in \mathbb{N}$ apropriado. Seja q um número primo em $\mathbb{N}$ com $q > p_l$, e seja $r = q \times \prod\{p_k : k \in A e k < l\}$. Podemos verificar que uma estrutura $\mathcal{M}^*$ para a linguagem de Q^c cujo reduto é modelo de Q e tal que $c^{\mathcal{M}^*} = r$ é um modelo de Q^l. Assim, cada subconjunto finito de Q^c tem um modelo e, por compacidade, Q^c também tem um modelo, digamos

$\mathcal{M}$. Mais especificamente, digamos que $\mathcal{M}$ seja tal que $c^{\mathcal{M}} = a$. Dessa forma, a codifica o conjunto A, pois $A = \{n \in \mathbb{N} : \mathcal{M} \vDash \exists x(p_n \times x = a)\}$. Por fim, por Lowenheim-Skolem descendente, podemos considerar uma substrutura elementar enumerável de $\mathcal{M}$ na linguagem de Q^c. □

Teorema 4.3.1 (Teorema local de Tennenbaum)**.** *Existe um modelo não-padrão enumerável da aritmética de Robinson que não é computável.*

Demonstração. Segundo o teorema de Rosser da inseparabilidade [Rosser(1936)], existem conjuntos de números naturais padrão disjuntos que são computacionalmente enumeráveis e inseparáveis; digamos que sejam $A \subseteq \mathbb{N}$ e $B \subseteq \mathbb{N}$. Como esses conjuntos são computacionalmente enumeráveis, devem existir funções parciais computáveis f_A e f_B tais que $Dom(f_A) = A$ e $Dom(f_B) = B$. Digamos que M^A e M^B são as máquinas de Turing que computam, respectivamente, f_A e f_B. Considere a máquina $M^{AB} = M^A \cup M^B$, que pode ser não-determinista mas está bem definida porquanto $A \cap B = \varnothing$. De fato, em virtude de $A \cap B = \varnothing$, $\mathcal{A}^n_{M^{AB}} \vDash \neg((\phi_A(n) \to \psi_A(y)) \wedge (\phi_B(n) \to \psi_B(y)))$, onde ϕ_A e ϕ_B são as fórmulas de $\mathcal{T}^{\vec{s}}_{M^{AB}}$ correspondentes, respectivamente, às configurações iniciais de computações de M^A e M^B para a entrada n, ao passo que $\psi_A(y)$ e $\psi_B(y)$ são as fórmulas de $\mathcal{T}^{\vec{s}}_{M^{AB}}$ correspondentes, respectivamente, às configurações finais das computações de M^A e M^B com configurações iniciais ϕ_A e ϕ_B, sendo y qualquer instante de M^A e M^B no qual ambas as máquinas já pararam.

Considere $\mathcal{B}^n_{M^{AB}}$ um modelo não-padrão enumerável de $\mathcal{T}^n_{M^{AB}}$. Uma vez que $dom(\mathcal{A}^n_{M^{AB}}) = \mathbb{N}$ é um corte próprio de $\mathcal{B}^n_{M^{AB}}$, por transbordamento computacional, para cada $a > \mathbb{N}$, existe b com $a > b > \mathbb{N}$ tal que $\mathcal{B}^n_{M^{AB}} \vDash \neg((\phi_A(n) \to \psi_A(b)) \wedge (\phi_B(n) \to \psi_B(b)))$. Defina $C = \{n \in \mathbb{N} : M^A[n] \downarrow$ em tempo $x \leq a \in dom(\mathcal{B}^n_{M^{AB}})\}$. Claramente, $C \supseteq A$, pois se $n \in A$ então $\mathcal{A}^n_{M^{AB}} \vDash \phi_A(n) \to \psi_A(x)$ para algum $x \in dom(\mathcal{A}^n_{M^{AB}})$ e, por conseguinte, $\mathcal{B}^n_{M^{AB}} \vDash \exists x \leqslant a(\phi_A(n) \to \psi_A(x))$. Além disso, $C \cap B = \varnothing$, porque se $n \in B$ então $\mathcal{A}^n_{M^{AB}} \vDash \phi_B(n) \to \psi_B(y)$ para algum $y \in dom(\mathcal{A}^n_{M^{AB}})$ e, como no caso anterior, $\mathcal{A}^n_{M^{AB}} \vDash \exists y \leqslant a(\phi_B(n) \to \psi_B(y))$, o que implica $\mathcal{B}^n_{M^{AB}} \vDash \exists x \leqslant a\neg(\phi_A(n) \to \psi_A(x))$

devido a escolha de a. Dessa forma, concluímos que C é um conjunto tal que $A \subseteq C$ e $B \subseteq \mathbb{N} - C$.

Pelo lema 4.3.1, existe um modelo não-padrão enumerável $\mathcal{M}$ da aritmética de Robinson $\mathbf{Q}$ tal que $c \in dom(\mathcal{M})$ e

$$C = \{n \in \mathbb{N} : \mathcal{M} \vDash \exists z(\overbrace{z + \cdots + z}^{p_n} = c)\},$$

onde p_n é o n-ésimo número primo em $\mathcal{M}$. Considere $\mathcal{M}$ como sendo o modelo não-padrão da aritmética relativamente ao qual M^{AB} está definida. Desse modo, podemos considerar que $dom(\mathcal{B}^n_{M^{AB}}) = dom(\mathcal{M})$, pois basta construir $\mathcal{B}^n_{M^{AB}}$ adicionando as mesmas constantes usadas na construção de $\mathcal{M}$. Suponha que $\mathcal{M}$ é computável; mostraremos que essa hipótese implica que C é um conjunto computável, o que é um absurdo em virtude de A e B serem computacionalmente inseparáveis.

Como $\oplus$ em $\mathcal{M}$ é computável, existe uma máquina de Turing $M^{\oplus}$ que computa $\oplus$. Ora, o conjunto dos números primos é computável e, em particular, existe uma máquina de Turing M^P que o lista. Em virtude de $a \in dom(\mathcal{M})$, podemos decidir C através da máquina de Turing M^C com o seguinte programa:

1. Dado $n \in \mathbb{N}$, M^C chama M^A e imprime o k_n-ésimo elemento de A.
2. Dado o número k_n, M^C chama M^P e imprime o p_{k_n}-ésimo número primo em $dom(\mathcal{M})$.
3. Se existe $b \leq a$ em $dom(\mathcal{M})$ com $b\overbrace{\oplus \cdots \oplus}^{p_{k_n}}b = c$, então M^C retorna n.
4. Se não existe $b \leq a$ em $dom(\mathcal{M})$ com $b\overbrace{\oplus \cdots \oplus}^{p_{k_n}}b = c$, então M^C chama M^B e imprime o k_n-ésimo elemento de B.

Como $C = \{n \in \mathbb{N} : M^A[n] \downarrow \text{ em tempo } x \leq a \in dom(\mathcal{B}^n_{M^{A,B}})\}$, a cláusula (3) garante que M^C computacionalmente enumera C. Uma vez que $B \subseteq \mathbb{N} - C$,

a clásula (4) assegura que M^C computacionalmente enumera $\mathbb{N} - C$. Pelo teorema da complementação [Odifreddi(1992a), p.140], isso implica que M^C decide o conjunto C, o que contraria a definição de C. Uma vez que essa contradição tem como hipótese a existência de $M^{\oplus}$, concluímos que de fato $\oplus$ não pode ser computável. Com um argumento similar, podemos demonstrar que $\otimes$ também não é computável. □

Lembrando que $Q + I\Delta_0$ é a aritmética de Robinson Q mais o princípio de indução para fórmulas Δ_0, a seguir demonstramos o resultado de A. McAloon [McAloon(1982)], mas agora usando máquinas de Turing.

Teorema 4.3.2 (Teorema global de Tennenbaum)**.** *Todo modelo não-padrão da aritmética $Q + I\Delta_0$ não é computável.*

Demonstração. Seja $\mathcal{M}$ um modelo não-padrão da aritmética $Q + I\Delta_0$. Defina as máquinas de Turing M^A e M^B associadas aos conjuntos de números naturais padrão disjuntos que são computacionalmente enumeráveis e inseparáveis A e B como no teorema 4.3.1. Uma vez que máquinas de Turing são definíveis por Σ_1-fórmulas da aritmética, devem existir Δ_0-fórmulas $\phi(x, y)$ e $\psi(x, z)$ tais que $n \in A$ se, e somente se, $\mathcal{N} \vDash \exists y\phi(n, y)$ e $m \in B$ se, e somente se, $\mathcal{N} \vDash \exists z\phi(m, z)$. Em virtude de $A \cap B = \varnothing$ bem como a aritmética $Q + I\Delta_0$ ser Δ_0-completa, $\mathcal{M} \vDash \forall x\forall y\neg(\phi(x, y) \wedge \psi(x, z))$ para todo $k \in \mathbb{N}$. Por Δ_0-transbordamento de $Q + I\Delta_0$ no corte próprio $\mathbb{N}$, existe $a > \mathbb{N}$ em $dom(\mathcal{M})$ tal que $\mathcal{M} \vDash \forall x\forall y\forall z \leqslant a\neg(\phi(x, y) \wedge \psi(x, z))$.

Dado que $A \subseteq \mathbb{N}$ e $B \subseteq \mathbb{N}$ e $\mathcal{N}$ é um segmento inicial de todo modelo não-padrão de $Q + I\Delta_0$, podemos considerar A e B como subconjuntos de $dom(\mathcal{M})$. Agora digamos que M^A e M^B computam sobre $dom(\mathcal{M})$. Então o elemento $a \in dom(\mathcal{M})$ do parágrafo anterior é tal que $\mathcal{A}^n_{M^{AB}} \vDash \forall y \leqslant a\neg((\phi_A(n) \to \psi_A(y)) \wedge (\phi_B(n) \to \psi_B(y)))$, onde $\mathcal{A}^n_{M^{AB}}$, ϕ_A, ψ_A, ϕ_B e ψ_B são como no teorema 4.3.1. Ademais, considere o conjunto C da demonstração do teorema 4.3.1. Dessa forma, o resto do argumento é igual ao apresentado no teorema 4.3.1. □

Essa diferença entre versões local e global do teorema de Tennenbaum não existe na literatura especializada nesse tema. Ela foi possível de ser estabelecida porque adotamos uma estratégia de demonstração baseada em máquinas de Turing. Note que o teorema de Tennenbaum local vale para a aritmética de Robinson. Ademais, o limite Δ_0 no teorema global de Tennenbaum está em conformidade com o resultado de Shepherdson citado acima, segundo o qual existem modelos não-padrão computáveis de teorias aritméticas com o princípio de indução restrito às fórmulas abertas. Talvez esse tipo de distinção possa contribuir para a compreensão dos limites do teorema de Tennenbaum.

4.4 Funções numéricas em modelos aritméticos não-padrão

Apesar do teorema de Tennenbaum mostrar que as funções numéricas básicas (adição e multiplicação) não são computáveis em certos modelos aritméticos não-padrão, existem outras funções e relações que são computáveis em modelos não-padrão da aritmética; por exemplo, a relação de ordem e as funções sucessor e predecessor truncada são computáveis[6]. Supondo que a tese de Church-Turing seja verdadeira, deve haver máquinas de Turing que computam tais funções. Como é possível uma máquina de Turing computar uma função numérica definida sobre números naturais de um modelo não-padrão da aritmética? Os próximos resultados apresentam uma resposta a essa pergunta.

Lema 4.4.1. *Seja M uma máquina de Turing que computa uma função numérica n-ária f com $n > 0$ sobre números naturais de um modelo não-padrão da aritmética $\mathcal{M}$. Desse modo, se $f(a_1, \ldots, a_n) = b$, onde algum a_i ou b é um número não-padrão de $\mathcal{M}$, então existe uma configuração de $C_M^{a_1,\ldots,a_n}$ com celas não-padrão.*

[6] Cf. [D'Aquino(1997)]

Demonstração. Seja $\mathcal{M}$ um modelo não-padrão da aritmética, a_1 e b elementos não-padrão em $dom(\mathcal{M})$. Suponha que f é uma função unária - a demonstração é análoga no caso geral. Então, a representação binária de a_1 tem comprimento $\min\{x \in dom(\mathcal{M}) : \log_2(a_1 + 1) \leq x\}$, ou seja, $|a_1| = \min\{x \in dom(\mathcal{M}) : \log_2(a_1 + 1) \leq x\}$. Como $|a_1| > n$ para todo número $n \in \mathbb{N}$, segue-se que a configuração inicial da computação $C_M^{a_1}$ deve conter pelos menos a cela não-padrão $|a_1|$, i.e., $C_M^{a_1}(0) = (\oslash, 1, (2, 2, s_1, \ldots, s_{|a_1|}, 2, 2))$ onde cada $s_i \in \{0, 1\}$ e $s_1, \ldots, s_{|a_1|}$ é a representação binária de a_1. Pela mesma razão, o resultado também é válido no caso em que b é um número não-padrão. Considere $r_1, \ldots, r_m$ a representação binária de b. Como $f(a_1, \ldots, a_n) = b$, deve existir uma computação $C_M^{a_1}$ de M tal que sua configuração final é $C_M^{a_1}(i) = ((s'_1, \ldots, s'_k), 0, (2, 2, r_1, \ldots, r_m, 2, 2, r'_{m+3}, \ldots, r'_{m+l}))$ onde $m = |b|$ é não-padrão. □

Lema 4.4.2. *Seja M uma máquina de Turing que computa uma função numérica n-ária f com $n > 1$ sobre números naturais de um modelo não-padrão da aritmética $\mathcal{M}$. Desse modo, se $f(a_1, \ldots, a_n) = b$, onde algum a_i ou b é um número não-padrão de $\mathcal{M}$, então existe uma configuração de $C_M^{a_1,\ldots,a_n}$ com instantes não-padrão.*

Demonstração. Seja $\mathcal{M}$ um modelo não-padrão da aritmética, a_1, a_2 e b elementos não-padrão em $dom(\mathcal{M})$. Suponha que f é uma função binária - a demonstração é análoga no caso geral. Como na proposição anterior, $|a_1| = \min\{x \in dom(\mathcal{M}) : \log_2(a_1 + 1) \leq x\}$ e $|a_2| = \min\{x \in dom(\mathcal{M}) : \log_2(a_2 + 1) \leq x\}$. Como f é uma função binária, em algum momento M terá que percorrer todos os símbolos as representações binárias a_1 e a_2 de a_1 e a_2 a fim de gerar como saída a representação binária de b. Isso significa que M precisará de pelo menos $|a_1| + |a_2|$ instantes para ler a_1 e a_2 e, posteriormente, escrever b. Sabemos que $|a_1|, |a_2| > n$ para todo $n \in \mathbb{N}$. Ora, se n é um instante padrão, então $n + 1$ também será um instante padrão. Portanto, haverá uma configuração de $C_M^{a_1,a_2}$ com instantes não-padrão. O caso para b não-padrão é similar ao lema 4.4.1, pois se M imprime b, ocupará uma cela não-padrão

e, conseqüentemente, um instante não-padrão. □

A partir desses lemas podemos derivar uma conclusão geral acerca da ocorrência de celas e instantes não-padrão em computações de máquinas de Turing.

Teorema 4.4.1. *Seja $\mathcal{M}$ um modelo não-padrão da aritmética. Considere M uma máquina de Turing que precisa percorrer toda a entrada $\vec{s}$ para imprimir a saída $\vec{r}$ e $\vec{s}$ ou $\vec{r}$ são representações binárias de números não-padrão em $dom(\mathcal{M})$. Então apenas internamente em $\mathcal{M}$ a teoria $\mathcal{T}_M^{\vec{s}}$ está bem definida.*

Demonstração. Pelos teoremas da correspondência 2.3.1 e completude 2.5.2, se uma configuração $C_M^{\vec{s}}(i)$ de M tem um elemento não-padrão a, então existe uma fórmula correspondente ϕ a $C_M^{\vec{s}}(i)$ na qual o nome de a ocorre em ϕ e existe uma demonstração $\psi_0, \ldots, \psi_n$ tal que $\psi_n = \phi$. Se a é uma cela não-padrão, $a > n$ para todo $n \in \mathbb{N}$ e, pelo teorema da correspondência, ϕ deve ser, externamente à $\mathcal{M}$, infinita; o que é impossível, porque a linguagem de Turing é finitária. Se a é um instante não-padrão, novamente $a > n$ para todo $n \in \mathbb{N}$. Pelo teorema da representação, existe uma demonstração $\psi_0, \ldots, \psi_n = \phi$, onde ψ_0 corresponde à configuração inicial de $C_M^{\vec{s}}(i)$. Como a é um instante não-padrão e ocorre em ϕ, essa demonstração $\psi_0, \ldots, \psi_n = \phi$ deve ser, externamente à $\mathcal{M}$, infinita; o que também não pode ser o caso, porque demonstrações em teorias de Turing são finitárias. Ora, os lemas 4.4.1 e 4.4.2 mostram que, ao computar sobre números não-padrão, M terá uma configuração de $C_M^{\vec{s}}$ com celas e instantes não-padrão. Portanto, se M computa uma função numérica sobre um modelo aritmético não-padrão $\mathcal{M}$, $\mathcal{T}_M^{\vec{s}}$ deve ser considerada como um objeto definido internamente em $\mathcal{M}$. □

Nesse sentido cabe investigar outro pressuposto da teoria da computabilidade, a saber: o fato óbvio de que todo número natural é computável. Essa obviedade pressupõe os números naturais padrão como objetos intencionados da computabilidade. Diante dos resultados obtidos até aqui, cabe analisar

se em realidade essa suposição não esconde uma distinção importante entre números naturais padrão e não-padrão.

O primeiro a tratar dos aspectos computacionais dos números foi o próprio Turing [Turing(1936)]. Turing estava interessado em saber quais números reais eram computáveis, ou seja, ele queria saber quais seqüências de dígitos poderiam ser reproduzida por máquinas de Turing. Assim, Turing definiu que um número real é *computável* quando existe uma máquina de Turing M tal que, para cada número natural $n \geq 1$ como entrada, M produz o n-ésimo dígito da expansão decimal do número real como sua saída. A seguinte definição formaliza essa intuição no contexto dos números naturais.

Definição 4.4.1. *Um número natural (padrão ou não) n é* Turing-computável *quando existe uma máquina de Turing M tal que $M[2] = \underline{n}$.*

Teorema 4.4.2. *Se M é uma máquina de Turing que computa um número natural não-padrão a de um modelo aritmético não-padrão $\mathcal{M}$, então M é, externamente, um conjunto infinito.*

Demonstração. Primeiramente, observe que se M computa um número natural n, então M tem pelo menos $\min\{x \in dom(\mathcal{M}) : \log_2(n+1) \leq x\}$ estados. De fato, suponha que M computa um número natural n. Sabemos que a representação binária $\vec{r}$ de n tem comprimento $|n| = \min\{x \in dom(\mathcal{M}) : \log_2(n+1) \leq x\}$. Assim, como M computa n, a configuração final de C_M^2 é $C_M^2(t) = ((s_1, \ldots, s_k), 0, (2, 2, r_1, \ldots, r_m, 2, 2, r'_{m+3}, \ldots, r'_{m+l}))$ onde $m = |n|$ e cada $r_i \in \{0, 1\}$. Por definição, a configuração inicial de C_M^2 é $C_M^2(0) = (\oslash, 1, (2, 2, 2, 2, 2))$, ou seja, M deve imprimir toda sequência de símbolos $r_1, \ldots, r_m$ em sua fita. Uma vez que M imprime $r_1, \ldots, r_m$, ao passar de r_i para r_{i+1}, a máquina M deve fazer uma mudança de estado, pois, caso contrário, M não poderia determinar quantos símbolos ela já imprimiu. Isso mostra que se M computa um número natural n, então M tem pelo menos $\min\{x \in dom(\mathcal{M}) : \log_2(n+1) \leq x\}$ estados.

Agora suponha que M é uma máquina de Turing que computa um número natural não-padrão a de um modelo aritmético não-padrão $\mathcal{M}$. Então $M[2] =$

$\underline{a}$ e, pelo que acabamos de demonstrar, M tem pelo menos $|a| = \min\{x \in dom(\mathcal{M}) : \log_2(a+1) \leq x\}$ estados. Externamente, sabemos que $|a|$ é maior do que qualquer número natural $n \in \mathbb{N}$, o que significa que, externamente, M tem uma quantidade infinita de estados. Logo, M deve ter uma quantidade infinita de instruções, i.e., M é externamente infinita. □

Corolário 4.4.1. *Toda máquina de Turing que computa um número natural não-padrão tem um código numérico não-padrão.*

Demonstração. Seja a um número natural não-padrão de um modelo aritmético não-padrão $\mathcal{M}$. Pelo teorema 4.4.2, uma máquina de Turing M que computa a deve ter pelo menos $\min\{x \in dom(\mathcal{M}) : \log_2(n+1) \leq x\}$ estados e será, externamente, infinita. Disso, segue-se que, em qualquer codificação numérica das máquinas de Turing, um número não-padrão $b \geq \min\{x \in dom(\mathcal{M}) : \log_2(n+1) \leq x\}$ terá que ocorrer na composição do código $\ulcorner M \urcorner$ de M, i.e., $\ulcorner M \urcorner \geq b$. Logo, $\ulcorner M \urcorner$ é um código númérico não-padrão. □

Observação 4.4.1. *Tanto na* análise não-padrão *de A. Robinson [Robinson(1966)] quanto na* teoria interna dos conjuntos *de E. Nelson [Nelson(1977)], conjuntos externamente infinitos podem ser considerados internamente finitos. Ora, externamente máquinas de Turing devem ser objetos finitos. Assim, o teorema 4.4.2 afirma que qualquer número natural não-padrão a de um modelo aritmético não-padrão $\mathcal{M}$ somente é computável internamente em $\mathcal{M}$, ou seja, máquinas de Turing computam internamente em um modelo aritmético*[7]*.*

Os resultados desta seção conjuntamente são esclarecedores. Primeiro, eles apresentam uma resposta ao problema de saber como é possível que máquinas de Turing computem sobre números naturais não-padrão. Segundo, tais resultados mostram que considerar computações dessa natureza apenas faz sentido se as concebermos internamente em algum modelo

[7] Seria interessante avaliar qual é a relação entre números naturais não-padrão e o número aleatório de G. Chaitin [Chaitin(2006)], porque externamente tais números também são aleatórios ainda que internamente não.

aritmético não-padrão. Dessa forma, os teoremas 4.4.1 e 4.4.2 afirmam conjuntamente a existência do seguinte princípio:

Princípio de internalidade aritmética : Máquinas, estruturas e teorias de Turing são objetos internos de algum modelo da aritmética.

Na próxima seção veremos, em especial, que a consideração de modelos não-padrão da aritmética é inevitável, mesmo quando restringimos a computabilidade clássica às funções numéricas padrão.

4.5 A tese de Church-Turing no contexto da computabilidade não-padrão

Para Kreisel [Kreisel(1965b)], um dos principais problemas em aberto da lógica contemporânea é apresentar um conjunto de axiomas naturais e indubitáveis sobre o conceito de computabilidade que permita-nos demonstrar as teses de Church e Turing. Recentemente, Dershowitz e Gurevich [Dershowitz and Gurevich(2008)] defenderam uma abordagem da computabilidade em termos de *máquinas de estados abstratos* - uma espécie de generalização do conceito de máquina de Turing proposta por Gurevich [Gurevich(2000)] - e postularam quatro axiomas sobre algoritmos a fim de resolver esse problema. Os três primeiros são chamados *postulados seqüenciais*:

P1 Um algoritmo é um sistema de transição de estados configuracionais e suas transições são funções parciais.

P2 Os estados configuracionais são estruturas de primeira ordem que compartilham uma assinatura finita e são idênticas a menos de isomorfismo. As transições preservam o domínio dos estados configuracionais e, ademais, transições e isomorfismos comutam.

P3 As transições são determinadas por um conjunto fixo e finito de termos fechados sobre uma assinatura, de modo que os estados configuracionais concordam sobre os valores desses termos e também sobre todos os passos da sequência computacional[8].

Esses postulados expressam conjuntamente que *algoritmos são sistemas de transição determinísticos e limitados nos quais os estados configuracionais são estruturas de primeira ordem.* O quarto postulado tem duas formas:

P4A Estados configuracionais iniciais são aritméticos ou brancos. Sob isomorfismo, os estados configuracionais iniciais compartilham as mesmas operações estáticas e existe exatamente um estado configuracional inicial para cada entrada.

P4B Existe uma encodificação ρ cuja restrição para os números naturais é computável e através da qual todas as operações estáticas sobre estados configuracionais iniciais são ρ-computáveis.

Os postulados *P4A* e *P4B* garantem a efetividade aritmética dos algoritmos. *P4A* formaliza o conceito de *algoritmo numérico*, ou seja, algoritmos que aplicam operações aritméticas sobre os números naturais. Esse postulado foi suficiente para Dershowitz e Gurevich demonstrar a tese de Church, usando máquinas de estados abstratos. Embora a tese de Church seja extensionalmente equivalente a tese de Turing, para demonstrar a segunda é preciso considerar algoritmos que operam sobre domínios mais amplos do que os números naturais, porque máquinas de Turing computam sobre símbolos arbitrários. Por isso, *P4B* foi proposto por Dershowitz e Gurevich, ele afirma que os estados configuracionais devem ser computacionalmente codificáveis. Noutras palavras, ao apresentarem o postulado *P4A* Dershowitz e Gurevich procuraram transferir para as máquinas de estados abstratos todas as restrições finitárias que estabelecemos ao definir as máquinas de Turing. No

[8] Cf. [Dershowitz and Gurevich(2008), p.310-324] para uma explicação detalhada do significado desses postulados

entanto, o próximo resultado indica que essas restrições não impedem o surgimento de elementos não-padrão em computações das máquinas de Turing, nem mesmo no âmbito da computabilidade clássica padrão.

Teorema 4.5.1. *Seja M uma máquina de Turing que computa uma função sobre números naturais padrão. Dessa forma, $M[\vec{s}]\uparrow$ se, e somente se, existe uma configuração $C_M^{\vec{s}}(b)$ onde b é um instante não-padrão.*

Demonstração. Suponha que $M[\vec{s}]\uparrow$. Seja $\mathcal{A}_M^{\vec{s}}$ o modelo padrão de $\mathcal{T}_M^{\vec{s}}$. Em virtude de $M[\vec{s}]\uparrow$, temos que $\mathcal{A}_M^{\vec{s}} \vDash \forall x\exists y(x < y \wedge \exists v\exists z(\neg v \approx 0 \wedge H(v,z,y)))$. Pela completude, $\mathcal{T}_M^{\vec{s}} \vdash \forall x\exists y(x < y \wedge \exists v\exists z(\neg v \approx 0 \wedge H(v,z,y)))$. Agora defina a extensão $\mathcal{T}_M^{[a]\vec{s}}$ de $\mathcal{T}_M^{\vec{s}}$ conforme a proposição 4.2.1. Considere $\mathcal{B}_M^{[a]\vec{s}}$ um modelo de $\mathcal{T}_M^{[a]\vec{s}}$. Desse modo, $\mathcal{B}_M^{[a]\vec{s}} \vDash \forall x\exists y(x < y \wedge \exists v\exists z(\neg v \approx 0 \wedge H(v,z,y)))$, ou seja, para todo $a \in dom(\mathcal{B}_M^{[a]\vec{s}})$, existe um $b > a$ tal que $\mathcal{B}_M^{[a]\vec{s}} \vDash a < b \wedge \exists v\exists z(\neg v \approx 0 \wedge H(v,z,b))$ e, em particular, essa sentença também é verdadeira quando a é um elemento não-padrão. Pelo teorema 4.2.1, segue-se que tal $b \in dom(\mathcal{B}_M^{[a]\vec{s}})$ deve ser um elemento não-padrão. Se $\mathcal{B}_M^{[a]\vec{s}} \vDash \exists v\exists z(\neg v \approx 0 \wedge H(v,z,b))$, então M está em algum estado $v \neq 0$, em alguma cela z, no instante b; o que significa que existe uma configuração $C_M^{\vec{s}}(b) = ((a_1,\ldots,a_k),v,(c_1,\ldots,c_m))$ onde b é um instante não-padrão e $a_1,\ldots,a_k,c_1,\ldots,c_m$ são símbolos. Logo, existe uma configuração $C_M^{\vec{s}}(b)$ onde b é um instante não-padrão.

Conversamente, suponha que exista uma configuração $C_M^{\vec{s}}(b)$ onde b é um instante não-padrão. M computa uma função sobre números naturais padrão e, por conseguinte, $\vec{s}$ é a representação binária de um número natural padrão. Sabemos que se, ao ler o símbolo r_i, a máquina M está no instante t padrão, então, ao ler o símbolo r_{i+1} no passo seguinte, M estará no instante padrão $t+1$. No entanto, por hipótese M tem uma configuração $C_M^{\vec{s}}(b)$ onde b é um instante não-padrão. Assim, pela definição de estrutura de Turing, deve existir uma estrutura $\mathcal{B}_M^{\vec{s}}$ com um instante não-padrão b em $dom(\mathcal{B}_M^{\vec{s}})$ associada à máquina de Turing M com entrada $\vec{s}$, o que implica por sua vez que existem infinitos $n \in dom(\mathcal{B}_M^{\vec{s}})$ com $n < b$ para os quais $\mathcal{B}_M^{\vec{s}} \vDash \exists v\exists z H(v,z,n)$, pois pelo menos todos os instantes padrão satisfazem tal

fórmula. Ora, $\mathcal{B}^{\vec{s}}_M \vDash \neg H(0,0,0)$, uma vez que a configuração inicial de M não tem o estado final. Suponha que $\mathcal{B}^{\vec{s}}_M \vDash H(w,x,t)$ onde t é um instante padrão. Como $\mathcal{B}^{\vec{s}}_M \vDash \exists v \exists z H(v,z,n)$ é verdadeira para uma quantidade infinita instantes padrão n, devemos ter que $w \neq 0$, pois, caso contrário, M teria parado em t, ou seja, não existiram infinitos n tais que $\mathcal{B}^{\vec{s}}_M \vDash \exists v \exists z H(v,z,n)$. Daí, $\mathcal{B}^{\vec{s}}_M \vDash \neg H(0,x,t)$. Ora, o instante $t+1$ também é padrão e, pela mesma razão, $\mathcal{B}^{\vec{s}}_M \vDash \neg H(0,x,t+1)$. Logo, por indução, concluímos que $\mathcal{B}^{\vec{s}}_M \vDash \forall y \neg H(0,x,y)$, o que significa que $M[\vec{s}] \uparrow$. □

Corolário 4.5.1. *Seja M uma máquina de Turing que computa uma função numérica sobre números naturais padrão. Dessa forma, $M[\vec{s}] \downarrow$ se, e somente se, não existe uma configuração de $C^{\vec{s}}_M(b)$ com b um instante não-padrão.*

Demonstração. Esse resultado é uma conseqüência lógica do teorema 4.5.1. □

Observação 4.5.1. *O corolário 4.5.1 afirma que o problema da parada é equivalente ao problema de saber se um instante não-padrão ocorrerá ou não em uma computação. Uma vez que o problema da parada é indecidível, tal problema também é indecidível. Logo, as restrições finitárias que delimitam o conceito de máquina de Turing não impedem,* a priori*, o surgimento de instantes não-padrão nas computações de uma máquina de Turing.*

O próximo resultado mostra sob que circunstâncias celas não-padrão também podem aparecer em computações de uma máquina de Turing que computa uma função numérica sobre números naturais padrão.

Teorema 4.5.2. *Seja M uma máquina de Turing que computa uma função numérica sobre números naturais padrão. Dessa forma, $C^{\vec{s}}_M$ ocupa uma parcela infinita da fita se, e somente se, existe uma configuração de $C^{\vec{s}}_M(i) = ((a_1,\ldots,a_k),q,(b_1\ldots,b_m))$, onde k ou m são celas não-padrão.*

Demonstração. Suponha que $C^{\vec{s}}_M$ ocupa uma parcela infinita da fita. Então, $\mathcal{A}^{\vec{s}}_M \vDash \forall x \exists y (x < y \wedge \exists v \exists z T(v,y,z))$ e, com um argumento similar ao usado no

teorema 4.2.1, concluiremos que $C_M^{\vec{s}}$ tem uma cela não-padrão. A conversa também pode ser obtida como no teorema 4.5.1. □

Corolário 4.5.2. *Seja M uma máquina de Turing que computa uma função numérica sobre números naturais padrão. Dessa forma, $C_M^{\vec{s}}$ ocupa uma parcela finita da fita se, e somente se, não existe uma configuração de $C_M^{\vec{s}}(i) = ((a_1, \ldots, a_k), q, (b_1 \ldots, b_m))$, onde k ou m são celas não-padrão.*

Demonstração. Esse resultado é uma consequência direta do teorema 4.5.2. □

Observação 4.5.2. *O teorema 4.5.2 não pode ser expresso em termos de computações que não param, porque o fato de uma computação não parar não implica que ela use uma porção infinita da fita; mas se ocupar uma porção infinita da fita, então certamente a computação não pára.*

Ora, o axioma $P4B$ foi proposto por Dershowitz e Gurevich para garantir que estados configuracionais de máquinas de estados abstratos e, em particular, configurações de máquinas de Turing sempre fossem codificáveis de forma computável. Para isso, eles pensaram que bastaria impor restrições sobre as configurações iniciais e operações de uma máquina de Turing. Ora, os teoremas 4.5.1 e 4.5.2 mostram que não podemos decidir quando ocorrerão instantes ou celas não-padrão em configurações de máquinas de Turing. Dessa forma, o seguinte resultado mostra que o axioma $P4B$ pode não ser verdadeiro.

Teorema 4.5.3. *Seja $\mathcal{M}$ um modelo não-padrão da aritmética de Peano. Fixe uma codificação numérica $\ulcorner\ \urcorner$ das configurações de máquinas de Turing tal que $\ulcorner\ \urcorner$ está definida em termos da adição ou multiplicação. Assim, se as máquinas de Turing em $\mathbf{Maq}$ computam em $\mathcal{M}$, então existem computações de máquinas de Turing em $\mathbf{Maq}$ que não são computacionalmente codificáveis.*

Demonstração. Pelos teoremas 4.5.1 e 4.5.2, se uma máquina de Turing $M \in Maq$ não pára ou ocupa uma parcela infinita da fita durante uma de suas computações, então existe uma computação de M com um instante ou cela não-padrão. Digamos que seja a computação $C_M^{\vec{s}}$. Disso decorre que o código numérico $\ulcorner C_M^{\vec{s}}(i) \urcorner$ de alguma configuração de $C_M^{\vec{s}}$ terá que ser um número $a = \ulcorner C_M^{\vec{s}}(i) \urcorner$ maior do que todos os números em $C_M^{\vec{s}}$, ou seja, a terá que ser um número natural não-padrão. Pelo teorema de Tennenbaum, a adição e multiplicação não são operações computáveis sobre os números naturais não-padrões da aritmética de Peano. Logo, $\ulcorner C_M^{\vec{s}}(i) \urcorner$ é obtido computacionalmente. □

Corolário 4.5.3. *Seja M uma máquina de Turing em Maq e ρ uma encodificação da configurações iniciais de M cuja restrição para os números naturais é computável e tal que todas operações de M são ρ-computáveis. Então existem modelos aritméticos nos quais algumas configurações de M podem não ser computacionalmente codificáveis.*

A tese de Turing é extensionalmente equivalente à tese de Church *módulo* codificação. Assim, aparentemente, o corolário 4.5.3 não invalida a demonstração de Dershowitz e Gurevich da tese de Church, porque ela pressupõe os números naturais padrão como dados. No entanto, uma vez que, segundo a observação 4.5.1, o problema de saber se um número não-padrão ocorrerá em uma computação é indecidível, porque deveríamos excluir os elementos não-padrão do escopo da computabilidade?

O teorema 4.5.1 mostra que o fato de máquinas de Turing computarem sobre seqüências de símbolos arbitrárias permite o surgimento de elementos não intencionados, conquanto a classe das funções computáveis não seja alterada. Ora, o axioma $P4B$ é falso apenas se aceitarmos que máquinas de Turing operam em modelos não-computáveis. Assim, aparentemente, não precisamos excluir os modelos aritméticos não-padrão como um todo, bastaria assumir que máquinas de Turing operam em um modelo computável. Todavia, qual seria a razão para pressupormos que máquinas de Turing operam apenas em

modelos aritméticos computáveis? Isso não é uma justificativa *ad hoc* ou até mesmo circular?

Por fim, também existe a possibilidade de definirmos codificações que não usem a adição e a multiplicação, a fim de salvaguardarmos a veracidade do axioma $P4B$. Contudo, que tipo de codificação numérica poderia evitar a adição e a multiplicação?

Dado o trabalho de Dershowitz e Gurevich, podemos dizer que apresentar uma resposta a essas perguntas é equivalente a resolver o problema original levantado por Church e Turing sobre a relação entre nosso conceito intuitivo de algoritmo e definições formais da computabilidade. Em especial, isso mostra que a distinção entre finitude e infinitude, presente na distinção entre números naturais padrão e não-padrão, está estritamente vinculada às características do conceito de computação. Mais do que isso, a consideração de modelos não-padrão da aritmética parece ser inevitável, mesmo quando restringimos a computabilidade clássica às funções numéricas padrão. Ao que tudo indica, nos resta, então, investigar o que tais modelos têm a nos mostrar sobre a computabilidade.

5. CARACTERIZAÇÕES MODELO-TEÓRICAS DO PROBLEMA *P* VERSUS *NP*

The answer, my friend, is blowing in the wind.
Bob Dylan

Este capítulo caracteriza em termos modelo-teóricos o problema ***P*** versus ***NP***, mostrando, assim, uma barreira aritmética para a possibilidade de solução desse problema. Na seção 5.1 definimos os principais conceitos relevantes para o propósito deste capítulo. Na seção 5.2, a partir de um trabalho prévio de J. Paris e C. Dimitracopoulos, apresentamos uma caracterização global do problema ***P*** versus ***NP***. Nas seções 5.3 e 5.4 formulamos uma caracterização local do mesmo problema, seguindo, nesse caso, os trabalhos prévios de A. Máté e C. Sureson. Na seção 5.5 mostramos que uma possível solução do problema ***P*** versus ***NP*** deve usar técnicas essencialmente diferentes das formuladas por J. Paris e L. Harrington.

5.1 O universo da complexidade computacional

A *teoria da complexidade* é um conjunto de resultados sobre o que torna alguns problemas computacionalmente difíceis e outros fáceis [Sipser(2006)]. A teoria da *complexidade computacional* investiga a complexidade dos problemas em termos do *tempo* e do *espaço* necessários para computar suas soluções. Como as máquinas de Turing são o modelo de computação mais simples, os problemas são analisados enquanto problemas solucionáveis por algoritmos implementados diretamente através de máqinas de Turing. Por

isso podemos considerar a complexidade computacional como uma disciplina da computabilidade [Odifreddi(1992b), Cooper(2004)].

Dada uma máquina de Turing M, a complexidade computacional de um problema solucionável por M é uma função do comprimento das palavras que são entradas de M, desconsiderando-se qualquer outro parâmetro. O método de *análise do pior caso* considera a maior complexidade computacional de todas as entradas de um dado comprimento. Por outro lado, o método de *análise por média dos casos* considera a média da complexidade computacional de todas as entradas de um dado comprimento. Os principais problemas em aberto na teoria da complexidade estão associados à análise do pior caso.

Definição 5.1.1. *Seja M uma máquina de Turing determinista total. A* complexidade temporal *de M é a função $t : \mathbb{N} \to \mathbb{N}$ tal que $t(n)$ é a quantidade máxima de passos que M usa em qualquer computação com entrada de comprimento n. A* complexidade espacial *de M é a função $s : \mathbb{N} \to \mathbb{N}$ tal que $s(n)$ é a quantidade máxima de celas que M usa, além da própria entrada, em qualquer computação com entrada de comprimento n. Quando $t(n)$ é a complexidade temporal de M, dizemos que M* processa em tempo $t(n)$. *Similarmente, quando $s(n)$ é a complexidade espacial de M, dizemos que M* processa em espaço $s(n)$.

Geralmente a medida exata da complexidade de um algoritmo é uma expressão complexa, por isso costumamos apenas estimá-la. A *análise assintótica* procura entender a complexidade de um algoritmo quando esse processa entradas grandes. Fazemos isso ao considerar somente os termos de maior ordem em uma expressão algébrica que mede a complexidade das computações de um algoritmo, desconsiderando tanto os coeficientes quanto qualquer termo de ordem menor. A análise assintótica é legitima porque o valor do termo de maior ordem domina os outros termos para entradas suficientemente grandes, e, por conseguinte, podemos desconsiderá-los.

Definição 5.1.2. *Sejam f e g duas funções dos números naturais $\mathbb{N}$ nos números reais positivos $\mathbb{R}^+$. Definimos que $f(n) = O(g(n))$ se existem números*

naturais c *e* n_0 *tais que, para todo número natural* $n \geq n_0$, $f(n) \leq cg(n)$. *Nesse caso,* $g(n)$ *é denominada um* limitante superior assintótico *para* $f(n)$, *ou mais brevemente,* $g(n)$ *um* limitante superior *para* $f(n)$.

Geralmente a complexidade computacional limita-se a máquinas de Turing com uma única fita e com acesso sequencial. Essa escolha não é arbitrária porque as diferentes variações nas máquinas de Turing relativas à quantidade de fitas e ao tipo de memória são todas *polinomialmente equivalentes*. Isso significa que uma máquina de Turing pode simular qualquer outra com mais fitas ou com outro tipo de memória com no máximo um acréscimo polinomial do tempo e espaço de processamento. Desse modo, a complexidade computacional concentra-se em aspectos da complexidade que não são afetados por diferenças polinomiais.

Definição 5.1.3. *Se* $t : \mathbb{N} \to \mathbb{N}$ *é uma função de complexidade temporal, então a* classe de complexidade temporal *de* $t(n)$ *é o conjunto* $TIME(t(n))$ *das linguagens* $L \subseteq \{0,1\}^*$ *decididas por máquinas de Turing que processam em tempo* $O(t(n))$. *Similarmente, se* $s : \mathbb{N} \to \mathbb{N}$ *é uma função de complexidade espacial, então a* classe de complexidade espacial *de* $s(n)$ *é o conjunto* $SPACE(s(n))$ *das linguagens* $L \subseteq \{0,1\}^*$ *que são decididas por máquinas de Turing que processam em espaço* $O(s(n))$.

As investigações em complexidade computacional constituem-se basicamente em classificações das linguagens em $TIME(t(n))$ e $SPACE(s(n))$. Os problemas fundamentais dizem respeito à relação entre classes decidíveis em $TIME(t(n))$ e $SPACE(s(n))$ por máquinas de Turing deterministas e não-deterministas, onde $t(n)$ e $s(n)$ são polinômios.

Definição 5.1.4. $DTIME(t(n))$ *é a menor subclasse de* $TIME(t(n))$ *que contém todas as linguagens decidíveis por máquinas de Turing deterministas e* $NTIME(t(n))$ *é a menor subclasse de* $TIME(t(n))$ *que contém todas as linguagens decidíveis por máquinas de Turing não-deterministas. Similarmente,*

$DSPACE(t(n))$ é a menor subclasse de $SPACE(t(n))$ que contém todas as linguagens decidíveis por máquinas de Turing deterministas e $NSPACE(t(n))$ é a menor subclasse de $SPACE(t(n))$ que contém todas as linguagens decidíveis por máquinas de Turing não-deterministas. As classes de complexidade fundamentais são as seguintes:

- $P = \bigcup_{k>0} DTIME(n^k)$
- $NP = \bigcup_{k>0} NTIME(n^k)$
- $PSPACE = \bigcup_{k>0} DSPACE(n^k)$
- $NPSPACE = \bigcup_{k>0} NSPASE(n^k)$

Noutras palavras, a classe P é a classe das linguagens que são decidíveis em tempo polinomial por máquinas de Turing deterministas, ao passo que NP é a classe análoga, mas com referência a máquinas de Turing não-deterministas. Analogamente para $PSPACE$ e $NPSPACE$, mas dessa vez considerando a complexidade espacial. Essas classes são importantes porque NP e $NPSPACE$ são simuláveis por máquinas de Turing deterministas, mas em princípio parece ser necessário um uso exponencial de tempo e espaço, respectivamente. Os problemas computáveis deterministicamente com complexidade polinomial parecem ser tratáveis, ao passo que os computáveis deterministicamente com complexidade exponencial são certamente intratáveis - nesse último caso, para entradas não muito grandes, os algoritmos precisam de tempo e espaço maior do que a quantidade de partículas no universo. Disso decorre o caráter fundamental dessas classes de complexidade.

Surpreendentemente, W. J. Savitch [Savitch(1970)] demonstrou que $PSPACE = NPSPACE$, usando o fato de que o espaço pode ser reutilizado. No entanto, a pergunta sobre a relação entre P e NP continua em aberto, sendo largamente considerado o principal problema da teoria da complexidade. No que se segue apresentaremos caracterizações desse problema, denotado por $P \stackrel{?}{=} NP$, em termos de modelos não-padrão da aritmética.

Ao fazermos isso, usaremos as técnicas desenvolvidas nesta tese acerca da computabilidade de Turing sobre números naturais não-padrão.

5.2 Uma caracterização global do problema $P \stackrel{?}{=} NP$

Analogamente à hierarquia aritmética de S. Kleene [Kleene(1943)], L. Stockmeyer [Stockmeyer(1976)] definiu uma hierarquia da complexidade temporal de problemas decidíveis.

Definição 5.2.1. *Para todo $n \in \mathbb{N}$, uma linguagem $L \subseteq \{0,1\}^*$* pertence à classe Σ_n^p *(n-ésimo nível da hierarquia polinomial) se existe uma máquina de Turing M que processa em tempo polinomial e um polinômio p tal que:*

$$w \in L \text{ se, e somente se,}$$
$$\exists w_1 \in \{0,1\}^{p(|w|)} \forall w_2 \in \{0,1\}^{p(|w|)} \cdots Q w_n \in \{0,1\}^{p(|w|)} M[w, w_1, w_2, \ldots, w_n] = 1,$$

onde Qw_i denota $\forall w_i$ ou $\exists w_i$ conforme i seja par ou ímpar ($1 \leq i \leq n$), respectivamente, e $\{0,1\}^{p(|w|)}$ é o subconjunto de $\{0,1\}^*$ composto por palavras com tamanho no máximo $p(|w|)$. Ademais, definimos que $\Pi_n^p = co\Sigma_n^p = \{\{0,1\}^* - L : L \in \Sigma_n^p\}$. Assim, a hierarquia polinomial *é a classe de linguagens $PH = \bigcup_{n\in\mathbb{N}} \Sigma_n^p$.*

Observe que $\Sigma_0^p = \Pi_0^p = P$, $NP = \Sigma_1^p$ e $coNP = \Pi_1^p$. Também podemos verificar que $\Sigma_n^p \subseteq \Sigma_{n+1}^p$, $\Pi_n^p \subseteq \Pi_{n+1}^p$, $\Sigma_n^p \subseteq \Pi_{n+1}^p \subseteq \Sigma_{n+2}^p$ e $\Pi_n^p \subseteq \Sigma_{n+1}^p \subseteq \Pi_{n+2}^p$, porque ao adicionarmos um novo quantificador e uma nova palavra, podemos desprezá-la - isso não altera a aceitação de uma palavra no nível inferior da hierarquia. Portanto, a hierarquia polinomial é cumulativa e, conseqüentemente, $PH = \bigcup_{n\in\mathbb{N}} \Pi_n^p$.

Problemas em níveis superiores da hierarquia PH parecem ter complexidade bastante maior do que aqueles em níveis inferiores. Ademais, se $P = NP$, então PH colapsa, i.e., existe um $k \in \mathbb{N}$ tal que $PH = \Sigma_n^p$, a saber, $k = 0$. Por isso, alguns teóricos da complexidade acreditam que essa é uma evidência de

que $P \neq NP$ [Sipser(2006), Arora and Barak(2009)]. Uma vez que o colapso de PH e o problema $P \stackrel{?}{=} NP$ estão estritamente vinculados, existem vários estudos comparativos da hierarquia PH com outras hierarquias[1]. Uma dessas hierarquias que tem suscitado interesse é a hierarquia aritmética limitada.

Definição 5.2.2. *A* hierarquia aritmética limitada $\Delta_0^{\mathbb{N}}$ *é definida do seguinte modo:*

1. $\Sigma_0^a = \Pi_0^a = \{X \subseteq \mathbb{N} : x \in X$ *se, e somente se,* $\mathcal{N} \vDash \phi(x)\}$*, onde* ϕ *é uma fórmula livre de quantificadores.*

2. $\Sigma_n^a = \{X \subseteq \mathbb{N} : x \in X$ *se, e somente se,* $\mathcal{N} \vDash \exists x_1 < \tau_1 \forall x_2 < \tau_2 \cdots Qx_n < \tau_n \phi(x, x_1, x_2, \ldots, x_n)\}$*, onde* Qx_i *denota* $\forall x_i$ *ou* $\exists x_i$ *conforme* i *seja par ou ímpar (*$1 \leq i \leq n$*), respectivamente.*

3. $\Pi_n^a = \{X \subseteq \mathbb{N} : x \in X$ *se, e somente se,* $\mathcal{N} \vDash \forall x_1 < \tau_1 \exists x_2 < \tau_2 \cdots Qx_n < \tau_n \phi(x, x_1, x_2, \ldots, x_n)\}$*, onde* Qx_i *denota* $\forall x_i$ *ou* $\exists x_i$ *conforme* i *seja impar ou par (*$1 \leq i \leq n$*), respectivamente.*

4. $\Delta_n^a = \Sigma_n^a \cap \Pi_n^a$ *e* $\Delta_0^{\mathbb{N}} = \bigcup_{n \in \mathbb{N}} \Delta_n^a$.

J. Paris e C. Dimitracopoulos [Paris and Dimitracopoulos(1982)] foram os primeiros a observar que a hierarquia $\Delta_0^{\mathbb{N}}$ está vinculada à definibilidade uniforme da teoria completa de um modelo aritmético não-padrão. Mais precisamente, considere dois números não-padrão a e b de um modelo $\mathcal{M}$ não-padrão enumerável da aritmética. Considerando a e b enquanto estruturas, dizemos que a teoria completa $Th(a)$ do número não-padrão a é *uniformemente definível* em b caso exista uma fórmula aritmética $Sat(x, y)$ tal que $b \vDash Sat(a, \ulcorner \phi \urcorner)$ se, e somente se, $a \vDash \phi$ para toda sentença ϕ da linguagem aritmética, onde Sat é o predicado de satisfatibilidade na aritmética[2]. H. Lessan [Lessan(1978)] mostrou que, para todo número não-padrão a, $Th(a)$ é

[1] Cf. [Arora and Barak(2009)] para referências.
[2] Cf. [Kaye(1991), p.104-129] para mais detalhes sobre Sat.

uniformemente definível em qualquer $b > 2^{a^{\mathbb{N}}}$, onde $2^{a^{\mathbb{N}}}$ denota 2^{a^k} para todo $k \in \mathbb{N}$. Doravante, usaremos a notação $x^{\mathbb{N}}$ nesse sentido. Dessa forma, Paris e Dimitracopoulos constataram que o colapso da hierarquia $\Delta_0^{\mathbb{N}}$ é equivalente a possibilidade de reduzir $2^{a^{\mathbb{N}}}$ para $a^{\mathbb{N}}$.

Dado que as computações de uma máquina de Turing não-determinista N sobre uma entrada x em tempo $|x|^k$ podem, para um x suficientemente grande, serem codificadas por um número menor do que $x^{|x|^{2k}}$, a hierarquia PH consiste precisamente dos subconjuntos X de $\mathbb{N}$ que podem ser escritos na forma $X = \{x \in \mathbb{N} : \mathcal{N} \vDash \exists \vec{x}_1 \leq x^{p_1(|x|)} \cdots \exists \vec{x}_n \leq x^{p_n(|x|)} \phi(\vec{x}_1, \ldots, \vec{x}_n, x)\}$, onde cada p_i é um polinômio e ϕ é uma fórmula aberta. Portanto, o colapso da hierarquia PH no nível 0 é equivalente à possibilidade de definir uniformemente $Th(a)$ em qualquer $b > a^{|a|^{\mathbb{N}}}$. Essa foi a principal conclusão de Paris e Dimitracopoulos em [Paris and Dimitracopoulos(1982)].

A fim de mostrar as potencialidade dos recursos modelo-aritméticos para o tratamento do colapso da hierarquia PH, Paris e Dimitracopoulos exploram a relação entre $Th(a)$, $Th(a^2)$, $Th(a^{|a|})$, $Th(2^a)$, etc. O problema consiste em saber se $Th(a)$ determina $Th(a^2)$ ou $Th(a^{|a|})$ ou talvez $Th(2^a)$, etc. Note que todos os modelos de $Th(a)$ são elementarmente equivalentes, posto que $Th(a)$ é uma teoria completa. Portanto, a pergunta sobre se $Th(a)$ determina $Th(a^2)$ é equivalente a perguntar se $a \equiv b$ implica $a^2 \equiv b^2$. Claramente, $Th(a)$ determina $Th(a^2)$, $Th(a^3)$, $Th(a^4)$, e assim por diante - porque $a^n = a \overbrace{\times \cdots \times}^{n \text{ vezes}} a$. Paris e Dimitracopoulos mostraram, todavia, que $Th(a)$ não determina $Th(2^a)$. A partir desse resultado eles formularam uma definição semelhante à seguinte.

Definição 5.2.3. *Seja $\mathcal{M}$ um modelo não-padrão enumerável da aritmética. A relação de* equivalência polinômial determinista *definida sobre $\mathcal{M}$ é a relação binária $\equiv_P$ tal que $a \equiv_P b$ se, e somente se, para toda máquina de Turing determinista M e todo $m \in \mathbb{N}$, $M[a] = 1$ em tempo $|a|^m$ se, e somente se, $M[b] = 1$ em tempo $|b|^m$. Similarmente, também definimos a relação de* equivalência polinômial não-determinista $\equiv_{NP}$, *a qual envolve máquinas de Turing não-deterministas.*

Observação 5.2.1. *O fato de a e b serem polinomialmente equivalentes não implica que a ou b está na classe P. A relação de equivalência polinômial entre a e b apenas indica que eles são indistinguíveis por máquinas de Turing deterministas que processam em tempo polinômial.*

De acordo com essa definição, os resultados de Paris e Dimitracopoulos mostram que $P = NP$ se, e somente se, $a \equiv_P b$ implica $a \equiv b$, para todos números naturais a e b em um modelo não-padrão enumerável da aritmética $\mathcal{M}$. A seguir apresentamos uma nova demonstração da caracterização de Paris e Dimitracopoulos do problema $P \stackrel{?}{=} NP$.

Teorema 5.2.1. *Seja $\mathcal{M}$ um modelo não-padrão enumerável da aritmética. Os seguinte enunciados são equivalentes:*

1. $P \neq NP$.

2. *Existem números naturais não-padrão $a, b \in \mathcal{M}$ tais que $a \equiv_P b$ e $a \not\equiv b$.*

Demonstração. Seja $\mathcal{M}$ um modelo não-padrão enumerável da aritmética. Suponha que $P = NP$. Então HP colapsa com a classe P e, conseqüentemente, $\Delta_0^{\mathbb{N}} \subseteq P$ posto que $\Delta_0^{\mathbb{N}} \subseteq HP$. Assim, se $a \equiv_P b$ para quaisquer $a, b \in dom(\mathcal{M})$, então a e b estão no mesmo Δ_0-subconjunto de $\mathcal{M}$ e, por conseguinte, $a \equiv_{\Delta_0} b$. Como a e b são $\equiv_{\Delta_0}$-equivalentes, eles codificam os mesmos conjuntos numéricos e, assim, $a \equiv b$.

Reciprocamente, suponha que, para todo a e b em $dom(\mathcal{M})$, $a \equiv_P b$ implica $a \equiv b$, mas $P \neq NP$. Como $P \neq NP$, então existe uma máquina de Turing não-determinista N que processa em tempo polinomial e discorda de qualquer máquina de Turing determinista que processam em tempo polinomial. Primeiramente, mostraremos, em especial, que existem a e b em $dom(\mathcal{M})$ tais que $a \equiv_P b$ e N discorda de todas as máquinas de Turing deterministas no que respeita às representações binárias $\underline{a}$ e $\underline{b}$ de a e b. Suponha, pelo contrário, que, para todos $a, b \in dom(\mathcal{M})$ tais que $a \equiv_P b$, N concorda com algumas máquinas de Turing deterministas $M_1, \ldots, M_n$ que aceitam $\underline{a}$ e $\underline{b}$ em tempo polinomial,

ou seja, no modelo $\mathcal{M}$ é verdade que se $M_1[a] = M_1[b], \ldots, M_n[a] = M_n[b]$ então $N[a] = N[b]$. Ora, isso também deve valer no modelo aritmético padrão $\mathcal{N}$. Assim, para decidir se N aceita o conjunto $C \subseteq \mathbb{N}$ basta computar com $M_1, \ldots, M_n$ e verificar se essas máquinas aceitam C. Mas nesse caso temos uma contradição, posto que, por hipótese, N é uma máquina que não aceita conjuntos na classe P. Demonstrado isso, digamos que N aceita $\underline{a}$ mas rejeita $\underline{b}$ e $a \equiv_P b$.

Em virtude de $N[a] = 1$ em tempo $|a|^m$ para algum $m \in \mathbb{N}$, pelo teorema da completude para máquinas não-deterministas, a teoria $\mathcal{T}_N^{\underline{a}}$ em $\mathcal{M}$ é tal que $\mathcal{T}_N^{\underline{a}} \vdash \phi_{\underline{a}}(0) \rightarrow \psi_{\underline{a}}(|a|^m)$, onde $\phi_{\underline{a}}(z)$ é a fórmula correspondente à configuração inicial de $C^{\underline{a}}$ e $\psi_{\underline{a}}(z)$ é a fórmula correspondente à configuração final de $C^{\underline{a}}$, sendo z um parâmetro para instantes. Por hipótese $a \equiv_P b$ implica $a \equiv b$, e, como estamos assumindo que $a \equiv_P b$, derivamos $a \equiv b$. Sabemos que todo modelo não-padrão enumerável da aritmética é computacionalmente saturado[3]. Os modelos a e b são enumeráveis bem como elementarmente equivalentes. Assim, pela teoria dos modelos[4], $a \cong b$. Conseqüentemente, a teoria $\mathcal{T}_N^{\underline{b}}$ em $\mathcal{M}$ é tal que $\mathcal{T}_N^{\underline{b}} \vdash \phi_{\underline{b}}(0) \rightarrow \psi_{\underline{b}}(|a|^m)$, onde $\phi_{\underline{b}}(z)$ e $\psi_{\underline{b}}(z)$ são como anteriormente mas agora considerando $C^{\underline{b}}$. Pelo teorema da corretude, isso significa que $N[b] = 1$; o que é uma contradição, porque havíamos suposto que $N[b] = 0$.

Logo, a partir da hipótese de que, para todos $a, b \in dom(\mathcal{M})$, $a \equiv_P b$ implica $a \equiv b$, derivamos que $P = NP$. □

O teorema 5.2.1 apresenta uma caracterização *global* do problema $P \stackrel{?}{=} NP$ porque ele não expressa as características locais dos modelos aritméticos que não são elementarmente equivalentes. A demonstração que formulamos combina aspectos da própria demonstração de Paris e Dimitracopoulos com as técnicas desenvolvidas nesta tese. O argumento de que (1) implica (2) é exatamente como o original; apenas acrescentamos alguns pormeno-

[3] Cf. [Kaye(1991)]
[4] Cf. [Hodges(2005)]

res. A inovação encontra-se na estratégia para mostrar que (2) implica (1). Aqui evitamos o uso da codificação das máquinas de Turing na linguagem da aritmética, explorando, por um lado, teorias de Turing e, por outro, a saturação computacional de modelos não-padrão enumeráveis da aritmética.

5.3 Uma caracterização local do problema $P \stackrel{?}{=} NP$ (Primeira parte)

A partir do teorema 5.2.1 de Paris e Dimitracopoulos, A. Máté [Máté(1990)] formulou uma heurística, baseada no problema *3-SAT*, para usar modelos aritméticos não-padrão no estudo do problema $P \stackrel{?}{=} NP$.

O problema *3-SAT* consiste em saber se uma fórmula proposicional em forma normal conjuntiva CNF com cláusulas de três literais é satisfatível. Pela decidibilidade da lógica proposicional, sabemos que esse problema é decidível. Mais do que isso, o procedimento usual de teste via tabelas-verdade é um algoritmo exponencial que decide o problema da satisfabilidade para qualquer tipo de fórmula proposicional; em especial, as fórmulas em CNF. Considerando alguma medida polinômial de fórmulas proposicionais (por exemplo, a quantidade de dígitos necessária para codificá-las como seqüências binárias), o teorema de Cook-Levin [Cook(1971), Levin(1973)] afirma que $P = NP$ se, e somente se, existe uma máquina de Turing determinista que decide *3-SAT* em tempo polinomial. Por isso, dizemos que o problema *3-SAT* é *NP-completo*: *3-SAT* está em NP e qualquer outro problema em NP pode ser reduzido a ele através de uma máquina de Turing determinista em tempo polinomial.

Agora considere ϕ uma fórmula proposicional em *3-SAT*. Sabemos que ϕ é satisfatível se, e somente se, o conjunto das cláusulas que ocorrem em ϕ é consistente. Seja $\mathcal{M}_1$ um modelo não-padrão enumerável da aritmética e C um subconjunto de $dom(\mathcal{M}_1)$ composto por números que codificam os conjuntivos que ocorrem em ϕ. Digamos que o número não-padrão $n \in dom(\mathcal{M}_1)$

é a cardinalidade de C internamente em $\mathcal{M}_1$. Suponha que C é um conjunto inconsistente em $\mathcal{M}_1$, i.e., não existe uma valoração em $\mathcal{M}_1$ das variáveis proposicionais que ocorrem nas fórmulas codificadas em C que tornem todas as fórmulas de C verdadeiras. Como $\mathcal{M}_1$ não codifica todos os conjuntos de números naturais (existe uma quantidade não-enumerável de tais conjuntos), pode ser que em alguma extensão $\mathcal{M}_2$ de $\mathcal{M}_1$ exista uma valoração v de C que a torne verdadeira, ou seja, em $\mathcal{M}_2$ o conjunto C seria consistente. Se alguém conseguir fazer isso de modo que $\mathcal{M}_1$ e $\mathcal{M}_2$ sejam modelos da aritmética iguais para todos números menores ou iguais a n, então $P \neq NP$.

Para verificar a conclusão do parágrafo acima, note que se existisse uma máquina de Turing determinista M que decidisse em tempo polinomial n^k, para algum $k \in \mathbb{N}$, se ϕ é ou não satisfatível, então M seria capaz de distinguir os modelos $\mathcal{M}_1$ e $\mathcal{M}_2$ até o número n^k. No entanto, a valoração v tanto em $\mathcal{M}_1$ quanto em $\mathcal{M}_2$ é uma função $v : n \to \{0, 1\}$, o que implica que o código de v é um número maior do que 2^n, ou seja, $\mathcal{M}_1$ e $\mathcal{M}_2$ são idênticos até n^k para qualquer número $k \in \mathbb{N}$. Logo, não existe uma maneira de decidir a satisfatibilidade de ϕ em tempo polinomial através de uma máquina de Turing determinista.

No que se segue apresentaremos uma formalização dessa heurística proposta por Máté [Máté(1990)], usando as técnicas desenvolvidas nesta tese acerca da computabilidade de Turing sobre números naturais não-padrão.

Para começar, a estrutura $\mathcal{M}_2$ mencionada acima não pode ser uma extensão própria da estrutura $\mathcal{M}_1$, devido ao teorema de Matijasevic-Robinson-Davis-Putnam (MRDP)[5]. Segundo esse teorema, qualquer Σ_1-fórmula ψ é equivalente no modelo padrão da aritmética $\mathcal{N}$ a uma equação diofantina, i.e.:

Para toda Σ_1-fórmula ψ da linguagem aritmética, existem polinômios P e Q

[5] Máté estava ciente disso [Máté(1990), p.179].

com coeficientes em $\mathbb{N}$ tais que

$$\mathcal{N} \vDash \forall x(\psi(x) \leftrightarrow \exists y P(y, x) \approx Q(y, x)).$$

Ora, a consistência de uma fórmula proposicional ϕ em *3-SAT* pode ser expressa através de uma Δ_0-fórmula $\psi(l)$ da aritmética, onde l é o código numérico de ϕ. Noutras palavras, a consistência do conjunto C mencionado acima pode ser expressa por uma Δ_0-fórmula aritmética $\psi(l)$ tanto na estrutura $\mathcal{M}_1$ quanto em $\mathcal{M}_2$. Aplicando o teorema MRDP à fórmula $\psi(l)$, temos que $\mathcal{M}_1 \vDash \psi(l)$ se, e somente se, $\mathcal{M}_2 \vDash \psi(l)$. Portanto, o valor de verdade de $\psi(l)$ não pode mudar ao passarmos de $\mathcal{M}_1$ para uma extensão $\mathcal{M}_2$. Por isso, Máté introduziu o seguinte conceito.

Definição 5.3.1 ([Máté(1990)])**.** *Sejam $\mathcal{M}_1$ e $\mathcal{M}_2$ dois modelos não-padrão enumeráveis da aritmética. Considere $n \in dom(\mathcal{M}_1)$. Assim, $\mathcal{M}_2$ é uma* extensão de $\mathcal{M}_1$ até n*, o que denotamos por $\mathcal{M}_1 \preceq_n \mathcal{M}_2$, quando $\{i \in \mathcal{M}_1 : i \leq^{\mathcal{M}_1} n\} \subseteq \mathcal{M}_2$ e, para quaisquer $i, j \leq^{\mathcal{M}_1} n$, $i +^{\mathcal{M}_1} j = i +^{\mathcal{M}_2} j$ e $i \times^{\mathcal{M}_1} j = i \times^{\mathcal{M}_2} j$. Quando $\mathcal{M}_1 \preceq_n \mathcal{M}_2$ para algum n, chamamos $\mathcal{M}_2$ de uma* extensão parcial *de $\mathcal{M}_1$. Ademais, dizemos que $\mathcal{M}_1$ e $\mathcal{M}_2$ são* isomórficos até n *quando $\mathcal{M}_1 \preceq_n \mathcal{M}_2$ e $\mathcal{M}_2 \preceq_n \mathcal{M}_1$; nesse caso escrevemos $\mathcal{M}_1 \simeq_n \mathcal{M}_2$.*

Esse conceito de extensão parcial de um modelo aritmético não-padrão está implicitamente presente no teorema 5.2.1 de Paris e Dimitracopoulos. De fato, se duas estruturas aritméticas $\mathcal{M}_1$ e $\mathcal{M}_2$ são idênticas de um ponto de vista de máquinas de Turing deterministas que computam em tempo polinomial, mas ainda assim $\mathcal{M}_1$ e $\mathcal{M}_2$ não são elementarmente equivalentes, só pode ser que uma delas é extensão parcial da outra.

Lema 5.3.1 ([Máté(1990)])**.** *Sejam $\mathcal{M}_1$ e $\mathcal{M}_2$ modelos não-padrão enumeráveis da aritmética e $n \in dom(\mathcal{M}_1) \cap dom(\mathcal{M}_2)$ um número não-padrão tal que $\mathcal{M}_1 \simeq_n \mathcal{M}_2$ e $\mathcal{M}_1 \preceq_{2^{n^{\mathbb{N}}}} \mathcal{M}_2$. Considere o elemento $b \in dom(\mathcal{M}_1)$ tal que $b < (2^{n^k})^{\mathcal{M}_1}$ para algum $k \in \mathbb{N}$, e suponha que b codifica internamente*

uma sequência $\vec{b}$ *de tamanho* $r \in dom(\mathcal{M}_1)$. *Então* b *também codifica internamente uma sequência* $\vec{b}$ *de comprimento* r *em* $\mathcal{M}_2$ *e* $\forall i \in \mathcal{M}_2(i <^{\mathcal{M}_1} r \rightarrow (\vec{b}(i)^{\mathcal{M}_1} = \vec{b}(i)^{\mathcal{M}_2}))$. *Assim, se* $\{i \in \mathcal{M}_2 : i \leq^{\mathcal{M}_2} r\} \subseteq \mathcal{M}_1$, *então* b *codifica a mesma sequência tanto em* $\mathcal{M}_1$ *quanto em* $\mathcal{M}_2$.

Demonstração. Esse resultado é uma consequência das propriedades da codificação numérica através da função de Gödel β e da função pareamento $\langle\ \rangle$.

A primeira é tal que $\beta : \mathbb{N}^3 \rightarrow \mathbb{N}$ sendo $\beta(c, d, i) = rm(c, ((i + 1)d) + 1)$, onde $rm(m, n)$ é resto de m divido por n. Essa função β é computável e, para qualquer seqüência finita de números naturais $(a_0, \ldots, a_n)$, existem números naturais c e d tais que $\beta(c, d, i) = a_i$ para $0 \leq i \leq n$ - trata-se de fato bastante conhecido que a existência desses números c e d depende do teorema do resto chinês. Assim, dizemos que a tripla de números naturais (c, d, n) *codifica* a seqüência $(a_0, \ldots, a_n)$.

Já a segunda é tal que $\langle\rangle : \mathbb{N}^2 \rightarrow \mathbb{N}$ sendo $\langle x, y\rangle = \frac{1}{2}(x+y)(x+y+1)+y$. Dada uma tripla (c, d, n) que codifica a seqüência $(a_0, \ldots, a_n)$, podemos apresentar um número b que codifica essa seqüência estabelecendo que $b = \langle\langle c, d\rangle, n\rangle$.

A propriedade importante nesse processo de codificação via β e $\langle\ \rangle$ é que, para qualquer seqüência $(a_0, \ldots, a_n)$ de números naturais, o código b dessa seqüência é tal que $b < 2^{s^4}$, onde s é um número natural arbitrário tal que $s \geq n$ e $s! \geq a_0, \ldots, a_n$. Dessa forma, concluímos que b codifica a mesma sequência tanto em $\mathcal{M}_1$ quanto em $\mathcal{M}_2$ porque com as funções β e $\langle\ \rangle$ podemos codificar cada componente de uma seqüência com valores menores do que $(2^{n^k})^{\mathcal{M}_1}$. □

Lema 5.3.2 ([Máté(1990)])**.** *Sejam* $\mathcal{M}_1$ *e* $\mathcal{M}_2$ *modelos não-padrão enumeráveis da aritmética e* $n \in dom(\mathcal{M}_1) \cap dom(\mathcal{M}_2)$ *um número não-padrão tal que* $\mathcal{M}_1 \simeq_n \mathcal{M}_2$ *e* $\mathcal{M}_1 \preceq_{2^{n^{\mathbb{N}}}} \mathcal{M}_2$. *Se* $\vec{b} = (b_i : i < r)$ *é uma sequência tal que* $\langle\vec{b}\rangle \in dom(\mathcal{M}_1)$ *sendo* $r < n^k$, *algum* $k \in \mathbb{N}$, *e* $b_i < 2^{n^k}$, *para todo* $i < r$, *então* $\langle\vec{b}\rangle \in dom(\mathcal{M}_2)$. *Em particular, para cada* $k \in \mathbb{N}$, *todo subconjunto de* n^k *codificado em* $\mathcal{M}_1$ *também é codificado em* $\mathcal{M}_2$.

Demonstração. Pelas propriedades da codificação usando as funções β e $\langle\ \rangle$, sabemos que o código da sequência $\vec{b}$ é menor do que $2^{n^{4k}}$. Como $r < n^k$, o resultado segue-se do lema 5.3.1. □

As demonstrações desses lemas 5.3.1 e 5.3.2 não têm nada de muito especial. Em realidade, trata-se de resultados bastante conhecidos acerca das funções β e $\langle\ \rangle$, aplicados ao conceito de extensão parcial. No entanto, os lemas 5.3.1 e 5.3.2 permitiram a Máté o seguinte resultado importante.

Proposição 5.3.1 ([Máté(1990)])**.** *Sejam $\mathcal{M}_1$ e $\mathcal{M}_2$ modelos não-padrão enumeráveis da aritmética e $n \in dom(\mathcal{M}_1) \cap dom(\mathcal{M}_2)$ um número não-padrão tal que $\mathcal{M}_1 \simeq_n \mathcal{M}_2$ e $\mathcal{M}_1 \preceq_{2^{n^{\mathbb{N}}}} \mathcal{M}_2$. Então $(x^y)^{\mathcal{M}_1} = (x^y)^{\mathcal{M}_2}$ para todo $x, y \in dom(\mathcal{M}_1)$ tal que $x, y < (n^k)^{\mathcal{M}_1}$ para algum $k \in \mathbb{N}$.*

Demonstração. Considere a seqüência $((x^i)^{\mathcal{M}_1} : i < n^k)$. Essa seqüência é codificável em $\mathcal{M}_1$ e, pelo lema 5.3.2, também é codificável em $\mathcal{M}_2$ porquanto $(x^i)^{\mathcal{M}_1} < (2^{x\times i})^{\mathcal{M}_1} < (2^{n^{2\times k}})^{\mathcal{M}_1}$ desde que $x, y < (n^k)^{\mathcal{M}_1}$. Assim, podemos usar indução internamente em $\mathcal{M}_2$ para mostrar que $((x^i)^{\mathcal{M}_1} : i < n^k) = ((x^i)^{\mathcal{M}_2} : i < n^k)$. De fato, suponha que essas sequências são diferentes. Pelo princípio de indução em $\mathcal{M}_2$, existe o menor $j \in \mathcal{M}_2$ tal que $(x^j)^{\mathcal{M}_1} \neq (x^j)^{\mathcal{M}_2}$. Externamente, todavia, temos que $(x^{j-1})^{\mathcal{M}_1} = (x^{j-1})^{\mathcal{M}_2}$ e, por conseguinte,

$$(x^j)^{\mathcal{M}_1} = ((x^{j-1})^{\mathcal{M}_1} \times x)^{\mathcal{M}_1} = ((x^{j-1})^{\mathcal{M}_2} \times x)^{\mathcal{M}_2} = (x^j)^{\mathcal{M}_2},$$

o que contradiz a hipótese de que as seqüência são diferentes. Logo, em virtude de $y < n^k$, $(x^y)^{\mathcal{M}_1} = (x^y)^{\mathcal{M}_2}$. □

Essa proposição mostra que, para elementos menores do que $n^{\mathbb{N}}$, a exponenciação comporta-se da mesma maneira em ambas as estruturas $\mathcal{M}_1$ e $\mathcal{M}_2$. Na abordagem de Máté, esse resultado é crucial, porque ele usa a codificação das máquinas de Turing para tratar da aceitação de seqüência com comprimento não-padrão. Na abordagem que propomos, ela será necessária na

próxima seção apenas porque precisamos garantir que as exponenciações formadas por números menores do que $n^{\mathbb{N}}$ resultem no mesmo número tanto em $\mathcal{M}_1$ quanto em $\mathcal{M}_2$. Em especial, note que a demonstração dessa proposição 5.3.1 é interessante porque ela mostra a distinção entre o uso interno e externo do princípio de indução. Distinção de usos que argumentamos também ser necessária quando tratamos de computações em modelos não-padrão da aritmética através de máquinas de Turing.

A heurística de Máté afirma que a construção de modelos não-padrão da aritmética $\mathcal{M}_1$ e $\mathcal{M}_2$ tais que $\mathcal{M}_1 \simeq_n \mathcal{M}_2$ e $\mathcal{M}_1 \preccurlyeq_{2^{n^k}} \mathcal{M}_2$, para algum número não-padrão $n \in dom(\mathcal{M}_1) \cap dom(\mathcal{M}_2)$ e para todo $k \in \mathbb{N}$, implica $P \neq NP$. Explicamos que tal consequência advinha do fato de que $\mathcal{M}_1 \simeq_n \mathcal{M}_2$ implica $\mathcal{M}_1 \simeq_{n^k} \mathcal{M}_2$, para qualquer número $k \in \mathbb{N}$, e, portanto, não haveria maneira de alguma máquina de Turing distinguir em tempo polinomial n^k, para algum $k \in \mathbb{N}$, os valores de verdade de ϕ em $\mathcal{M}_1$ e $\mathcal{M}_2$. Uma vez que Máté não dispunha das técnicas desenvolvidas nesta tese acerca da computação em modelos aritméticos não-padrão, ele teve que percorrer um itinerário indireto. Em contrapartida, agora podemos apresentar a abordagem de Máté de forma direta. O próximo lema, constatado por C. Sureson [Sureson(1995)], será o passo crucial para isso.

Lema 5.3.3 ([Sureson(1995)])**.** *Se $\mathcal{M}_1$ e $\mathcal{M}_2$ são dois modelos não-padrão enumeráveis da aritmética e $n \in dom(\mathcal{M}_1) \cap dom(\mathcal{M}_2)$ é um número não-padrão tal que $\mathcal{M}_1 \simeq_n \mathcal{M}_2$ e $\mathcal{M}_1 \preceq_{2^n} \mathcal{M}_2$, então $\mathcal{M}_1 \simeq_{(n^{\mathbb{N}})^{\mathcal{M}_1}} \mathcal{M}_2$.*

Demonstração. Suponha que $\mathcal{M}_1 \simeq_n \mathcal{M}_2$ e $\mathcal{M}_1 \preceq_{2^n} \mathcal{M}_2$. A demonstração é por indução em $k \in \mathbb{N}$.

Como $\mathcal{M}_1 \simeq_n \mathcal{M}_2$, segue-se que $(n)^{\mathcal{M}_1} = (n)^{\mathcal{M}_2}$ e, portanto, $\mathcal{M}_1 \simeq_{(n^k)^{\mathcal{M}_1}} \mathcal{M}_2$ para $k = 1$. Suponha que $\mathcal{M}_1 \simeq_{(n^k)^{\mathcal{M}_1}} \mathcal{M}_2$. Assim, $(n^k)^{\mathcal{M}_1} = (n^k)^{\mathcal{M}_2}$ e, ademais, $(n^{k+1})^{\mathcal{M}_1} = (n^k)^{\mathcal{M}_1} \times^{\mathcal{M}_1} n = (n^k)^{\mathcal{M}_2} \times^{\mathcal{M}_2} n = (n^{k+1})^{\mathcal{M}_2}$. Agora mostraremos que $\{x \in \mathcal{M}_2 : \mathcal{M}_2 \vDash x < n^{k+1}\} \subseteq \mathcal{M}_1$.

Se $\mathcal{M}_2 \vDash x < n^{k+1}$, então, pelo princípio de indução da aritmética de Peano, sabemos que existem $a <_{\mathcal{M}_2} n$ e $b <_{\mathcal{M}_2} n^k$ tais que $x = (a \times^{\mathcal{M}_2} n^k) +^{\mathcal{M}_2} b$.

Por hipótese de indução, $a, b, (n^k)^{\mathcal{M}_1} = (n^k)^{\mathcal{M}_2} \in \mathcal{M}_1$ e todos são menores do que $(2^n)^{\mathcal{M}_1}$. Uma vez que $\mathcal{M}_2$ é uma extensão parcial de $\mathcal{M}_1$ até $(2^n)^{\mathcal{M}_1}$, concluímos que $a \times^{\mathcal{M}_2} n^k = a \times^{\mathcal{M}_1} n^k$ e são menores do que $(2^n)^{\mathcal{M}_1}$. Assim, $(a \times^{\mathcal{M}_2} n^k) +^{\mathcal{M}_2} b = (a \times^{\mathcal{M}_1} n^k) +^{\mathcal{M}_1} b$, o que significa que $x \in \mathcal{M}_1$. Logo, $\mathcal{M}_1$ e $\mathcal{M}_2$ são aritmeticamente isomórficos até $(n^{k+1})^{\mathcal{M}_1} = (n^{k+1})^{\mathcal{M}_2}$. □

Teorema 5.3.1. *Se $\mathcal{M}_1$ e $\mathcal{M}_2$ são dois modelos não-padrão enumeráveis da aritmética e $n \in dom(\mathcal{M}_1) \cap dom(\mathcal{M}_2)$ é um número não-padrão tal que $\mathcal{M}_1 \simeq_n \mathcal{M}_2$ e $\mathcal{M}_1 \preceq_{2^{n^{\mathbb{N}}}} \mathcal{M}_2$, então $n^{\mathbb{N}} \equiv_P 2^n$.*

Demonstração. Assuma a hipótese. Primeiramente, vamos mostrar que, para qualquer palavra $\vec{s}$ em $\{0,1\}^*$ tal que $|\vec{s}| \leq n^{\mathbb{N}}$ e qualquer máquina de Turing determinista M, $M[\vec{s}] = 1$ em $t \leq n^k$ passos com $t \in dom(\mathcal{M}_1)$ se, e somente se, $M[\vec{s}] = 1$ em $t \leq n^k$ passos com $t \in dom(\mathcal{M}_2)$. Em particular, isso implica que $(n^{\mathbb{N}})^{\mathcal{M}_1} \equiv_P (2^n)^{\mathcal{M}_2}$, porque a representação binária de 2^n tem comprimento menor ou igual a n^k para $k \in \mathbb{N}$.

Seja $\vec{s}$ uma palavra em $\{0,1\}^*$ tal que $|\vec{s}| \leq (n^k)^{\mathcal{M}_1}$ para $k \in \mathbb{N}$. Suponha que M é uma máquina de Turing tal que $M[\vec{s}] = 1$ em $t \leq n^k$ passos com $t \in dom(\mathcal{M}_1)$. Pelo teorema da completude 2.5.2, $\mathcal{T}_M^{\vec{s}} \vdash \phi(0) \to \psi(t)$ em $\mathcal{M}_1$, onde $\phi(x)$ é a fórmula correspondente a configuração inicial de $C^{\vec{s}}$ e $\psi(x)$ é a fórmula correspondente a configuração final de $C^{\vec{s}}$, sendo x uma variável para o parâmetro de tempo. Pelo lema 5.3.3, $\mathcal{M}_1 \simeq_{n^k} \mathcal{M}_2$ para todo $k \in \mathbb{N}$ e, por conseguinte, $(t \leq n^k)^{\mathcal{M}_1} = (t \leq n^k)^{\mathcal{M}_2}$, ou seja, $t^{\mathcal{M}_1} = t^{\mathcal{M}_2}$. Mais do que isso, $(|\vec{s}| \leq n^k)^{\mathcal{M}_1} = (|\vec{s}| \leq n^k)^{\mathcal{M}_2}$, o que significa que a entrada $\vec{s}$ é exatamente a mesma palavra tanto em $\mathcal{M}_1$ quanto em $\mathcal{M}_2$. Dessa forma, $\mathcal{T}_M^{\vec{s}} \vdash \phi(0) \to \psi(t)$ em $\mathcal{M}_2$. Como $|2^n| \leq n$, $\mathcal{T}_M^{\vec{s}} \vdash \phi(0) \to \psi(|2^n|^k)$ em $\mathcal{M}_2$, ou seja, $M[\vec{s}] = 1$ em $t \leq n^k$ passos com $t \in dom(\mathcal{M}_2)$ para $k \in \mathbb{N}$. Reciprocamente, se $M[\vec{s}] = 1$ em $t \leq |2^n|^k \leq n^k$ passos com $t \in dom(\mathcal{M}_2)$ para $k \in \mathbb{N}$, então com um argumento similar podemos mostrar que $M[\vec{s}] = 1$ em $t \leq n^k$ passos com $t \in dom(\mathcal{M}_1)$.

Para concluir, observe que a proposição 5.3.1 afirma que $(n^k)^{\mathcal{M}_1} = (n^k)^{\mathcal{M}_2}$ para todo $k \in \mathbb{N}$ e $(2^n)^{\mathcal{M}_1} = (2^n)^{\mathcal{M}_2}$. Dessa forma, podemos omitir os superescritos em $(n^{\mathbb{N}})^{\mathcal{M}_1} \equiv_P (2^n)^{\mathcal{M}_2}$, ou seja, demonstramos que $n^{\mathbb{N}} \equiv_P 2^n$. □

Isso finaliza nossa primeira parte da caracterização local do problema $P \stackrel{?}{=} NP$. Cabe destacar que Máté apresentou uma caracterização do problema $NP \stackrel{?}{=} coNP$, onde $coNP$ é a classe complementar de NP. Obviamente, se $NP \neq coNP$, então $P \neq NP$ - a recíproca não é verdadeira. A partir do teorema 5.3.1 podemos facilmente derivar o resultado de Máté.

5.4 Uma caracterização local do problema $P \stackrel{?}{=} NP$ (Segunda parte)

A fim de concluirmos nossa caracterização local do problema $P \stackrel{?}{=} NP$, precisaremos de uma análise pormenorizada da construção de modelos não-padrão da aritmética partindo da hipótese de que $P \neq NP$. Isso, por sua vez, exige uma caracterização sintática das classes P e NP. Felizmente, S. Buss [Buss(1986)] definiu uma linguagem aritmética na qual podemos definir uma hierarquia que coincide com a hierarquia PH.

Definição 5.4.1. *A* linguagem aritmética de Buss L_b *é a linguagem* $L_b = L \cup \{|x|, \lfloor \frac{x}{2} \rfloor, 2^{|x|\times|y|}\}$, *onde* $|x|$ *é um símbolo funcional unário que representa a função que determina o comprimento da representação binária de* x, *ou seja,* $|x| = \lceil \log_2(x+1) \rceil$[6], *os demais símbolos também são funcionais e sua aridade e significado intencionado estão indicados pelas próprias expressões. A* hierarquia de Buss $\Delta^b_{\mathbb{N}} = \bigcup_{i\leq 0} \Sigma^b_i = \bigcup_{i\leq 0} \Pi^b_i$ *é definida do seguinte modo:*

- $\Sigma^b_0 = \Pi^b_0$ *é o conjunto das* L_b*-fórmulas cujas fórmulas são quantificadas por quantificadores da forma* $\forall x < |\tau(y)|$ *e* $\exists x < |\tau(y)|$;
- Σ^b_{n+1} *e* Π^b_{n+1} *são os menores conjuntos de* L_b*-fórmulas tais que:*
  - $\Sigma^b_n = \Pi^b_n \subseteq \Sigma^b_{n+1}$ *e* $\Sigma^b_n = \Pi^b_n \subseteq \Pi^b_{n+1}$;
  - *Se* $\phi(x\vec{y}) \in \Sigma^b_n$, *então* $\exists x < \tau(y)\phi(x\vec{y}) \in \Sigma^b_n$ *e* $\forall x < |\tau(y)|\phi(x\vec{y}) \in \Sigma^b_n$;

[6] $\lceil x \rceil$ representa a *função teto* que é o menor inteiro maior ou igual a x.

- *Se* $\phi(x\vec{y}) \in \Pi^b_n$, *então* $\forall x < \tau(y)\phi(x\vec{y}) \in \Pi^b_n$ *e* $\exists x < |\tau(y)|\phi(x\vec{y}) \in \Sigma^b_n$;
- $\Sigma^b_n = \Pi^b_n \subseteq \Sigma^b_{n+1}$ *e* $\Sigma^b_n = \Pi^b_n \subseteq \Pi^b_{n+1}$ *são fechadas sob* $\wedge$ *e* $\vee$;
- $\phi(x\vec{y}) \in \Sigma^b_n$ *se, e somente se,* $\neg\phi(x\vec{y}) \in \Pi^b_n$ *bem como* $\phi(x\vec{y}) \in \Pi^b_n$ *se, e somente se,* $\neg\phi(x\vec{y}) \in \Sigma^b_n$.

Buss demonstrou que $\Delta^b_{\mathbb{N}} = PH$. Dessa forma, a classe NP é a classe dos conjuntos de números naturais definíveis por Σ^b_1-fórmulas, $coNP$ por Π^b_1-fórmulas, e P é a classe dos conjuntos que são definíveis tanto por fórmulas em Σ^b_1 quanto por fórmulas em Π^b_1. No que se segue apresentaremos uma generalização do resultado demonstrado por C. Sureson [Sureson(1995), p.61-65]. A estratégia de demonstração é igual aquela desenvolvida por Sureson, exceto pelo fato de que mencionaremos fórmulas da linguagem de Buss mas não usaremos as funções $|x|$, $\lfloor \frac{x}{2} \rfloor$ e $2^{|x|\times|y|}$ na construção dos modelos.

Lema 5.4.1 ([Sureson(1995)])**.** *Seja* $\mathcal{M}$ *um modelo enumerável não-padrão da aritmética. Nesse caso, os seguintes enunciados são verdadeiros:*

1. *Se* ϕ *é uma fórmula aritmética de Buss tal que* $\phi \in \Sigma^b_n - \Pi^b_n$, *então existem submodelos* $\mathcal{M}_1$ *e* $\mathcal{M}_2$ *de* $\mathcal{M}$ *e existe* $m \in \mathcal{M} - \mathcal{N}$ *tal que* $\Sigma^b_n((m)^{\mathcal{M}_2}) \subseteq \Sigma^b_n((m)^{\mathcal{M}_1})$ *bem como* $\mathcal{M}_1 \vDash \phi(m)$ *e* $\mathcal{M}_2 \vDash \neg\phi(m)$.

2. *Se* ψ *é uma fórmula aritmética de Buss tal que* $\psi \in \Pi^b_n - \Sigma^b_n$, *então existem submodelos* $\mathcal{M}_1$ *e* $\mathcal{M}_2$ *de* $\mathcal{M}$ *e existe* $m \in \mathcal{M} - \mathcal{N}$ *tal que* $\Pi^b_n((m)^{\mathcal{M}_2}) \subseteq \Pi^b_n((m)^{\mathcal{M}_1})$ *bem como* $\mathcal{M}_1 \vDash \psi(m)$ *e* $\mathcal{M}_2 \vDash \neg\psi(m)$.

Demonstração. Seja $\mathcal{M}$ um modelo enumerável não-padrão da aritmética, $\phi \in \Sigma^b_n - \Pi^b_n$ e $\psi \in \Pi^b_n - \Sigma^b_n$. Para começar, mostraremos que, dado um $k \in \mathbb{N}$, $\mathcal{N} \vDash \exists x, y \forall n < k(n = \ulcorner\xi\urcorner \rightarrow ((Sat(n,x) \rightarrow Sat(n,y)) \wedge \neg\phi(x) \wedge \phi(y)))$ para $\xi \in \Sigma^b_n$ e, ulteriormente, mostraremos que $\mathcal{N} \vDash \exists x, y \forall n < k(n = \ulcorner\xi\urcorner \rightarrow ((Sat(n,x) \rightarrow Sat(n,y)) \wedge \neg\psi(x) \wedge \psi(y))$ para $\xi \in \Pi^b_n$, onde Sat indica o predicado de satisfatibilidade da aritmética de Buss. No primeiro caso, suponha, pelo contrário, que $\mathcal{N} \vDash \forall x, y \exists n < k(n = \ulcorner\xi\urcorner \wedge ((Sat(n,x) \rightarrow Sat(n,y)) \rightarrow (\phi(x) \vee \neg\phi(y))))$. Defina $I = \{\ulcorner\xi\urcorner \in \mathbb{N} : \xi \in \Sigma^b_n \wedge \ulcorner\xi\urcorner < k\}$. Assim,

(*) $\mathcal{N} \vDash \forall x, y(\bigwedge_{\ulcorner\xi\urcorner \in I}(\xi(x) \to \xi(y)) \to (\phi(x) \vee \neg\phi(y)))$

Considere $\sigma \in 2^{(k)}$. Para $i < k$, defina

$$\phi_i^\sigma(x) = \begin{cases} x \approx x & \text{se não existe } \xi \in \Sigma_n^b \text{ tal que } \ulcorner\xi\urcorner = i, \\ \xi(x) & \text{se existe } \xi \in \Sigma_n^b \text{ tal que } \ulcorner\xi\urcorner = i \text{ e } \sigma(i) = 1, \\ \neg\xi(x) & \text{se existe } \xi \in \Sigma_n^b \text{ tal que } \ulcorner\xi\urcorner = i \text{ e } \sigma(i) = 0. \end{cases}$$

Se existir $x \in \mathbb{N}$ para o qual $\mathcal{N} \vDash \bigwedge_{i<k} \phi_i^\sigma(x)$, seja x_σ o menor desses x. Agora defina $J = \{\sigma \in 2^{(k)} : \mathcal{N} \vDash \bigwedge_{i<k} \phi_i^\sigma(x_\sigma) \wedge \mathcal{N} \vDash \phi(x_\sigma)\}$. Considere a Π_n^b-fórmula $\zeta(x) = \bigvee_{\sigma\in J} \bigwedge_{\sigma(i)=0} \phi_i^\sigma(x)$. Mostraremos que $\mathcal{N} \vDash \forall x(\zeta(x) \leftrightarrow \phi(x))$, o que é um absurdo porque $\phi \in \Sigma_n^b - \Pi_n^b$.

Suponha que $\mathcal{N} \vDash \phi(x)$. Considere $\sigma_x \in 2^{(k)}$ tal que

$$\sigma_x(i) = \begin{cases} 1 & \text{se } \ulcorner\xi\urcorner = i \text{ e } \mathcal{N} \vDash \xi(x), \\ 0 & \text{se } \ulcorner\xi\urcorner = i \text{ e } \mathcal{N} \vDash \neg\xi(x). \end{cases}$$

Assim x_{σ_x} está bem definido e $\mathcal{N} \vDash \bigwedge_{\ulcorner\xi\urcorner\in I}(\xi(x_{\sigma_x}) \to \xi(x))$; o que implica que $\mathcal{N} \vDash \phi(x_{\sigma_x}) \vee \neg\phi(x)$ devido a (*). Como $\mathcal{N} \vDash \phi(x)$, temos que $\sigma_x \in J$ e $\mathcal{N} \vDash \bigwedge_{\sigma_x(i)=0} \phi_i^\sigma(x)$, ou seja, $\mathcal{N} \vDash \zeta(x)$. Conversamente, suponha que $\mathcal{N} \vDash \bigvee_{\sigma\in J} \bigwedge_{\sigma(i)=0} \phi_i^\sigma(x)$. Então existe $\sigma \in J$ e $\mathcal{N} \vDash \bigwedge_{\sigma(i)=0} \phi_i^\sigma(x)$. Suponha que $\ulcorner\xi\urcorner \in I$ e $\mathcal{N} \vDash \neg\xi(x_\sigma)$. Assim, $\sigma(i) = 0$, porque $\sigma(i) = 1$ implicaria $\mathcal{N} \vDash \xi(x_\sigma)$ - pela definição de x_σ. Isso significa que $\mathcal{N} \vDash \bigwedge_{\sigma_x(i)=0} \phi_i^\sigma(x) \to \neg\xi(x)$ e, uma vez que $\mathcal{N} \vDash \bigwedge_{\sigma_x(i)=0} \phi_i^\sigma(x)$, temos $\mathcal{N} \vDash \neg\xi(x)$. Portanto, $\mathcal{N} \vDash \bigwedge_{\ulcorner\xi\urcorner\in I} \xi(x) \to \xi(x_{\sigma_x})$; o que implica que $\mathcal{N} \vDash \phi(x) \vee \neg\phi(x_\sigma)$ devido a (*). Em virtude de $\sigma \in J$, derivamos que $\mathcal{N} \vDash \phi(x)$.

Assim demonstramos que, dado um $k \in \mathbb{N}$, $\mathcal{N} \vDash \exists x, y \forall n \prec k(n = \ulcorner\xi\urcorner \to ((Sat(n, x) \to Sat(n, y)) \wedge \neg\phi(x) \wedge \phi(y))$ para $\xi \in \Sigma_n^b$. Para mostrar que, dado um $k \in \mathbb{N}$, $\mathcal{N} \vDash \exists x, y \forall n \prec k(n = \ulcorner\xi\urcorner \to ((Sat(n, x) \to Sat(n, y)) \wedge \neg\psi(x) \wedge \psi(y))$ para $\xi \in \Pi_n^b$, defina $\psi_i^\sigma(x)$ como em $\phi_i^\sigma(x)$ mas trocando Σ_n^b por Π_n^b. Nesse caso, $\zeta(x) = \bigvee_{\sigma\in J} \bigwedge_{\sigma(i)=0} \psi_i^\sigma(x)$ será uma Σ_n^b-fórmula. Afora isso, o argumento é análogo ao caso anterior.

Com uma redução ao absurdo similar a desenvolvida até aqui, também

podemos mostrar que, dado um $k \in \mathbb{N}$, $\mathcal{N} \vDash \exists x \geqslant y \forall n \prec k(n = \ulcorner \xi \urcorner \rightarrow ((Sat(n,x) \rightarrow Sat(n,y)) \wedge \neg\phi(x) \wedge \phi(y))$ para $\xi \in \Sigma_n^b$ e $\mathcal{N} \vDash \exists x \geqslant y \forall n \prec k(n = \ulcorner \xi \urcorner \rightarrow ((Sat(n,x) \rightarrow Sat(n,y)) \wedge \neg\psi(x) \wedge \psi(y))$ para $\xi \in \Pi_n^b$. Para tanto, basta considerar as fórmulas $\zeta(x) = \bigvee_{\sigma \in J}(\bigwedge_{\sigma(i)=0} \phi_i^\sigma(x) \wedge x \geq x_\sigma)$ e $\zeta(x) = \bigvee_{\sigma \in J}(\bigwedge_{\sigma(i)=0} \psi_i^\sigma(x) \wedge x \geq x_\sigma)$ no primeiro e segundo caso, respectivamente, e observar que em ambos $x \geq x_{\sigma_x}$. Assim, podemos derivar contradições. Uma que esses enunciados são verdadeiros em $\mathcal{N}$ também o são em $\mathcal{M}$ e, por conseguinte, $\mathcal{M} \vDash \forall k \exists x \geqslant y \forall n \prec k(n = \ulcorner \xi \urcorner \rightarrow ((Sat(n,x) \rightarrow Sat(n,y)) \wedge \neg\phi(x) \wedge \phi(y))$ para $\xi \in \Sigma_n^b$ e $\mathcal{M} \vDash \exists x \geqslant y \forall n \prec k(n = \ulcorner \xi \urcorner \rightarrow ((Sat(n,x) \rightarrow Sat(n,y)) \wedge \neg\psi(x) \wedge \psi(y))$ para $\xi \in \Pi_n^b$. Em especial, essas sentenças também devem ser verdadeiras para algum $k \in \mathcal{M} - \mathcal{N}$. Ora, essas duas sentenças afirmam que existem dois números não-padrão $m_2 \geq m_1$ de $\mathcal{M}$ para os quais $\Sigma_n^b(m_2) \subseteq \Sigma_n^b(m_1)$, $\Pi_n^b(m_2) \subseteq \Pi_n^b(m_1)$ e $\mathcal{M} \vDash \neg\phi(m_2) \wedge \phi(m_1)$ e $\mathcal{M} \vDash \neg\psi(m_2) \wedge \psi(m_1)$. Para concluir, considere $\mathcal{M}_1$ e $\mathcal{M}_2$ dois submodelos de $\mathcal{M}$ que diferem, entre outras coisas, pelo fato de que $m_2 \in \mathcal{M}_2 - \mathcal{M}_1$ e $m_1 \in \mathcal{M}_1 - \mathcal{M}_2$. Considere $(m)^{\mathcal{M}_1} = m_1$ e $(m)^{\mathcal{M}_2} = m_2$. Logo, $\mathcal{M}_1 \vDash \phi(m)$ e $\mathcal{M}_2 \vDash \neg\phi(m)$ bem como $\mathcal{M}_1 \vDash \psi(m)$ e $\mathcal{M}_2 \vDash \neg\psi(m)$. □

Teorema 5.4.1. *Se $\mathcal{M}$ é um modelo enumerável não-padrão da aritmética e ϕ é uma fórmula da aritmética de Buss tal que ou $\phi \in \Sigma_n^b - \Pi_n^b$ ou $\phi \in \Pi_n^b - \Sigma_n^b$, então existem dois submodelos $\mathcal{M}_1$ e $\mathcal{M}_2$ de $\mathcal{M}$ tais que $n \in \mathcal{M}_1 \cap \mathcal{M}_2$, $\mathcal{M}_1 \simeq_{(n)^{\mathcal{M}_1}} \mathcal{M}_2$, $\mathcal{M}_1 \preceq_{(2^{n^{\mathbb{N}}})^{\mathcal{M}_1}} \mathcal{M}_2$ e existe $m \leq (2^n)^{\mathcal{M}_1}$ para o qual $\mathcal{M}_1 \vDash \phi(m)$ e $\mathcal{M}_2 \vDash \neg\phi(m)$.*

Demonstração. Seja $\mathcal{M}$ um modelo enumerável não-padrão da aritmética, e ϕ uma fórmula da aritmética de Buss tal que ou $\phi \in \Sigma_n^b - \Pi_n^b$ ou $\phi \in \Pi_n^b - \Sigma_n^b$. Pelo lema 5.4.1, em qualquer caso, sabemos que exitem submodelos $\mathcal{M}_1$ e $\mathcal{M}_2$ de $\mathcal{M}$ tais que $\mathcal{M}_1 \vDash \phi(m)$ e $\mathcal{M}_2 \vDash \neg\phi(m)$. Precisamos apenas refinar $\mathcal{M}_1$, $\mathcal{M}_2$ e m de modo que eles tenham as propriedades desejadas. Para tanto, considere $(c_u : u \in \mathbb{N})$ uma lista com todos os elementos de $\mathcal{M}$, e mostraremos que os submodelos $\mathcal{M}_1$ e $\mathcal{M}_2$ de $\mathcal{M}$ podem ser definidos em termos de um homomorfismo h, o qual será gerado a partir de duas subsequências $(a_i : i \in \mathbb{N})$

e $(b_i : i \in \mathbb{N})$ de $(c_u : u \in \mathbb{N})$ usando uma espécie de argumento de ida-e-volta. Para tanto, suponha que $(a_j : j \leq 2l)$ e $(b_j : j \leq 2l)$ já estão definidas e são tais que $a_0 = b_0 = (2^n)^{\mathcal{M}}$ bem como $\Sigma^b_0(a_0, \ldots, a_{2l}) \subseteq \Sigma^b_0(b_0, \ldots, b_{2l})$ se $\phi \in \Sigma^b_n - \Pi^b_n$ e $\Pi^b_0(a_0, \ldots, a_{2l}) \subseteq \Pi^b_0(b_0, \ldots, b_{2l})$ se $\phi \in \Pi^b_n - \Sigma^b_n$. Adicionaremos os próximos elementos das sequências $(a_j : j \leq 2l)$ e $(b_j : j \leq 2l)$ de forma que os tipos da primeira continue sendo um subconjunto dos tipos da segunda, conforme $\phi \in \Sigma^b_n - \Pi^b_n$ ou $\phi \in \Pi^b_n - \Sigma^b_n$.

No passo $2l + 1$ defina b_{2l+1} como sendo o primeiro elemento b em $(c_u : u \in \mathbb{N})$ tal que $b < n^k$, para algum $k \in \mathbb{N}$, e $b \notin \{b_0, \ldots, b_{2l}\}$. Considere os conjuntos $\Phi(x) = \{\phi(a_0, \ldots, a_{2l}, x) \to \phi(b_0, \ldots, b_{2l}, b_{2l+1}) : \phi \in \Sigma^b_n\}$ e $\Psi(x) = \{\psi(a_0, \ldots, a_{2l}, x) \to \psi(b_0, \ldots, b_{2l}, b_{2l+1}) : \psi \in \Pi^b_n\}$. Obteremos o elemento a_{2l+1} ao mostrar que $\Phi(x)$ e $\Psi(x)$ são realizáveis em $\mathcal{M}$ conforme $\phi \in \Sigma^b_n - \Pi^b_n$ ou $\phi \in \Pi^b_n - \Sigma^b_n$, respectivamente. Como $\Phi(x)$ e $\Psi(x)$ são computáveis, apenas precisamos mostrar que eles também são finitamente satisfatíveis em $\mathcal{M}$; consideraremos o primeiro caso, pois, para o segundo, basta considerar um argumento dual. Seja $\phi_0, \ldots, \phi_s$ um conjunto finito qualquer de Σ^b_n-fórmulas tais que $\mathcal{M} \vDash \neg\phi_0(b_0, \ldots, b_{2l}, b_{2l+1}), \ldots, \mathcal{M} \vDash \neg\phi_s(b_0, \ldots, b_{2l}, b_{2l+1})$. Daí $\mathcal{M} \vDash \exists y \prec n^k \bigwedge_{j \leq l} \neg\phi_j(b_0, \ldots, b_{2l}, y)$. Uma vez que $\Sigma^b_0(a_0, \ldots, a_{2l}) \subseteq \Sigma^b_0(b_0, \ldots, b_{2l})$, $\Pi^b_0(b_0, \ldots, b_{2l}) \subseteq \Pi^b_0(a_0, \ldots, a_{2l})$ e, consequentemente, $\mathcal{M} \vDash \exists x \prec n^k \bigwedge_{j \leq s} \neg\phi_j(a_0, \ldots, a_{2l}, x)$ porque $\bigwedge_{j \leq s} \neg\phi_j(a_0, \ldots, a_{2l}, x) \in \Pi^b_0$. Considerando d um tal x, $\mathcal{M} \vDash \bigwedge_{j \leq s}(\phi_j(a_0, \ldots, a_{2l}, d) \to \phi_j(b_0, \ldots, b_{2l}, b_{2l+1}))$. Dessa forma, o conjunto $\Phi(x)$ é finitamente satisfatível em $\mathcal{M}$ e, em virtude de $\mathcal{M}$ ser um modelo computacionalmente saturado, $\Phi(x)$ é um tipo computacionalmente realizável. Seja a_{2l+1} um número que realiza $\Phi(x)$. Portanto, $(\Sigma^b_n(a_0, \ldots, a_{2l+1}))^{\mathcal{M}} \subseteq (\Sigma^b_n(b_0, \ldots, b_{2l+1}))^{\mathcal{M}}$.

No passo $2l + 2$ defina a_{2l+2} como sendo o primeiro elemento a em $(c_u : u \in \mathbb{N})$ tal que $a < 2^{n^k}$, para algum $k \in \mathbb{N}$, e $a \notin \{a_0, \ldots, a_{2l+1}\}$. Considere o conjunto de fórmulas $\Phi(y) = \{\phi(a_0, \ldots, a_{2l+2}) \to \phi(b_0, \ldots, b_{2n+1}, y) : \phi \in \Sigma^b_1\}$ e $\Phi(y) = \{\psi(a_0, \ldots, a_{2l+2}) \to \psi(b_0, \ldots, b_{2n+1}, y) : \psi \in \Sigma^b_1\}$. Obteremos o elemento b_{2l+2} de forma análoga ao parágrafo anterior. Considere, então, um con-

junto finito de Σ_0^b-fórmulas $\phi_0, \ldots, \phi_s$ tais que $\mathcal{M} \vDash \phi_0(a_0, \ldots, a_{2l+2}), \ldots, \mathcal{M} \vDash \phi_s(a_0, \ldots, a_{2l+2})$. Daí, $\mathcal{M} \vDash \exists x \prec 2^{n^k} \bigwedge_{j \leq l} \phi_j(a_0, \ldots, a_{2l+1}, x)$ e, dado que no parágrafo anterior mostramos que $(\Sigma_0^b(a_0, \ldots, a_{2l+1}))^{\mathcal{M}} \subseteq (\Sigma_0^b(b_0, \ldots, b_{2l+1}))^{\mathcal{M}}$, então $\mathcal{M} \vDash \exists y \prec 2^{n^k} \bigwedge_{j \leq l} \phi_j(b_0, \ldots, b_{2l+1}, y)$. Assim, similarmente ao parágrafo anterior, obtemos o elemento b_{2l+2} para o qual $(\Sigma_0^b(a_0, \ldots, a_{2l+2}))^{\mathcal{M}} \subseteq (\Sigma_0^b(b_0, \ldots, b_{2l+2}))^{\mathcal{M}}$.

Feito isso, defina a função $h : \{a_i : i \in \mathbb{N}\} \to \{a_i : i \in \mathbb{N}\}$ tal que $h(a_i) = b_i$ para todo $i \in \mathbb{N}$. Claramente, h é uma função injetora. O passo $2l+2$ garante que h é uma função de $(2^{n^{\mathbb{N}}})^{\mathcal{M}}$ para $(2^{n^{\mathbb{N}}})^{\mathcal{M}}$. O passo $2l+1$ assegura, por sua vez, que h é sobrejetora em $(n^{\mathbb{N}})^{\mathcal{M}}$. Como $\mathcal{M}$ é um modelo da aritmética, para cada $l \in \mathbb{N}$, o conjunto das fórmulas sem variáveis livres verdadeiras sobre $a_0, \ldots, a_l$ e $b_0, \ldots, b_l$ é exatamente igual, ou seja, h é realmente um homomorfismo.

Agora estamos prontos para definir os submodelos $\mathcal{M}_1$ e $\mathcal{M}_2$ de $\mathcal{M}$. Para tanto, estipulamos que $\mathcal{M}_2 = \mathcal{M}$ e definimos $\mathcal{M}_1$ do seguinte modo:

- O domínio de $\mathcal{M}_1$ é o conjunto $h[(2^{n^{\mathbb{N}}})^{\mathcal{M}_2}] \cup \{x \in \mathcal{M}_2 : (2^{n^{\mathbb{N}}})^{\mathcal{M}_2} < x\}$;
- $x +^{\mathcal{M}_1} y = \begin{cases} x +^{\mathcal{M}_2} y & \text{se } x, y < (2^{n^{\mathbb{N}}})^{\mathcal{M}_2} \text{ ou } (2^{n^{\mathbb{N}}})^{\mathcal{M}_2} < x, y \\ h^{-1}(x) +^{\mathcal{M}_2} y & \text{se } x < (2^{n^{\mathbb{N}}})^{\mathcal{M}_2} \text{ e } (2^{n^{\mathbb{N}}})^{\mathcal{M}_2} < y \end{cases}$
- $\times^{\mathcal{M}_1}$ é definida de forma análoga à $+^{\mathcal{M}_1}$;
- $<^{\mathcal{M}_1}$ é restrição de $<^{\mathcal{M}_2}$ ao domínio de $\mathcal{M}_1$.

Agora considere a função $\widehat{h} : \mathcal{M}_2 \to \mathcal{M}_1$ tal que

$$\widehat{h}(x) = \begin{cases} h(x) & \text{se } x < (2^{n^{\mathbb{N}}})^{\mathcal{M}} \\ Id(x) & \text{se } (2^{n^{\mathbb{N}}})^{\mathcal{M}_2} < x \end{cases}$$

Claramente, $\widehat{h}$ é um isomorfismo; o que significa que $\mathcal{M}_1$ e $\mathcal{M}_2$ são modelos da aritmética. Por isso, temos as seguintes consequências. Primeiro, $\widehat{h}(n^{\mathcal{M}_2}) = n^{\mathcal{M}_1}$. Segundo, como $n^k < 2^{n^k}$ para todo $k \in \mathbb{N}$,

$\widehat{h}((n^k)^{\mathcal{M}_2}) = \widehat{h}((n)^{\mathcal{M}_2} \overbrace{\times^{\mathcal{M}_2} \cdots \times^{\mathcal{M}_2}}^{k \text{ vezes}} (n)^{\mathcal{M}_2}) = \widehat{h}((n)^{\mathcal{M}_2}) \overbrace{\times^{\mathcal{M}_1} \cdots \times^{\mathcal{M}_1}}^{k \text{ vezes}} h((n)^{\mathcal{M}_2}) = (n)^{\mathcal{M}_1} \overbrace{\times^{\mathcal{M}_1} \cdots \times^{\mathcal{M}_1}}^{k \text{ vezes}} (n)^{\mathcal{M}_1} = (n^k)^{\mathcal{M}_1}$. A partir disso também derivamos que, terceiro, $\widehat{h}((2^{n^k})^{\mathcal{M}_2}) = (2^{n^k})^{\mathcal{M}_1}$; o que significa que podemos omitir os subscritos de todos esses números. Por fim, $\mathcal{M}_1 \simeq_n \mathcal{M}_2$ e $\mathcal{M}_1 \preccurlyeq_{2^{n^k}} \mathcal{M}_2$, uma vez que $\{x \in \mathcal{M}_2 : x <^{\mathcal{M}_2} n\} = \widehat{h}[\{x \in \mathcal{M}_2 : x <^{\mathcal{M}_2} n\}] \subseteq \mathcal{M}_1$ e $+^{\mathcal{M}_1}$ bem como $\times^{\mathcal{M}_1}$ são restrições de $+^{\mathcal{M}_2}$ e $\times^{\mathcal{M}_2}$ para números menores do que 2^{n^k}.

Para finalizar, o lema 5.4.1 assere que existe um número $m \in \mathcal{M} - \mathcal{N}$ e uma Σ_n^b-fórmula ou Π_n^b-fórmula ϕ, conforme $\phi \in \Sigma_n^b - \Pi_n^b$ ou $\phi \in \Pi_n^b - \Sigma_n^b$, tal que $\mathcal{M}_1 \vDash \phi(m)$ e $\mathcal{M}_2 \vDash \neg\phi(m)$. Considere m algum número tal que $m \leq 2^n$. Em virtude das definições da adição e multiplicação, ao consideramos $m \leq 2^n$ e algum número $n > 2^{n^k}$ teremos que $m +^{\mathcal{M}_1} n \neq m +^{\mathcal{M}_2} n$ bem como $m \times^{\mathcal{M}_1} n \neq m \times^{\mathcal{M}_2} n$. Logo, para uma ϕ na qual ocorre adição e multiplicação, poderemos ter $\mathcal{M}_1 \vDash \phi(m)$ e $\mathcal{M}_2 \vDash \neg\phi(m)$. □

Teorema 5.4.2. *Os seguintes enunciados são equivalentes:*

1. $P \neq NP$;
2. *Existem modelos não-padrão da aritmética* $\mathcal{M}_1$ *e* $\mathcal{M}_2$ *com* $n \in (\mathcal{M}_1 \cap \mathcal{M}_2) - \mathcal{N}$ *tal que* $\mathcal{M}_1 \simeq_n \mathcal{M}_2$ *e* $\mathcal{M}_1 \preceq_{(2^{n^k})^{\mathcal{M}_1}} \mathcal{M}_2$ *para todo* $k \in \mathbb{N}$ *e existe uma* Σ_1*-fórmula aritmética* $\phi(x)$ *e um número* $m \leq 2^n$ *tal que* $\mathcal{M}_1 \vDash \phi(m)$ *e* $\mathcal{M}_2 \vDash \neg\phi(m)$.

Demonstração. Primeiramente, mostraremos por contraposição que (2) implica (1). Pelo teorema 5.3.1, $n^k \equiv_P 2^n$ para todo $k \in \mathbb{N}$. Uma vez que máquinas de Turing são definíveis por Σ_1-fórmulas, isso significa que n^k e 2^n estão no mesmo Σ_1-conjunto. Suponha que $P = NP$. Então a hierarquia polinomial HP colapsa com a classe P e, por conseguinte, $\Delta_0^{\mathbb{N}} \subseteq P$ porquanto $\Delta_0^{\mathbb{N}} \subseteq HP$. Nesse caso também temos que $n^k \equiv_{NP} 2^n$ para todo $k \in \mathbb{N}$, ou seja, n^k e 2^n também estão no mesmo Π_1-conjunto. Pela proposição 5.3.1, sabemos que $(2^n)^{\mathcal{M}_1} = (2^n)^{\mathcal{M}_2}$. Portanto, não existem modelos não-padrão da

aritmética $\mathcal{M}_1$ e $\mathcal{M}_2$ com $n \in (\mathcal{M}_1 \cap \mathcal{M}_2) - \mathcal{N}$, $\mathcal{M}_1 \simeq_n \mathcal{M}_2$ e alguma Σ_1-fórmula aritmética $\phi(x)$ tal que $\mathcal{M}_1 \vDash \phi(m)$ e $\mathcal{M}_2 \vDash \neg\phi(m)$ sendo $m \leq (2^n)^{\mathcal{M}_1} = (2^n)^{\mathcal{M}_2}$.

Agora demonstraremos que (1) implica (2). Suponha que $P \neq NP$. Então existe uma fórmula ψ da linguagem aritmética de Buss tal que ou $\psi \in \Sigma_1^b - \Pi_1^b$ ou $\psi \in \Pi_1^b - \Sigma_1^b$. Em ambos os casos, pelo teorema 5.4.1, existem dois submodelos $\mathcal{M}_1$ e $\mathcal{M}_2$ de um modelo arbitrário enumerável não-padrão da aritmética de Buss $\mathcal{M}^b$ tais que $n \in \mathcal{M}_1 \cap \mathcal{M}_2$, $\mathcal{M}_1 \simeq_{(n)^{\mathcal{M}_1}} \mathcal{M}_2$, $\mathcal{M}_1 \preccurlyeq_{(2^{n^k})^{\mathcal{M}_1}} \mathcal{M}_2$ e existe $m \leq (2^n)^{\mathcal{M}_1}$ para o qual $\mathcal{M}_1 \vDash \psi(m)$ e $\mathcal{M}_2 \vDash \neg\psi(m)$. Sabemos que toda função numérica computável tem uma Σ_1 definição na linguagem aritmética, i.e., existe uma Σ_1-fórmula ϕ da linguagem aritmética de Peano tal que, para todos $m_1, \ldots, m_k, n \in \mathbb{N}$, $f(m_1, \ldots, m_k) = n$ se, e somente se, $\mathcal{N} \vDash \phi(m_1, \ldots, m_k, n)$. Ora, as funções $|x|$, $2^{|x|\times|y|}$ e $\lfloor \frac{x}{2} \rfloor$ da aritmética de Buss são todas computáveis e, consequentemente, a fórmula $\psi(x)$ acima define um conjunto computável A. Assim, deve existir uma Σ_1-fórmula $\phi(x)$ da linguagem aritmética de Peano que define A. Como na demonstração do teorema 5.4.1, usamos apenas funções definíveis na linguagem da aritmética de Peano, podemos definir os submodelos $\mathcal{M}_1$ e $\mathcal{M}_2$ em um modelo arbitrário enumerável não-padrão da aritmética de Peano $\mathcal{M}$. Logo, existem submodelos $\mathcal{M}_1$ e $\mathcal{M}_2$ de $\mathcal{M}$ tais que $n \in \mathcal{M}_1 \cap \mathcal{M}_2$, $\mathcal{M}_1 \simeq_{(n)^{\mathcal{M}_1}} \mathcal{M}_2$, $\mathcal{M}_1 \preccurlyeq_{(2^{n^{\mathbb{N}}})^{\mathcal{M}_1}} \mathcal{M}_2$ e existe $m \leq (2^n)^{\mathcal{M}_1}$ para o qual $\mathcal{M}_1 \vDash \phi(m)$ e $\mathcal{M}_2 \vDash \neg\phi(m)$. □

Corolário 5.4.1. *Seja $\mathcal{M}$ um modelo não-padrão enumerável da aritmética. Os seguintes enunciados são equivalentes:*

1. *$P \neq NP$.*
2. *Existe um número não-padrão $n \in \mathcal{M}$ e existe $m \leq 2^n$ tal que $n^k \equiv_P m$ mas $n^k \not\equiv_{\Sigma_1} m$ para todo $k \in \mathbb{N}$.*

A construção dos modelos $\mathcal{M}_1$ e $\mathcal{M}_2$ usando apenas funções definíveis na linguagem da aritmética de Peano foi crucial na demonstração do teorema 5.4.2. Apesar da demonstração aqui apresentada ser substancialmente diferente daquela apresentada por Máté-Sureson, essa caracterização local do

problema $P \stackrel{?}{=} NP$ é exatamente igual à caracterização do $NP \stackrel{?}{=} coNP$ apresentada por Máté-Sureson, exceto pelo fato de que no caso de $NP \stackrel{?}{=} coNP$ o enunciado (2) envolve Δ_0-fórmulas. Isso está de acordo com o fato de que $NP \neq coNP$ implica $P \neq NP$, mas a recíproca não é verdadeira.

5.5 Uma barreira aritmética em complexidade computacional

A caracterização global do problema $P \stackrel{?}{=} NP$ mostra que a possibilidade de separar as classes P e NP é equivalente à possibilidade de modificar a adição ou multiplicação de um modelo não-padrão enumerável da aritmética $\mathcal{M}_1$ de modo a gerar um outro modelo $\mathcal{M}_2$ que não seja elementarmente equivalente a $\mathcal{M}_1$, apesar de ambos serem polinomialmente indistinguíveis por máquinas de Turing. A caracterização local especifica que essa modificação deve ser feita entre $n^{\mathbb{N}}$ e 2^n, sendo n o número não-padrão que indica o ponto até o qual os modelos $\mathcal{M}_1$ e $\mathcal{M}_2$ são idênticos. Máté apresentou dois métodos que indicam a possibilidade de fazermos isso. O primeiro foi a técnica de interpolação de T.J. Grilliot [Grilliot(1985)], que lhe permitiu construir modelos não-padrão na linguagem da adição e divisão nos quais essas operações podem apresentar comportamentos distintos quando aplicadas a números não-padrão suficientemente grandes. O outro método foi a construção de ultrapotências limitadas de S. Kochen e S. Kripke [Kochen and Kripke(1982)], a qual lhes permitiu definir modelos não-padrão da aritmética para mostrar que o enunciado de Paris-Harrington é independente da aritmética de Peano.

O método de interpolação de T. J. Grilliot aplica-se, todavia, à linguagem da adição e divisão, não sendo claro como podemos estendê-lo para a linguagem da aritmética. Mais do que isso, a abordagem de Grilliot depende de uma caracterização sintática dos modelos a serem construídos. Isso significa que para demonstrar que $P \neq NP$, segundo o método de Grilliot, precisamos saber como separar as classes P e NP. Já a abordagem de Kochen e Kripke parece ser mais promissora, porque ela consiste numa modificação

da técnica algébrica de ultraprodutos que gera, sem uso de componentes sintáticos, um modelo da aritmética de Peano não elementarmente equivalente ao modelo padrão. Noutras palavras, o método de Kochen e Kripke permite a aplicação de técnicas algébricas ao problema $P \stackrel{?}{=} NP$ e, portanto, parece que poderíamos usá-lo para separar as classes P e NP. No que se segue mostraremos que isso não é verdade, porquanto sejam necessárias modificações no método de Kochen e Kripke que dependem de uma análise profunda do próprio problema $P \stackrel{?}{=} NP$.

A técnica algébrica de ultraprodutos quando aplicada ao modelo padrão $\mathcal{N}$ gera a ultrapotência $\mathcal{N}^I/U$ que permite construir modelos não-padrão da aritmética de Peano. No entanto, $\mathcal{N}^I/U$ é elementarmente equivalente a $\mathcal{N}$ e, consequentemente, essa ultrapotência não pode ser usada para demonstrar resultados de independência. Assim, S. Kochen e S. Kripke em [Kochen and Kripke(1982)] modificaram um pouco a técnica de ultraprodutos a fim de gerar um modelo aritmético não elementarmente equivalente ao modelo padrão. A estratégia de Kochen e Kripke foi encontrar um subconjunto $\mathcal{N}^{I^*}$ de $\mathcal{N}^I$ composto por funções computáveis $f : I \to \mathcal{N}$ tal que a ultrapotência $\mathcal{N}^{I^*}/U$ é um modelo da aritmética de Peano capaz de ser modificado, tornando-se um modelo não elementarmente equivalente a $\mathcal{N}$. Kochen e Kripke denominaram essa ultrapotência $\mathcal{N}^{I^*}/U$ de *ultrapotência restrita*. A partir da construção dessa ultrapotência restrita, o enunciado que Kochen e Kripke mostraram ser independente da aritmética da Peano é uma conseqüência do enunciado que J. Paris e L. Harrington [Paris and Harrington(1977)] demonstraram ser independente da aritmética de Peano[7].

Para construir a ultrapotência restrita $\mathcal{N}^{I^*}/U$, Kochen e Kripke estabeleceram uma condição mínima de fecho. Antes de enunciá-la, convencionamos que, dadas $f \in \mathcal{N}^I$ e $g \in \mathcal{N}^I$, escreveremos $f \leq g$ para indicar que $f(i) \leq g(i)$

[7] A inovação de Kochen e Kripke, relativamente ao trabalho original de Paris e Harrington, foi ter obtido uma demonstração direta, sem passar pelo segundo teorema da incompletude de Gödel.

para todo $i \in I$. A condição mínima das técnicas desenvolvidas por Kochen e Kripke é que o conjunto $\mathcal{N}^{I^*}$ seja fechado sob $\leq$, ou seja, se $f \leq g \in \mathcal{N}^{I^*}$ então $f \in \mathcal{N}^{I^*}$. Uma das propriedades importantes de ultraprodutos que a ultrapotência restrita $\mathcal{N}^{I^*}/U$ satisfaz é o teorema de Loś para Δ_0-fórmulas. Abaixo apresentamos essa demonstração com alguns ligeiros detalhes adicionais. Convencionamos denotar as funções em $\mathcal{N}^{I^*}$ por $\vec{a}$, uma vez que elas são de fato seqüências de elementos de $\mathcal{N}$ indexados por I - essa convenção torna a demonstração mais clara.

Proposição 5.5.1 (Teorema de Loś restrito [Kochen and Kripke(1982)])**.** *Seja $\mathcal{N}^{I^*}/U$ uma ultrapotência restrita fechada sob $\leq$. Dessa forma, para toda Δ_0-fórmula $\phi(x_0, \ldots, x_{n-1})$ e n-tupla $(\vec{a}_0/U, \ldots, \vec{a}_{n-1}/U)$ de elementos de $\mathcal{N}^{I^*}/U$,*

$$\mathcal{N}^{I^*}/U \vDash \phi[\vec{a}_0/U, \ldots, \vec{a}_{n-1}/U] \text{ se, e somente se,}$$
$$\{i \in I : \mathcal{N} \vDash \phi[\vec{a}_0(i), \ldots, \vec{a}_{n-1}(i)]\} \in U.$$

Demonstração. A demonstração é similar ao argumento do teorema Loś, exceto para as fórmula quantificadas. Em especial, o caso delicado é quando ϕ é da forma $\exists x_{n-1} \prec \tau\psi(x_0, \ldots, x_{n-2}, x_{n-1})$. Nesse caso, suponha que $S = \{i \in I : \mathcal{N} \vDash \exists x_{n-1} \prec \vec{b}(i)\psi(\vec{a}_0(i), \ldots, \vec{a}_{n-2}(i), x_{n-1})\} \in U$. Daí, para cada $i \in S$, existe um número $a_i < \vec{b}(i)$ tal que $\mathcal{N} \vDash \psi[\vec{a}_0(i), \ldots, \vec{a}_{n-2}(i), a_i]$. Considere as sequências $\vec{c} : \mathbb{N} \to \mathbb{N}$ tais que

$$\vec{c}(i) = \begin{cases} a_i & \text{se } i \in S \\ 0 & \text{se } i \notin S \end{cases}$$

Uma vez que $\vec{c} < \vec{b} \in \mathcal{N}^{I^*}$ temos $\vec{c} \in \mathcal{N}^{I^*}$ e, então, $\{i \in I : \mathcal{N} \vDash \psi[\vec{a}_0(i), \ldots, \vec{a}_{n-2}(i), \vec{c}(i)]\} \in U$. Pela hipótese de indução $\mathcal{N}^{I^*}/U \vDash \psi[\vec{a}_0/U, \ldots, \vec{a}_{n-2}/U, \vec{c}/U]$ e, portanto, $\mathcal{N}^{I^*}/U \vDash \exists x_{n-1} \prec \vec{b}\psi[\vec{a}_0/U, \ldots, \vec{a}_{n-2}/U, x_{n-1}]$. Conversamente, suponha que $\mathcal{N}^{I^*}/U \vDash \exists x_{n-1} \prec \vec{b}/U\psi[\vec{a}_0/U, \ldots, \vec{a}_{n-2}/U, x_{n-1}]$. Então existe uma sequência $\vec{c} < \vec{b} \in \mathcal{N}^{I^*}$ tal que $\mathcal{N}^{I^*}/U \vDash \psi[\vec{a}_0/U, \ldots, \vec{a}_{n-2}/U, \vec{c}/U]$. Pela hipótese de indução, $\{i \in$

$I : \mathcal{N} \vDash \psi[\vec{a}_0(i), \ldots, \vec{a}_{n-2}(i), \vec{c}(i)]\} \in U$. Uma vez que $\{i \in I : \mathcal{N} \vDash \psi[\vec{a}_0(i), \ldots, \vec{a}_{n-2}(i), \vec{c}(i)]\} \subseteq \{i \in I : \mathcal{N} \vDash \exists x_{n-1} < \vec{b}(i)\psi(\vec{a}_0(i), \ldots, \vec{a}_{n-2}(i), x_{n-1})\}$ e U é um ultrafiltro, $\{i \in I : \mathcal{N} \vDash \exists x_{n-1} < \vec{b}(i)\psi(\vec{a}_0(i), \ldots, \vec{a}_{n-2}(i), x_{n-1})\} \in U$. □

Uma conseqüência simples do teorema de Loś restrito, mas bastante reveladora para os nossos propósitos, é a seguinte.

Corolário 5.5.1. *Seja $\mathcal{N}^{I^*}/U$ uma ultrapotência restrita fechada sob $\leq$. Então, $\mathcal{N} \equiv_{\Delta_0} \mathcal{N}^{I^*}/U$.*

Demonstração. Defina a função $e : \mathcal{N} \to \mathcal{N}^{I^*}/U$, chamada *imersão diagonal*, tal que $e(a) = \vec{a}/U$ onde $\vec{a}(i) = a$ para todo $i \in I$. Mostraremos que e é uma imersão Δ_0-elementar de $\mathcal{N}$ para $\mathcal{N}^{I^*}/U$. Seja $\phi(x_0, \ldots, x_{n-1})$ uma Δ_0-fórmula e $a_0, \ldots, a_{n-1}$ números naturais em $\mathcal{N}$. Pela definição de ultrapotência, $\mathcal{N} \vDash \phi[a_0, \ldots, a_{n-1}]$ se, e somente se, $\{i \in I : \mathcal{N} \vDash \phi[a_0, \ldots, a_{n-1}]\} \in U$. Considerando $\vec{a} \in \mathcal{N}^{I^*}$ como acima, isso acontece se, e apenas se, $\{i \in I : \mathcal{N} \vDash \phi[\vec{a}_0(i), \ldots, \vec{a}_{n-1}(i)]\} \in U$ e, devido ao teorema de Loś restrito, isso significa que $\mathcal{N}^{I^*}/U \vDash \phi[\vec{a}_0/U, \ldots, \vec{a}_{n-1}/U]$. Portanto, $\mathcal{N} \vDash \phi[a_0, \ldots, a_{n-1}]$ se, e somente se, $\mathcal{N}^{I^*}/U \vDash \phi[e(a_0), \ldots, e(a_{n-1})]$, i.e., $\mathcal{N} \equiv_{\Delta_0} \mathcal{N}^{I^*}/U$. □

Os modelos definidos por Kochen e Kripke, nos quais se constata a falsidade do enunciado de Paris-Harrigton, são ultrapotências restritas $\mathcal{N}^{I^*}/U$ fechadas sob $\leq$. Ora, o corolário 5.5.1 assere que esses modelos são elementarmente Δ_0-equivalentes ao modelo padrão $\mathcal{N}$. Noutras palavras, eles não podem ser usados para definir modelos não-padrão em que alguma Δ_0-fórmula seja falsa. No entanto, a equivalência de Máté-Sureson envolve justamente a possibilidade de construir tais modelos. Conseqüentemente, ultrapotências restritas $\mathcal{N}^{I^*}/U$ fechadas sob $\leq$ não podem ser usadas para demonstrar que $NP \neq coNP$.

Note que nesta tese apresentamos uma caracterização local do problema $P \stackrel{?}{=} NP$, a qual envolve Σ_1-fórmulas. O seguinte resultado mostra que a possibilidade de usarmos ultrapotências restritas para resolver o problema $P \stackrel{?}{=} NP$ também não é apropriada.

Corolário 5.5.2. *Seja $\mathcal{N}^{I^*}/U$ uma ultrapotência restrita fechada sob $\leq$. Dessa forma, para toda Σ_1-fórmula $\phi(x_0,\ldots,x_{n-1})$ e n-tupla $(\vec{a}_0/U,\ldots,\vec{a}_{n-1}/U)$ de elementos de $\mathcal{N}^{I^*}/U$, se $\mathcal{N}^{I^*}/U \vDash \phi[\vec{a}_0/U,\ldots,\vec{a}_{n-1}/U]$, então $\{i \in I : \mathcal{N} \vDash \phi[\vec{a}_0(i),\ldots,\vec{a}_{n-1}(i)]\} \in U$.*

Demonstração. A demonstração é análoga ao argumento do teorema de Loś restrito, basta retirar o limitante superior na fórmula limitada. □

Observação 5.5.1. *Como $\mathcal{N}^{I^*}$ não contem todas as funções de I em $\mathcal{N}$, sabemos que a recíproca do corolário 5.5.2 não é verdadeira.*

Noutras palavras, o corolário 5.5.2 mostra que existe uma *barreira modelo-aritmética* para a separação das classes de complexidade P e NP: *qualquer técnica de separação dos modelos aritméticos não-padrão que satisfaz a condição de fecho sob $\leq$ não pode ser usada para resolver o problema $P \stackrel{?}{=} NP$.* Por exemplo, esse é o caso com respeito às técnicas na linha de J. Paris e L. Harrington [Paris and Harrington(1977)].

A fim de esclarecermos o que entendemos por *barreira aritmética*, convém mencionar as três barreiras técnicas já conhecidas para a solução do problema $P \stackrel{?}{=} NP$.

A primeira é a *relativização*. Qualquer técnica de separação de classes de complexidade que possa ser aplicada uniformente num universo com oráculos sem considerar as características desses oráculos é uma técnica *relativa*. Por exemplo, o método de diagonalização é relativo. T. Baker, J. Gill e R. Solovay demonstraram em [Baker et al.(1975)Baker, Gill, and Solovay] que existem oráculos tanto para $P = NP$ quanto para $P \neq NP$. Noutras palavras, técnicas relativas não resolverão o problema $P \stackrel{?}{=} NP$.

A segunda é a *naturalização*. Qualquer técnica de separação de classes de complexidade que esteja baseada na construção de funções booleanas com as propriedades combinatórias naturais de Razborov e Rudich [Razborov and Rudich(1997)] é uma técnica *natural*. Por exemplo, as técnicas combinatórias sobre limitantes inferiores no tamanho de cir-

cuitos[8]. Em termos gerais, podemos dizer que A. A. Razborov, S. Rudich [Razborov and Rudich(1997)] demonstram que se for possível separar as classes P e NP usando técnicas naturais, então existem algoritmos rápidos para distinguir funções aleatórias de funções pseudo-aleatórias e, conseqüentemente, também podemos encontrar algoritmos rápidos para problemas que tencionamos demonstrar serem difíceis. Isso é um indício de que técnicas naturais também não resolverão o problema $P \stackrel{?}{=} NP$.

A terceira é a *algebrização*. Segundo S. Aaronson e A. Wigderson [Aaronson and Wigderson(2009)], um *oráculo algébrico* é um oráculo que pode avaliar não apenas uma função booleana f mas também uma extensão de grau inferior f^* de f sobre um corpo finito ou sobre o anel dos inteiros. Qualquer técnica de separação de classes de complexidade que use oráculos algébricos é *algébrico-relativa*. Por exemplo, a técnica de aritmetização, usada por A. Shamir em [Shamir(1992)] para demonstrar que $IP = PSPACE$, reinterpreta cada fórmula boolena ϕ em um polinômio de grau inferior ϕ^* sobre um corpo ou anel, ou seja, a aritmetização de Shamir é uma técnica algébrica. Aaronson e Wigderson em [Aaronson and Wigderson(2009)] mostraram que qualquer técnica algébrico-relativa não pode ser usada para separar as classes P e NP.

Uma explicação detalhada de cada uma dessas barreiras demandaria um trabalho à parte. Contudo, esse esboço é suficiente para verificarmos que o tipo de barreira aritmética que estamos apontando tem natureza diferente da relativização, naturalização e algebrização. Mostramos que qualquer técnica modelo-teórica relacionada aos resultados de independência aritmética na linha de Paris e Harrington não pode ser usada para a separação das classes de complexidade P e NP, sem ajustes que talvez impossibilitem o próprio uso dessas técnicas.

[8] Cf. [Sipser(1992)] para detalhes.

6. CONCLUSÃO

Nulla si sa, tutto s'immagina.
Federico Fellini

6.1 Análise dos resultados

No capítulo 2 formulamos as bases de uma abordagem modelo-teórica da computabilidade clássica, usando lógica de primeira ordem. Os resultados de correspondência, correção, representação e completude entre máquinas, estruturas e teorias de Turing indicam a adequação das formulações desenvolvidas. Destaca-se o fato de que tenhamos obtido a completude entre estruturas e teorias de Turing, ao introduzirmos o princípio de indução em teorias de Turing.

Se teorias de Turing tivessem uma capacidade expressiva equivalente à aritmética de Robinson, por exemplo, elas seriam incompletas - pelo primeiro teorema da incompletude de Gödel. Uma vez que teorias de Turing são completas com relação às estruturas de Turing, podemos dizer que elas são fracas face aos aspectos numéricos que compõem as estruturas de Turing. Segundo H. Comon, Y. Jurski [Comon and Jurski(1998)], o fecho transitivo de computações de máquinas com registro é definível na aritmética de Presburger. Noutras palavras, a aritmética da adição é suficiente para expressarmos as computações de máquinas de Turing. Isso está em conformidade com nossa afirmação de que teorias de Turing têm expressividade fraca.

Um ponto intrigante é que a expressividade restrita de teorias de Turing permite-lhes internalizar um predicado de satisfatibilidade tarskiano

[Tarski(1983)][1]. Como determinamos se uma máquina de Turing $\boldsymbol{M}$ computa uma dada função f? Por definição, devemos verificar que $\boldsymbol{M}$ concorda com f em todas as entradas e saídas. Mas como verificamos que $\boldsymbol{M}$ concorda com f em *todas* as entradas e saídas? É claro que fazemos uma espécie de indução na quantidade de entradas. A constatação de que $\boldsymbol{M}$ concorda ou não com f em uma entrada é, por sua vez, imediata. Nesse sentido, podemos afirmar que o princípio de indução em teorias de Turing é uma consequência da capacidade de determinarmos que $\boldsymbol{M}$ computa f e a capacidade de internalização da satisfatibilidade em teorias de Turing é uma conseqüência da capacidade de constatarmos imediatamente que $\boldsymbol{M}$ concorda com f relativamente a uma entrada particular.

No capítulo 3 analisamos as características da computabilidade clássica de um ponto de vista modelo-teórico. Os resultados sobre estabilidade, absoluticidade, universalidade e logicidade permitem-nos sustentar que a lógica de primeira ordem é suficiente para os propósitos da computabilidade clássica. Em especial, as demonstrações de que teorias de Turing têm uma lógica minimal intuicionista e também são capazes de internalizar um operador de negação clássico merecem destaque.

D. Hilbert [Hilbert(1900), Hilbert(1967)] formulou seu programa fundacional da matemática, chamado *programa de Hilbert*[2], porque acreditava que existe uma parte pré-teórica da matemática assentada apenas na intuição pura de signos concretos. Por isso, Hilbert pensou que, conquanto as operações com conceitos abstratos possam ser questionadas como incertas, a parte pré-teórica da matemática seria inquestionável, porque ela consiste apenas de numerais finitários, seqüências de traços que não representam nada, mas sobre os quais podemos efetuar operações tais como concatenação, comparação, sucessão, e assim por diante. Esse domínio é tão elementar que não possui contradições, simplesmente porque não existe estrutura lógica

[1] S. Kripke [Kripke(1996)] comenta algo nesse sentido com respeito a teorias das funções recursivas.

[2] Cf. [Zach(2003)]

nas proposições desse âmbito. A partir desse âmbito pré-matemático, Hilbert propôs, então, construir a metamatemática, a qual trataria de símbolos e seqüências finitas desses símbolos - sentenças e demonstrações seriam consideradas como seqüências de símbolos. Hilbert era tão confiante na capacidade da metamatemática formulada dessa forma que chegou a pensar que sistemas tais como o *Principia Mathematica* de B. Russell e A. N. Whitehead [Russell and Whitehead(1910, 1912, 1913)] poderiam ser formalizados a partir da parte pré-teórica da matemática. Se tal afirmação fosse o caso, o problema da consistência geral da matemática passaria a residir na demonstração de que sistemas metamatemáticos tais como o *Principia Mathematica* não seriam inconsistentes.

Posto que Máquinas de Turing sejam dispositivos de manipulação de seqüências de traços, faz sentido afirmar que elas operam justamente no âmbito pré-matemático tematizado por Hilbert. A possibilidade de desenvolvermos a computabilidade clássica através da lógica de primeira ordem confirma, portanto, a perspectiva de Hilbert. A constatação de que teorias de Turing têm uma lógica minimal intuicionista concorda com a ênfase construtivista que Hilbert defendeu ser inerente ao âmbito pré-matemático. Por fim, a capacidade de teorias de Turing internalizarem um operador de negação clássico mostra que, apesar de não existir estrutura lógica nas proposições no âmbito das computações de máquinas de Turing, podemos internalizar os aspectos metalógicos através da própria estrutura aritmética envolvida nas computações. Dessa forma, parece filosoficamemte defensável que, se quisermos argumentar pela invalidade da tese de Hilbert, outro tipo de análise deve ser buscada.

No capítulo 4 investigamos a computabilidade de Turing em modelos não-padrão da aritmética. A demonstração do teorema de Tennenbaum bem como a avaliação da demonstração da tese de Turing proposta por Dershowitz e Gurevich indicam que a computabilidade de Turing sobre números não-padrão deve ser levada em conta quando consideramos os limites da com-

putação. Mais do que isso, a descoberta do princípio de internalidade aritmética na computabilidade mostra que o próprio conceito de computação é relativo ao modelo aritmético no qual as máquinas de Turing operam.

O princípio de internalidade aritmética não afirma que as características do conceito de computação mudam conforme o modelo aritmético considerado. Ele apenas assere que as máquinas de Turing computam sempre sobre um horizonte aritmético, o qual é pressuposto. O âmbito pré-matemático de Hilbert, composto apenas por símbolos, não é desprovido de significação, pois máquinas de Turing computam sobre símbolos que sempre são interpretados em algum modelo aritmético. L. Bellotti [Bellotti(2007)] tem argumentado que a possibilidade de instabilidade na noção de finitude, derivada da existência de modelos não-padrão da aritmética, interfere na sintaxe de sistemas formais. Noutras palavras, o âmbito pré-matemático de Hilbert não é independente da perspectiva metamatemática que temos dele. O princípio de internalidade aritmética da computabilidade enfatiza que a computação depende das características lógicas da teoria aritmética. Somente com teorias aritméticas de segunda ordem podemos garantir que o universo da computabilidade envolve apenas números naturais padrão.

Em virtude de números reais poderem ser representados como conjuntos infinitos de números naturais - por exemplo, através de cortes de Dedekind de racionais - e a aritmética de segunda ordem permitir quantificação sobre tais conjuntos, podemos formalizar os números reais na aritmética de segunda ordem [Simpson(2009)]. Ora, os números reais estão entre os *objetos ideais* cujo uso na matemática Hilbert pensou ser justificável através do âmbito pré-matemático. Conforme S. Shapiro [Shapiro(1991)], não é necessário apelar para a aritmética de segunda ordem plena, basta a aritmética de segunda ordem fraca (com quantificação apenas sobre conjuntos finitos). Não obstante, a semântica da lógica de segunda ordem fraca pressupõe a noção de conjuntos das partes de um conjunto dado, ou seja, precisamos da teoria dos conjuntos. De qualquer forma, portanto, o que precisamos para garantir que os números

naturais padrão são os elementos básicos da computabilidade está para além do que Hilbert aceitaria como constituinte do âmbito pré-matemático.

No capítulo 5 apresentamos caracterizações modelo-aritméticas do problema P versus NP. As técnicas desenvolvidas nesta tese permitiram demonstrações simples e unificadas das caracterizações global e local desse problema. Os resultados sobre os limites de técnicas modelo-teóricas apresentados no capítulo 5 indicam certa barreira aritmética para a possibilidade de solução do problema $P \stackrel{?}{=} NP$.

Se adicionarmos à barreira aritmética descoberta nesta tese os aspectos aritméticos paradoxais apontados por S. Kurtz, M. J. O'Donnell e S. Royer [Kurtz et al.(1987)Kurtz, O'Donnell, and Royer], S. Ben-David e S. Halevi [Ben-David and Halevi(1991)] acerca da complexidade computacional, podemos dizer que o problema $P \stackrel{?}{=} NP$ pode estar para além de nossas capacidades dedutivas. Seguindo S. Kurtz, M. J. O'Donnell e S. Royer [Kurtz et al.(1987)Kurtz, O'Donnell, and Royer], S. Ben-David e S. Halevi [Ben-David and Halevi(1991)] mostraram que $P \stackrel{?}{=} NP$ é independente da aritmética de Peano se, e somente se, as classes P e NP são praticamente indistinguíveis. Mais precisamente, $P \neq NP$ é independente da aritmética de Peano se, e somente se, o problema $3\text{-}SAT$ está em $DTIME(n^{f^{-1}(n)})$, onde f não é dominada pela hierarquia de Wainer - funções na hierarquia de Wainer tem taxa de crescimento maior do que a função de Ackermann. Assim, cabe a pergunta: como poderíamos construir modelos não-padrão da aritmética para demonstrar que $P \neq NP$, usando recursos formalizáveis na própria aritmética?

As caracterizações global e local do problema $P \stackrel{?}{=} NP$ que desenvolvemos a partir do trabalho de J. Paris e C. Dimitracopoulos [Paris and Dimitracopoulos(1982)] e A. Maté e C. Sureson [Máté(1990), Sureson(1995)] exibem outro caráter paradoxal de uma possível solução desse problema. O corolário 5.4.1 da caracterização local mostra que $P \neq NP$ se, e somente se, formos capazes de separar dois modelos não-padrão enumeráveis

da aritmética através de uma Σ_1-fórmula conquanto eles sejam polinomialmente indistinguíveis. Ora, PA é Σ_1-completa, i.e., todas as Σ_1-fórmulas verdadeiras no modelo padrão são demonstráveis na aritmética de Peano. Dessa forma, não podemos separar dois modelos através de Σ_1-fórmulas. No entanto, também não podemos usar esse fato para demonstrar que o problema $P \stackrel{?}{=} NP$ é insolucionável, porque sua negação afirma que quaisquer modelos não-padrão enumeráveis e polinomialmente indistinguíveis devem ser elementarmente equivalentes; o que pela caracterização global 5.2.1 seria equivalente a resolver o problema $P \stackrel{?}{=} NP$.

Em um artigo recente N.C.A. da Costa e F.A. Dória [da Costa and Doria(2003), da Costa and Doria(2006)] tratam da assim chamada "formalização exótica" do problema $P \stackrel{?}{=} NP$, defendendo que tal formulação seria independente de ZFC. Em contrapartida, L. Gordeev em [Gordeev(2010)] argumenta que as conclusões de [da Costa and Doria(2003), da Costa and Doria(2006)] na verdade não são características do problema $P \stackrel{?}{=} NP$, nem são conclusivas sobre sua consistência com ZFC. A análise que fazemos nesta tese a respeito das caracterizações aritméticas do problema $P \stackrel{?}{=} NP$ talvez possa contribuir para o esclarecimento das posições em questão nesse debate. Todavia, cabe destacar que a barreira aritmética proposta nesta tese não é evidência da independência do problema $P \stackrel{?}{=} NP$.

Nesse contexto, é importante mencionar que S. Aaronson [Aaronson(2003)], ao analizar a possibilidade de solucionarmos o problema $P \stackrel{?}{=} NP$, concluiu que talvez essa questão seja independente da aritmética e não possamos nem mesmo demonstrar isso, porquanto tal feito seja equivalente a solucionar o problema $P \stackrel{?}{=} NP$[3]. Dado que no capítulo 4 mostramos que existe um princípio de internalidade aritmética na computabilidade, os resultados que formulamos no capítulo 5 apresentam

[3] Cf. [da Costa and Doria(2007)] para mais referências sobre a metamatemática do problema $P \stackrel{?}{=} NP$.

uma barreira aritmética inerente à complexidade computacional e parecem indicar algo na direção que Aaronson apontou.

Essa interpretação dos resultados obtidos no capítulo 5 implica que o âmbito pré-matemático de Hilbert pode ter características ainda mais surpreendentes do que aquelas do âmbito aritmético, apontadas pelos teoremas da incompletude de Gödel. A distinção entre objetos finitários e infinitários está, portanto, longe de ser compreendida.

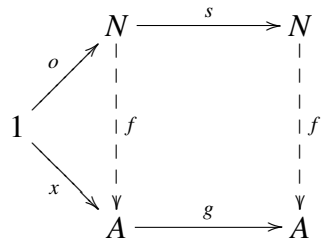
6.2 Trabalhos futuros

A idéia de abordar a computabilidade através dos métodos da teoria dos modelos tem inúmeros desdobramentos. No que se segue apresentamos quatro linhas de pesquisa que no momento nos parecem relevantes.

Uma linha de pesquisa seria *investigar as conseqüências da computabilidade não-padrão para resultados fundacionais.* D. Isles [Isles(1981), Isles(1992), Isles(2009)] tem sustentado que a validade de resultados importantes, tais como os teoremas da incompletude de Gödel e o problema da parada, pressupõem os números naturais padrão como modelo intencionado da aritmética. Isso parece ser corroborado pelos resultados de M. M. Richter e M. E. Szabo [Richter and Szabo(1983)] bem como P. H. Potgieter e E. E. Rosinger [Potgieter and Rosinger(2008)] acerca do problema da parada. As teses de Church e Turing, como já indicamos, parecem não depender do modelo aritmético considerado, mas suas possíveis demonstrações talvez dependam. A tese de Church e o princípio de Markov são realizáveis em topos efetivo [van Oosten(2008), cap. 3], isso porque, para toda categoria (pequena) $\mathcal{C}$, a categoria dos funtores conjuntisticamente valorados $\mathcal{S}et^{\mathcal{C}}$ tem objeto número natural.

Usualmente, em categorias com objeto terminal 1 definimos que um *objeto número natural* é um objeto N com morfismos $1 \xrightarrow{o} N \xrightarrow{s} N$ tal que, para qualquer objeto A e morfismos $1 \xrightarrow{x} A \xrightarrow{g} A$, existe um único morfismo f :

$N \to A$ para o qual o diagrama abaixo comuta:



Em $\mathcal{S}et^{\mathcal{C}}$, considerando $1 = \{0\}$ e $\mathbb{N} = \omega$, para o diagrama do objeto número natural comutar, o morfismo f deve ser tal que $f(0) = x(0)$ e $f \circ s(n) = g \circ f(n)$, ou seja, f é definido por *recursão* em x e g. Por isso, os morfismos da forma $f : N \to A$ são chamados *sequências*. A definição de objeto número natural parece, portanto, não excluir a possibilidade de considerarmos números naturais não-padrão. Ora, unificações dos resultados limitativos baseados nos diagramas de Cantor [Noson(2003), Germano and Mazzanti(2003)], entre eles o problema da parada, envolvem justamente o objeto número natural. Portanto, é preciso uma investigação mais detalhada para verificar se resultados fundacionais realmente dependem do modelo intencionado da aritmética.

Outra linha de pesquisa associada à computabilidade não-padrão seria *examinar os modelos de hipercomputação de um ponto de vista modelo-téorico*. Em geral, os modelos de hipercomputação agregam capacidades infinitárias às máquinas de Turing [Loff and Costa(2009), Copeland(2004)]. Em especial, as máquinas de Turing com tempo infinito definidas por J. Hamkins and A. Lewis [Hamkins and Lewis(2000)] suscitam uma análise modelo-teórica. A razão para isso é que R. D. Schindler [Schindler(2003)] demonstrou que $P \neq NP$ para essas máquinas e, ulteriormente, V. Deolalikar *et alli* [Deolalikar et al.(2005)Deolalikar, Hamkins, and Schindler] também mostraram que $P \neq NP \cap coNP$ para máquinas de Turing com tempo infinito. Como sugerido em [de Araújo(2010a)], seria interessante investigar as características infinitárias dessas máquinas de Turing, relacioná-las com as máquinas de Turing não-padrão e, assim, verificar quais conseqüências

poderíamos derivar para os problemas de complexidade computacional nas máquinas de Turing com tempo padrão e não-padrão.

Algumas diferenças entre máquinas de Turing com tempo infinito e máquinas de Turing com instantes não-padrão são patentes. Nas primeiras os instantes são ordinais transfinitos, mas nas segundas os instantes são internamente ordinais finitos, ao passo que externamente não são ordinais - externamente os números naturais não-padrão não são bem-ordenados. Por isso, as técnicas usadas em [Schindler(2003)] e [Deolalikar et al.(2005)Deolalikar, Hamkins, and Schindler] para a separação de classes de complexidade infinitárias não se aplicam diretamente no caso de máquinas de Turing com instantes não-padrão. Talvez algumas modificações permitam fazer isso. De qualquer forma, a própria investigação sobre essas distinções pode ser relevante para a compreensão das classes de complexidade no âmbito finitário.

Uma linha de pesquisa vinculada à idéia geral de uma abordagem modelo-teórica da computabilidade seria *expandi-la para as máquinas de Turing com oráculos*. Essas máquinas foram concebidas por A. Turing [Turing(1939)] com o intuito de estudar sistemas que demonstram fórmulas indecidíveis na aritmética. A partir do trabalho de E. Post [Post(1944)], as investigações sobre os graus de Turing, gerados por máquinas de Turing com oráculos, passaram a ser predominantes na teoria da computabilidade. Na década de 1970, intensificou-se o tratamento da teoria de primeira ordem dos graus de Turing [Odifreddi(1992a), p.530-550], a fim de especificar as características globais do universo de Turing $\mathcal{D}$, formado por classes de equivalência entre problemas insolúveis com mesmo grau de Turing.

B. Cooper [Cooper(1997a)] e, mais recentemente B. Cooper e P. Odifreddi [Cooper and Odifreddi(2003)], têm sustentado que as dificuldades na unificação de nossas teorias físicas do universo podem estar relacionadas ao fato do universo de Turing $\mathcal{D}$ não ser uma estrutura rígida; conforme proposto por Cooper em [Cooper(1997b)]. Segundo R. A. Shore [Shore(2006), p.384],

apesar dos esforços de Cooper para explicar sua proposta de demonstração da não rigidez de $\mathcal{D}$, ninguém conseguiu entender a construção apresentada em [Cooper(1997b)]. Desse modo, a conjectura sobre a rigidez de $\mathcal{D}$ permanece como um problema em aberto. Talvez uma análise modelo-teórica das máquinas de Turing com oráculos possa contribuir para o esclarecimento das técnicas propostas em [Cooper(1997b)].

A dificuldade aqui está no fato de que oráculos são objetos externos às máquinas de Turing. Assim, não é óbvio como poderíamos definir esse modelo de computação. Ademais, também não está claro em [Cooper(1997b)] qual é a relação entre a teoria de primeira ordem de $\mathcal{D}$ e a estrutura das máquinas de Turing com oráculos. Talvez uma abordagem modelo-téorica similar a que desenvolvemos nesta tese para máquinas de registros possa ser extendida para conter oráculos, tal como J. R. Shöenfield abordou a computabilidade relativa em [Shöenfield(1993), p.39-40]. Se Cooper e Odifreddi [Cooper and Odifreddi(2003)] estiverem corretos acerca das conseqüências epistêmicas da não rigidez de $\mathcal{D}$, então, apesar das dificuldades, convém tentar desenvolver esse estudo.

Para finalizar, outra linha de pesquisa associada ao uso de métodos da teoria dos modelos em computabilidade seria *desenvolver uma abordagem modelo-téorica das máquinas de Turing quânticas*. Em 1985, D. Deutsch [Deutsch(1985)] introduziu essas máquinas como modelo preciso de computação baseada na mecânica quântica. Mais tarde, Deutsch [Deutsch(1989)] também definiu os *circuitos quânticos*. Uma vez que A. Yao [Yao(1993)] demonstrou que esses dois modelos de computação quântica são equivalentes, hoje existe uma preferência pelos circuitos quânticos [Nielson and Chuang(2000)], devido à sua simplicidade.

Em contrapartida, M. Ying [Ying(2005)] sugeriu desenvolver uma análise das máquinas de Turing quânticas atráves da lógica quântica de G. D.Birkhoff e J. von Neumann [Birkhoff and von Neumann(1936)], a fim de tornar o estudo das máquinas de Turing quânticas mais palatável. Em

[Agudelo and Carnielli(2010)], J. Agudelo e W. Carnielli também analisaram a relação entre máquinas de Turing paraconsistentes e máquinas de Turing quânticas, tendo por objetivo o esclarecimento de conceitos quânticos. Como sugerido em [de Araújo(2010b)], uma abordagem modelo-teórica das máquinas de Turing quânticas através das lógicas quânticas exógenas de [Mateus and Sernadas(2006)] ou através da semântica de Kripke associada à lógica quântica de Birkof e von Neumann[4] poderia contribuir para a compreensão da estrutura lógica das máquinas de Turing quânticas, sobretudo no que diz respeito aos problemas indicados por [Shi(2002), Fouché et al.(2008)Fouché, Heidema, Jones, and Potgieter] acerca da universalidade quântica.

Existe uma série de desafios a uma possível abordagem modelo-téorica das máquinas de Turing quânticas. Para começar, geralmente essas máquinas são definidas em termos de espaços de Hilbert sobre o corpo dos complexos [Nielson and Chuang(2000)]. Isso significa que estruturas de Turing quânticas deverão ter um domínio composto, de alguma forma, por números complexos, ou seja, os aspectos numéricos serão mais complicados do que em máquinas de Turing clássicas. Mais do que isso, as configurações em máquinas de Turing quânticas são consideradas como superposições de configurações de máquinas de Turing clássicas. Portanto, teorias de Turing quânticas terão que conter axiomas para simular essas superposições, as quais tem características bastante intrincadas [Carpentieri(2003)].

Essas são apenas algumas diretrizes de pesquisas baseadas nas idéias propostas nesta tese. As dificuldades para desenvolvê-las são visíveis, porém, dada a relevância dos problemas em questão, parece ser importante tentar levar a cabo tais propostas.

[4] Cf. [Chiara and Giuntini(2002)].

REFERÊNCIAS BIBLIOGRÁFICAS

[Aaronson(2003)] S. Aaronson. Is p versus np formally independent? *Bulletin of the EATCS*, 81:109–136, 2003.

[Aaronson and Wigderson(2009)] S. Aaronson and A. Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 1(1):2–50, 2009.

[A.B.M. Brunner(2005)] W. C. A.B.M. Brunner. Anti-intuitionism and paraconsistency. *Journal of Applied Logic*, 3:161–184, 2005.

[Agudelo and Carnielli(2010)] J. Agudelo and W. Carnielli. Paraconsistent machines and their relation to quantum computing. *journal of Logic and Computation*, 2010.

[Arora and Barak(2009)] S. Arora and B. Barak. *Computational Complexity: A modern approach*. Cambridge Univeristy Press, Cambridge, 2009.

[Baker et al.(1975)Baker, Gill, and Solovay] T. Baker, J. Gill, and R. Solovay. Relativizations of the $p \stackrel{?}{=} np$ question. *Journal of the ACM*, 42: 401–420, 1975.

[Barendregt(1984)] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic*. North-Holland, Amsterdam, 1984.

[Barker-Plummer(2004)] D. Barker-Plummer. Turing machines. In E. Zalta, editor, *Stanford Encyclopedia of Philosophy*. 2004. avaliable at http://plato.stanford.edu/entries/turing-machine/.

[Bellotti(2007)] L. Bellotti. Formalization, syntax and the standard model of arithmetic. *Synthese*, 154:199–229, 2007.

[Ben-David and Halevi(1991)] S. Ben-David and S. Halevi. On the independence of p versus np. Technical Report 699, Technion, www.cs.technion.ac.il/shai/ph.ps.gz., 1991.

[Benacerraf(1965)] P. Benacerraf. What numbers could not be. *Philosophical Review*, 74:47–73, 1965.

[Berarducci and Otero(1996)] A. Berarducci and M. Otero. A recursive nonstandard model of normal open induction. *The journal of Symbolic Logic*, 61:1228–1241, 1996.

[Birkhoff and von Neumann(1936)] G. D. Birkhoff and J. von Neumann. The logic of quantum mechanics. *Annals of Mathematics*, 37:823–843, 1936.

[Boolos et al.(2002)Boolos, Burgess, and Jeffrey] G. Boolos, J. Burgess, and R. Jeffrey. *Computablility and Logic*. 4th ed. Cambridge Univeristy Press, Cambridge, 2002.

[Buss(1986)] S. R. Buss. *Bounded arithmetic*. Bibliopopolis, Naples, 1986.

[Carpentieri(2003)] M. Carpentieri. On the simulation of quantum turing machines. *Theoretical Computer Science*, 304:103–128, 2003.

[Chaitin(2006)] G. Chaitin. *Metamaths: The Quest for Omega*. Atlantic Books, New York, 2006.

[Chiara and Giuntini(2002)] M. L. D. Chiara and R. Giuntini. Quantum logic. In D. Gabbay and E. Guenthner, editors, *Handbook of Philosophical Logic*, volume 6 of *2ª ed*, pages 129–228. Kluwer Academic Publishers, 2002.

[Church(1933)] A. Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, 58(34):839–864, 1933.

[Church(1936)] A. Church. An unsolvable problem of elementary number theory. *American journal of Mathematics*, 58:345–363, 1936.

[Cohen(1987)] F. Cohen. Computer viruses: Theory and experiments. *Computers and Security*, 6:22–35, 1987.

[Cohen(1989)] F. Cohen. Computational aspects of computer viruses. *Computers and Security*, 8:325–244, 1989.

[Comon and Jurski(1998)] H. Comon and Y. Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In *CAV'98*, volume 1427 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 1998.

[Cook and Cogburn(2000)] R. T. Cook and J. Cogburn. What negation is not: intuitionism and '0 = 1'. *Analysis*, 60(1):5–12, 2000.

[Cook(1971)] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on the theory of computaing*, pages 151–158. 1971.

[Cooper(2004)] S. Cooper. *Computablility Theory*. Chapman and Hall/CRC mathematics, Florida, 2004.

[Cooper(1997a)] S. B. Cooper. Beyond gödel's theorem: The failure to capture information content. In A. Sorbi, editor, *Complexity, Logic and Recursion Theory*, volume 187 of *Lecture Notes in Pure and Applied Mathematics*, pages 93–122. Marcel Dekker, 1997a.

[Cooper(1997b)] S. B. Cooper. The turing universe is not rigid. Technical report 16, University of Leeds, Department of Pure Mathematics Preprint Series, 1997b.

[Cooper and Odifreddi(2003)] S. B. Cooper and P. Odifreddi. Incomputability in nature. In S. B. Cooper and S. S. Goncharov, editors, *Computability and Models*, pages 137–160. Kluwer Academic, 2003.

[Copeland(2004)] B. J. Copeland. Hypercomputation: philosophical issues. *Theoretical Computer Science*, 317:251–267, 2004.

[da Costa and Doria(2003)] N. da Costa and F. Doria. Consequences of an exotic definition for $p = np$. *Applied Mathematics and Computation*, 145:655–665, 2003.

[da Costa and Doria(2006)] N. da Costa and F. Doria. Addendum to "consequences of an exotic definition for $p = np$". *Applied Mathematics and Computation*, 172:1364–1367, 2006.

[da Costa and Doria(2007)] N. da Costa and F. Doria. On the metamathematics of the p vs. np question. *Applied Mathematics and Computation*, 38:81–92, 2007.

[D'Aquino(1997)] P. D'Aquino. Toward the limits of the tennenbaum phenomenon. *Notre Dame journal of Formal Logic*, 38:81–92, 1997.

[Davis(1956)] M. D. Davis. A note on universal turing machines. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, Annals of Mathematics Studies, pages 167–175. Princeton University Press, 1956.

[Davis(1957)] M. D. Davis. The definition of universal turing machine. *Proceedings of the American Mathematical Society*, 8(6):1125–1126, 1957.

[de Araújo(2010a)] A. de Araújo. Turing computability and nonstandard models of arithmetic. In F. Ferreira, H. Guerra, E. Mayordomo, and J. Rasga, editors, *Programs, Proofs, Processes*, Abstract and Handout Booklet of the 6th Conference on Computability in Europe (CiE2010), pages 38–49. CMATI, Lisboa, 2010a.

[de Araújo(2010b)] A. de Araújo. Universal turing machines without codification. In M. Slavkovik, editor, *Proceedings of the 15th student session of the European Summer School for Logic, Language and Information*, ESSLLI2010, pages 47–56. Copenhagen, 2010b.

[de Araújo and Carnielli(2010)] A. de Araújo and W. Carnielli. Nonstandard numbers: a semantic obstacle for modeling arithmetical reasoning. *The Logic journal of IGPL*, 2010.

[Dedekind(1888)] R. Dedekind. *Was sind und was sollen die Zahlen?* Braunschweig, 1888.

[Delvenne(2009)] J.-C. Delvenne. What is a universal computing machine. *Applied Mathematics and Computation*, 215:1368–1374, 2009.

[Deolalikar et al.(2005)Deolalikar, Hamkins, and Schindler] V. Deolalikar, J. Hamkins, and R. Schindler. $p \neq n \cap co - np$ for infinite time turing machines. *Journal of Logic and Computation*, 15(5):577–592, 2005.

[Dershowitz and Gurevich(2008)] N. Dershowitz and Y. Gurevich. A natural axiomatization of computability and proof of church's thesis. *Bulletin of Symbolic Logic*, 14(3):299–350, 2008.

[Deutsch(1985)] D. Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London*, Series A(400):97–117, 1985.

[Deutsch(1989)] D. Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London*, Series A(425):73–90, 1989.

[Dummett(1991)] M. Dummett. *The Logical Basis of Metaphysics*. Harvard University Press, Cambridge, 1991.

[Ebbinghaus and Flum(1999)] H. Ebbinghaus and J. Flum. *Finite model theory*. Spring Verlag, Berlin, 1999.

[Ebbinghaus et al.(2005)Ebbinghaus, Flum, and Thomas] H. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Spring Verlag, New York, 2005.

[Enderton(1977)] H. B. Enderton. Elements of recursion theory. In J. Barwise, editor, *Handbook of mathematical logic*, pages 527–566. Elsevier, Amsterdam, 1977.

[Fouché et al.(2008)Fouché, Heidema, Jones, and Potgieter] W. Fouché, J. Heidema, G. Jones, and P. H. Potgieter. Universality and programmability of quantum computers. *Theoretical Computer Science*, (403): 121–129, 2008.

[Fraïssé(1955)] R. Fraïssé. Sur quelques classifications des relation, basées sur des isomorphismes restreints. *Algebrique Mathematique*, 2:273–295, 1955.

[Friedman(1973)] H. M. Friedman. Countable models of set theories. In H. Rogers and A. R. D. Mathias, editors, *Lectures notes in mathematics*, volume 125, pages 539–573. Springer-Verlag, 1973.

[Gaifman(2004)] H. Gaifman. Non-standard models in a broader perspective. In A. Enayat and R. Kossak, editors, *Nonstandard Models of Arithmetic and Set Theory*, volume 361 of *Contemporary Mathematics*, pages 129–143. American Mathematical Society (AMS), New York, 2004.

[Gajardo and Mazoyer(2007)] A. Gajardo and J. Mazoyer. One head machines from a symbolic approach. *Theoretical Computer Science*, 370: 34–47, 2007.

[Gandy(1980)] R. Gandy. Church's thesis and principles for mechanisms. In ..., editor, *The Kleene symposium*, ..., pages 123–148. North-Holland, 1980.

[Gandy(1988)] R. Gandy. The confluence of ideas in 1936. In R. Herken, editor, *The Universal Turing Machine - A Half-Century Survey*, pages 51–102. Springer-Verlag, Wien, 1988.

[Gödel(1931)] K. Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.

[Gödel(1986a)] K. Gödel. On undecidable propositions of formal mathematical systems. In S. Feferman, J. W. D. Jr., S. C. Kleene, G. H. Moore, R. M. Solovay, and J. van Heijenoort, editors, *Kurt Gödel Collected Works*, volume I, pages 346–369. Oxford University Press, Oxford, 1986a.

[Gödel(1986b)] K. Gödel. Review of t. skolem's: On the imposibility of a complete characterization of the number sequence by means of a finite axiom system (1935). In S. Feferman, J. W. D. Jr., S. C. Kleene, G. H. Moore, R. M. Solovay, and J. van Heijenoort, editors, *Kurt Gödel Collected Works*, volume I, pages 376–380. Oxford University Press, Oxford, 1986b.

[Gödel(1986c)] K. Gödel. Über die länge von beweisen(1936). In S. Feferman, J. W. D. Jr., S. C. Kleene, G. H. Moore, R. M. Solovay, and J. van Heijenoort, editors, *Kurt Gödel Collected Works*, volume I, pages 396–398. Oxford University Press, Oxford, 1986c.

[Gödel(1986d)] K. Gödel. Review of church: a proof of freedon from contradiction (1936). In S. Feferman, J. W. D. Jr., S. C. Kleene, G. H. Moore, R. M. Solovay, and J. van Heijenoort, editors, *Kurt Gödel Collected Works*, volume I, pages 398–401. Oxford University Press, Oxford, 1986d.

[Germano and Mazzanti(2003)] G. M. Germano and S. Mazzanti. Cantor

diagrams: A unifying discussion of self-reference. *Applied Categorical Structures*, 11:313–336, 2003.

[Goldreich(2008)] O. Goldreich. *Computational Complexity - A Conceptual Perspective*. Cambridge University Press, Cambridge, 2008.

[Gordeev(2010)] L. Gordeev. A note on da costa-doria "exotic formalizations". *Journal Archive for Mathematical Logic*, 49:7–8, 2010.

[Grassmann(1861)] H. Grassmann. *Lehrbuch der Arithmetik für höhere Lehranstalten*. Berlin, 1861.

[Griffor(1999)] E. R. Griffor. *Handbook of Computability Theory*, volume 140 of *Studies in Logic and the Foundations of mathematics*. Elsevier, Amsterdam, 1999.

[Grilliot(1985)] T. J. Grilliot. Disturbing arithmetic. *The journal of Symbolic Logic*, 50(2):375–379, 1985.

[Gurevich(2000)] Y. Gurevich. Sequential abstract-state machines capture sequential algorithms. *ACM Transactions on Computational Logic*, 1 (1):77–111, 2000.

[Halbach and Horsten(2005)] V. Halbach and L. Horsten. Computational structuralism. *Philosophia Mathematica*, 13(2):174–186, 2005.

[Hamkins and Lewis(2000)] J. Hamkins and A. Lewis. Infinite time turing machines. *The journal of Symbolic Logic*, 65:567–604, 2000.

[Herbrand(1931)] J. Herbrand. Sur la non-contradiction de l'arithmétique. *J. Reine Angew. Math.*, (166):1–8, 1931.

[Heyting(1930)] A. Heyting. Die formalen regeln der intuitionischen logik. *Akad. Wiss. Phys. Math. Klasse*, Sitzungsberichste Preuss:42–56, 1930.

[Hilbert(1900)] D. Hilbert. Über den zahlbegriff. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, (8):180–84, 1900.

[Hilbert(1901)] D. Hilbert. Mathematische probleme. *Archiv der Mathematik und Physik*, 3(1):44–63, 213–237, 1901.

[Hilbert(1967)] D. Hilbert. Über die grundlagen der logik und der arithmetik. In J. van Heijenoort, editor, *From Frege to Gödel. A Source Book in Mathematical Logic.*, pages 367–392. 1967.

[Hinman(2005)] P. G. Hinman. *Fundamentals of mathematical logic*. A. K. Peters, Wellesley-Massachusetts, 2005.

[Hodges(2005)] W. Hodges. *Model Theory*. Encyclopedia of Mathematics and Applications. Cambridge University Press, Cambridge, 2005.

[Husserl(1901(2008))] E. Husserl. *Logische Untersuchungen*. Auf Grund des Nachlasses veröffentlicht vom Husserl-Archiv Leuven. Springer-Verlag, Berlin, 1901(2008).

[Husserl(1913(2008))] E. Husserl. *Ideen zu einer reinen Phänomenologie und phänomenologischen Philosophie*. Auf Grund des Nachlasses veröffentlicht vom Husserl-Archiv Leuven. Springer-Verlag, Berlin, 1913(2008).

[Immerman(1999)] N. Immerman. *Descriptive complexity*. Graduate texts in Computer Science. Springer Verlag, Berlin, 1999.

[Isles(1981)] D. Isles. On the notion of standard non-isomorphic natural number series. In F. Richman, editor, *Constructive Mathematics*, Lectures Notes in Mathematics, pages 111–134. Springer Verlag, Berin, 1981.

[Isles(1992)] D. Isles. What evidence is there that $2^{6}5536$ is a natural number? *Notre Dame journal of Formal Logic*, 3:465–480, 1992.

[Isles(2009)] D. Isles. First-order reasoning and primitive recursive natural number notations. *Studia Logica*, 96(1):49–64, 2009.

[Jech(2002)] T. Jech. *Set Theory*. Spring Verlag, Berlin, 2002.

[Johasson(1936)] I. Johasson. Der minimalkalkül, ein reduzierter intuitiostischer formalismus. *Compositio mathematica*, 4:119–136, 1936.

[Kaye(1991)] R. Kaye. *Models of Peano Arithmetic*. Oxford Logic Guides. Oxford University Press, Oxford, 1991.

[Kaye(1993)] R. Kaye. Open induction, tennenbaum phenomena, and complexity theory. In *Arithmetic, proof theory, and computational complexity*, pages 222–237. Oxford University Press, Prague, 1993.

[Kaye(2006)] R. Kaye. Tennenbaum's theorem for models of arithmetic. http://web.mat.bham.ac.uk/R.W.Kaye/papers/tennenbaum/tennenbaum.pdf, 2006.

[Kechris and Moschovakis(1977)] A. Kechris and Y. N. Moschovakis. Recursion in higher types. In J. Barwise, editor, *Handbook of mathematical logic*, pages 681–738. Elsevier, Amsterdam, 1977.

[Kelly(2004)] K. T. Kelly. Uncomputability: the problem of induction internalized. *Theoretical Computer Science*, 317:227–249, 2004.

[Kleene(1935)] S. Kleene. A theory of positive integers in formal logic. *American journal of Mathematics*, (57):53–173, 219–244, 1935.

[Kleene(1936)] S. Kleene. General recursive functions of natural numbers. *Math. Ann.*, (112):727–742, 1936.

[Kleene(1938)] S. Kleene. On notation for ordinal numbers. *Journal of Symbolic Logic*, 4(3):150–155, 1938.

[Kleene(1943)] S. Kleene. Recursive predicates and quantifiers. *Transactions of the American Mathematical Society*, 53(1):41–73, 1943.

[Kleene(1994)] S. C. Kleene. Turing's analysis of computability, and major applications of it. In R. Herken, editor, *The Universal Turing Machine - A Half-Century Survey*, pages 15–49. Springer-Verlag, Wien, 1994.

[Kochen and Kripke(1982)] S. Kochen and S. Kripke. Non-standard models of peano arithmetic. *L'Enseignement Mathématique*, 28:211–231, 1982.

[Kreisel(1965a)] G. Kreisel. Model-theoretical invariants: applications to recursive and hyperarithmetic operations. In e. a. Addison, editor, *Theory of Models*, pages 190–205. North Holland, Amsterdam, 1965a.

[Kreisel(1965b)] G. Kreisel. Mathematical logic. In T. Saaty, editor, *Lectures notes in modern mathematics III*, pages 95–195. Wiley and Sons, New York, 1965b.

[Kripke(1996)] S. Kripke. Elementary recursion theory and its applications to formal systems. unpublished preliminary draft, 1996.

[Kripke(2000)] S. Kripke. From the church-turing thesis to the first-order algorithm theorem. Recorded lecture at http://www.vanleer.org.il/eng/videoShow.asp?id=317 (Mar. 15, 2009), 2000.

[Kurtz et al.(1987)Kurtz, O'Donnell, and Royer] S. Kurtz, M. J. O'Donnell, and S. Royer. How to prove representation-independent independence results. *Information Proc. Lett.*, 24:5–10, 1987.

[Lessan(1978)] H. Lessan. *Models of arithmetic*. PhD thesis, University of Manchester, 1978.

[Levin(1973)] L. Levin. Universal search problems (em russo). *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

[Lewis and Papadimitriou(1998)] H. R. Lewis and C. H. Papadimitriou. *Elements of the theory of computation*. Prentice-Hall, New Jersey, 1998.

[Loff and Costa(2009)] B. Loff and J. F. Costa. Universal search problems (em russo). *Int. Journ. of Unconventional Computing*, 5:193–207, 2009.

[Mateus and Sernadas(2006)] P. Mateus and A. Sernadas. Weakly complete axiomatization of exogenous quantum propositional logic. *Information and Computation*, 204:771–794, 2006.

[Matijasevic(1970)] Y. Matijasevic. Enumerable sets are diophantine. *Soviet Math. Doklady*, (11):354–357, 1970.

[McAllister(2004)] A. M. McAllister. Turing upper bounds of jump ideals and scott sets. In A. Enayat and R. Kossak, editors, *Nonstandard Models of Arithmetic and Set Theory*, volume 361 of *Contemporary Mathematics*, pages 129–143. American Mathematical Society (AMS), New York, 2004.

[McAloon(1982)] A. McAloon. On the complexity of models of arithmetic. *The journal of Symbolic Logic*, 47:403–415, 1982.

[Minsky(1967)] M. L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, New Jersey, 1967.

[Moschovakis(1969)] Y. Moschovakis. Abstract computability and invariant definability. *The journal of Symbolic Logic*, 34(4):605–633, 1969.

[Máté(1990)] A. Máté. Nondeterministic polynomial-time computations and models of arithmetic. *The journal of the ACM*, 37(1):175–193, 1990.

[Nelson(1977)] E. Nelson. Internal set theory: a new approach to nonstandard analysis. *The Bulletin of the American mathematical Society*, 86 (6):1165–1198, 1977.

[Nielson and Chuang(2000)] M. A. Nielson and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2000.

[Noson(2003)] S. Noson. A universal approach to self-referential paradoxes, incompleteness and fixed points. *The Bulletin of Symbolic Logic*, 9: 362–386, 2003.

[Odifreddi(1992a)] P. Odifreddi. *Classical Recursion Theory: The theory of functions and sets of natural numbers, volume I*, volume 125 of *Studies in Logic and the Foundations of Mathematics*. North-Holand, Amsterdam, 1992a.

[Odifreddi(1992b)] P. Odifreddi. *Classical Recursion Theory: The theory of functions and sets of natural numbers, volume II*, volume 125 of *Studies in Logic and the Foundations of Mathematics*. North-Holand, Amsterdam, 1992b.

[Paris and Dimitracopoulos(1982)] J. Paris and C. Dimitracopoulos. Truth definitions for delta-zero formulae. In E. Engeler, H. Läuchli, and V. Strassen, editors, *Logic and Algorithmic*, volume 30 of *L'Enseignement Mathtmatique*, pages 317–329. Elsevier, Genève, 1982.

[Paris and Harrington(1977)] J. Paris and L. Harrington. A mathematical incompleteness in peano arithmetic. In J. Barwise, editor, *Handbook of mathematical logic*, pages 1133–1142. Elsevier, Amsterdam, 1977.

[Peano(2002)] G. Peano. Arithmetices principia, novo methodo exposita. In J. V. Heijenoort, editor, *From Frege to Godel 1879-1931*, Trad., pages 317–329. Harvard University Press, Cambridge, 2002. (1889).

[Post(1944)] E. Post. Recursively enumerable sets of positive integers and their decision problems. *The Bulletin of the American mathematical Society*, 50:284–316, 1944.

[Post(1994)] E. L. Post. Absolutely unsolvable problems and relatively undecidable propositions: Account of an anticipation. In M. Davis, editor, *Solvability, provability, definability: The collected works of Emil L. Post*, volume 30 of *L'Enseignement Mathtmatique*, pages 375–441. Birkhäuser, Boston, 1994.

[Potgieter and Rosinger(2008)] P. H. Potgieter and E. E. Rosinger. Ultrafilter and non-standard turing machines. volume 5204 of *Lectures Notes in Computer Science*, pages 220–227. 2008.

[Potgieter and Rosinger(2010)] P. H. Potgieter and E. E. Rosinger. Output concepts for a accelerated turing machines. *Natural Computing*, 9(4): 853–864, 2010.

[Razborov and Rudich(1997)] A. A. Razborov and S. Rudich. Natural proofs. *journal of Computer and Systems Sciences*, 55:24–35, 1997.

[Remmel and Crossley.(1993)] J. B. Remmel and J. N. Crossley. The work of anil nerode: a retrospective. In J. B. Remmel, J. N. Crossley, R. A. Soare, and M. E. Sweedler, editors, *Logical methods: In Honor of Anil Nerode's Sixtieth Birthday*, pages 1–. Birkhäuser, New Jersey, 1993.

[Richter and Szabo(1988)] M. Richter and M. Szabo. Nonstandard methods in combinatorics and theoretical computer science. *Studia Logica*, 47: 181–191, 1988.

[Richter and Szabo(1983)] M. M. Richter and M. E. Szabo. Towards a nonstandard analysis of programs. In A. E. Hurd, editor, *Nonstandard Analysis - Recent Developments*, volume 983 of *Lecture Notes in Mathematics*, pages 186–203. 1983.

[Robinson(1966)] A. Robinson. *Non-standard analysis*. Princeton Landmarks in Mathematics. Princeton University Press, New Jersey, 1966.

[Rosser(1936)] J. B. Rosser. Extensions of some theorems of gödel and church. *The journal of Symbolic Logic*, 1:87–91, 1936.

[Russell and Whitehead(1910, 1912, 1913)] B. Russell and A. N. Whitehead. *Principia Mathematica*, volume 1-3. Cambridge University Press, Cambridge, 1910, 1912, 1913.

[Savitch(1970)] W. J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *journal of Computer and System Sciences*, 4:177–192, 1970.

[Schindler(2003)] R. Schindler. $p \neq np$ for infinite time turing machines. *Monatshefte für Mathematik*, 139(4):335–340, 2003.

[Scott(1962)] D. Scott. Algebras of sets binumerable in complete extensions of arithmetic. In J. C. E. Dekker, editor, *Recursive function theory*, volume V of *American Mathematical Society Proceedings of Symposia in Pure Mathematics*, pages 117–122. 1962.

[Shamir(1992)] A. Shamir. Ip=pspace. *The journal of the ACM*, 39(4):869–877, 1992.

[Shapiro(1991)] S. Shapiro. *Foundations without Foundationalism: A case for Second-order Logic*, volume 17 of *Oxford Logic Guides*. Clarendon Press, Oxford, 1991.

[Shapiro(1997)] S. Shapiro. *Philosophy of Mathematics: Structure and Ontology*. Oxford Logic Guides. Oxford Univeresity Press, Oxford, 1997.

[Shöenfield(1993)] J. Shöenfield. *Recursion theory*, volume 1 of *Lectures Note in Logic*. Springer-Verlag, Berlin, 1993.

[Shepherdson(1964)] J. C. Shepherdson. A non-standard model for a free variable fragment of number theory. *Bulletin de l'Académie Polonaise des*

Sciences. Série des Sciences Mathématiques, Astronomiques et Physiques, 12:79–86, 1964.

[Shi(2002)] Y. Shi. Remarks on universal quantum computer. *Physics Letters A*, 293(5-6):277–282, 2002.

[Shore(2006)] R. A. Shore. Degree structures: Local and global investigations. *The Bulletin of Symbolic Logic*, 12(3):369–389, 2006.

[Sieg(2002)] W. Sieg. Calculations by man and machine: Conceptual analysis. In W. Sieg, R. Sommer, and C. Talcott, editors, *Reflections on the Foundations of Mathematics*, volume 15 of *Lecture Notes in Logic*, pages 390–409. Association for Symbolic Logic, Nantick, 2002.

[Sieg(2006)] W. Sieg. Gödel on computability. *Philosophia Mathematica*, 14 (3):189–207, 2006.

[Simpson(2009)] S. G. Simpson. *Subsystems of second order arithmetic*. Perspectives in Logic (2nd ed.). Cambridge University Press, Cambridge, 2009.

[Sipser(1992)] M. Sipser. The history and status of the p versus np question. In *Proceedings of ACM STOC'92*, volume V, pages 603–618. 1992.

[Sipser(2006)] M. Sipser. *Introduction to the theory of computation*. 2nd ed. Thomson Course Technology, Massachusetts, 2006.

[Skolem(1922)] T. Skolem. Einige bemerkugen zur axiomatischen begründung der mengenlehere. *Proceeding of the 5th Scan Math Congress at Helsinki*, pages 217–232, 1922.

[Skolem.(1929)] T. Skolem. Über einige grundlagenfragen der mathematic. *Skrifter, Vitenskapsakademiet i Oslo*, I(4):1–49, 1929.

[Skolem.(1934)] T. Skolem. Über die nichtcharakterisierbarkeit der zahlenreihe mittels endlich oder abzählbar unendlich vielen assagen mit asschliesslich zahlvariablen. *Fundamenta Mathematicae*, 23:150–161, 1934.

[Soare(1996)] R. Soare. Computability and recursion. *The Bulletin of Symbolic Logic*, 2(3):284–321, 1996.

[Soare(1977)] R. I. Soare. α-recursion theory. In J. Barwise, editor, *Handbook of mathematical logic*, pages 653–680. Elsevier, Amsterdam, 1977.

[Soare(1987)] R. I. Soare. *Recursively Enulllerable Sets and Degrees*. (Perspectives in mathematical logic. Springer-Verlag, Berlin, 1987.

[Soare(1999)] R. I. Soare. The history and concept of computability. In E. R. Griffor, editor, *Handbook of Computability Theory*, volume 140 of *Studies in Logic and the Foundations of mathematics*, pages 3–37. Elsevier, Amsterdam, 1999.

[Stegmüller(1976)] W. Stegmüller. *Hauptströmunger der Gegenwartsphilosophie*, volume I-II. Alfred Kröner Verlag Stuttgart, Tübingen, 1976.

[Steinhart(2002)] E. Steinhart. Logically possible machines. *Minds and Machines*, 12:259–280, 2002.

[Stockmeyer(1976)] L. Stockmeyer. The polinomial time hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.

[Sureson(1995)] C. Sureson. Np vs conp and models of arithmetic. *Theoretical Computer Science*, (147):55–67, 1995.

[Sylvan and Copeland(2000)] R. Sylvan and J. Copeland. Computability is logic-relative. In G. Priest and D. Hyde, editors, *Sociative Logics and Their Applications: Essays by the Late Richard Sylvan*, American Mathematical Society Proceedings of Symposia in Pure Mathematics, pages 189–199. Ashgate Publishing Company, 2000.

[Tarski(1983)] A. Tarski. The concept of truth in formalized languages (1933). In A. Tarski, editor, *Logic, Semantics, Metamathematics*, American Mathematical Society Proceedings of Symposia in Pure Mathematics, pages 152–278. Hackett Publishing, Indianapolis, 1983. 2ª ed.

[Tennenbaum(1959)] S. Tennenbaum. Non-archimedian models for arithmetic. *Notices of the American Mathematical Society*, 6:270, 1959.

[Turing(1936)] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265, 1936.

[Turing(1937)] A. M. Turing. Computability and lamba-definability. *journal of Symbolic Logic*, 2:153–163, 1937.

[Turing(1939)] A. M. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, 45:161–228, 1939.

[van Oosten(2008)] J. van Oosten. *Realizability: an introduction to its categorical side*, volume 152 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, Amsterdam, 2008.

[Wilmer(1985)] G. Wilmer. Bounded existential induction. *The journal of Symbolic Logic*, 50:72–90, 1985.

[Yao(1993)] A. C. Yao. Quantum circuit complexity. In *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, American Mathematical Society Proceedings of Symposia in Pure Mathematics, pages 352–360. IEEE Computer Society Press, Los Alamitos, 1993.

[Ying(2005)] M. Ying. A theory of computation based on quantum logic. *Theoretical Computer Science*, 344:134–207, 2005.

[Zach(2003)] R. Zach. Hilbert's program. In E. Zalta, editor, *Stanford Encyclopedia of Philosophy*. 2003. http://plato.stanford.edu/entries/hilbert-program/.